EdgeAdaptor: Online Configuration Adaption, Model Selection and Resource Provisioning for Edge DNN Inference Serving at Scale

Kongyange Zhao, Zhi Zhou, *Member, IEEE*, Xu Chen, *Senior Member, IEEE*, Ruiting Zhou, *Member, IEEE*, Xiaoxi Zhang, *Member, IEEE*, Shuai Yu, *Member, IEEE*, and Di Wu, *Senior Member, IEEE*

Abstract—The accelerating convergence of artificial intelligence and edge computing has sparked a recent wave of interest in edge intelligence. While pilot efforts focused on edge DNN inference serving for a single user or DNN application, scaling edge DNN inference serving to multiple users and applications is however nontrivial. In this paper, we propose an online optimization framework EdgeAdaptor for multi-user and multi-application edge DNN inference serving at scale, which aims to navigate the three-way trade-off between inference accuracy, latency, and resource cost via jointly optimizing the application configuration adaption, DNN model selection and edge resource provisioning on-the-fly. The underlying long-term optimization problem is difficult since it is NP-hard and involves future uncertain information. To address these dual challenges, we fuse the power of online optimization and approximate optimization into a joint optimization framework, via i) decomposing the long-term problem into a series of single-shot fractional problems with a regularization technique, and ii) rounding the fractional solution to a near-optimal integral solution with a randomized dependent scheme. Rigorous theoretical analysis derives a parameterized competition ratio of our online algorithms, and extensive trace-driven simulations verify that its empirical value is no larger than 1.4 in typical scenarios.

Index Terms—Edge intelligence, edge computing, DNN inference serving, online optimization

1 Introduction

Driven by the burgeoning as well as accelerating convergence of artificial intelligence (AI) and Internet-of-Things (IoT), we have recently witnessed an unprecedented booming of AI-of-Things or AI-empowered IoT applications. This new trend is known as AIoT [1], which has gained sparking interest from both academia and industrial. To materialize the vision of AIoT, edge intelligence has emerged as an enabling paradigm to address the last mile delivery issue faced by AI [2]. With edge intelligence, data- and computation-intensive AI tasks are pushed from the centralized cloud to the network edges to serve the ubiquitous IoT users and devices in closer proximity. This closer proximity spurs prominent benefits including reduced transmission delay, energy consumption, and wide-area-network (WAN) bandwidth usage [3].

While recognizing the prominent advantages of edge intelligence, we should also note that edge nodes are typically provisioned with limited resources (e.g., low-end and weak CPU/GPU). Such resource limitations deteriorate the performance of state-of-the-art AI models based on deep neural network (DNN) (e.g., VGG [4] and ResNet [5] for computer vision tasks), since the latter is far more resource-hungry and delay-sensitive in AIoT scenarios. To address this performance degradation incurred by the mismatch between resource supply and demand, *model compression* is widely applied to practical resource-constrained edge deployment [6], [7]. After applying model compression methods, such as model weight pruning and quantization [8], the compressed model is with less number of weights, smaller model size, and thus less resource consumption as shown in Fig. 1. In orthogonal to model compression, *application configuration adaption* can be also exploited to reduce the resource footprint of DNN inference serving. For computer vision tasks with image input, the resolution of the image can be degraded (e.g., from 720p to 240p as shown in Fig. 1) to accelerate the model inference [9].

Notably, pilot efforts on edge intelligence have mostly focused on energy-efficient DNN inference serving for a single user or application [10], [11], [12], the problem of edge DNN inference serving to multi-user and multi-application deployment at scale has been largely overlooked. Orchestrating DNN inference serving for multiple users and

- *Kongyange Zhao, Zhi Zhou, Xu Chen, Xiaoxi Zhang, Shuai Yu, and Di Wu are with the School of Computer Science and Engineering, Sun Yat-sen University (SYSU), Guangzhou 510006, China. E-mail: zhaokyg@mail2.sysu.edu.cn, {zhouzhi9, chenxu35, zhangxx89, yushuai, wudi27}@mail.sysu.edu.cn.*
- *Ruiting Zhou is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China. E-mail: ruitingzhou@whu.edu.cn.*

Manuscript received 23 November 2021; revised 17 May 2022; accepted 22 June 2022. Date of publication 7 July 2022; date of current version 31 August 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 62172454, 62072344, U20A20159, 61972432, 62102460, 62002397, U20A20177, and U1911201, in part by the Guangdong Basic and Applied Basic Research Foundation under Grants 2021B151520008 and 2021A1515011912, and in part by the Science and Technology Planning Project of Guangdong Province under Grants 2018B030322004 and 2021 A0505110008.
(Corresponding author: Zhi Zhou.)
Digital Object Identifier no. 10.1109/TMC.2022.3189186

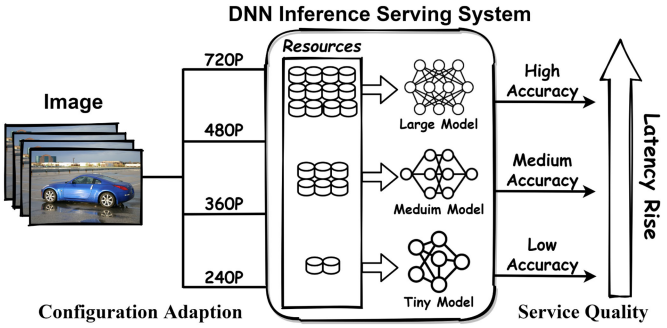

Fig. 1. An illustration of application configuration adaption and model compression for edge DNN inference serving.

applications is however nontrivial due to the following difficulties. For each application, its request can be served by multiple DNN models. Take the application of object detection task as an example, its request can be served by DNN model such as YOLO [13], SSD [14], or R-FCN [15], and such different models have a diverse resource-accuracy tradeoff. While for different applications, their requests can also share the same DNN model. For traffic analytic applications such as object counting (e.g., pedestrians, cars, bikes), traffic violations detection (e.g., jaywalking), and collision analysis (e.g., the collision between vehicles and pedestrians), they can share the same model instance (e.g., a container or a VM) for object detection to reduce the resource cost. To balance accuracy improvement brought by different models and resource cost saved by model sharing, how to choose the best model for each application and dynamically provision the number of model instances for each DNN model? Besides, launching a new model instance (e.g., a container or a VM) would incur instance switching cost which is associated with the hardware wear-and-tear. When the request arrival diminishes, destroying an idle model instance does not necessarily improve the cost-efficiency, as it would surge the instance switching cost if the request arrival grows shortly after.

To address the above difficulties, in this paper we advocate an online optimization framework EdgeAdaptor for joint configuration adaption, model selection and resource provisioning for cost-efficient edge DNN inference serving at scale. It minimizes the long-term resource cost and cumulative accuracy loss under predefined real-time performance requirements. In particular, such a problem can be formulated as a *mixed integer linear programming* (MILP). However, solving the above MILP problem is rather challenging. On the one hand, as the switching cost is incurred when launching new model instances, the resource provisioning decisions are temporally coupled over consecutive time slots. This temporal correlation makes the long-term cost minimization time-coupling and involving future system information. On the other hand, even with an offline setup in which the future system parameters are given beforehand, the long-term problem is still proven to be NP-hard.

By exploiting the structural properties of the problem, we simultaneously cope with the above dual challenges via fusing the regularization method for online algorithm design and a dependent rounding technique for approximation algorithm design. In particular, by leveraging the regularization method from the online learning literature [16], we first relax the integer constraint and regularize the time-coupling term in the long-term problem. Next, we temporally decompose it into a series of single-shot fractional subproblems that do not involve future information and thus can be readily solved. To maintain the feasibility of the fractional solution, we further round it to a feasible integer solution by carefully designing a randomized dependent rounding scheme. Essentially, at the core of the dependent rounding scheme, is to compensate each rounded-down model instance variable by another rounded-up model instance variable. Since the compensation avoids provisioning excessive model instances, the dependent rounding scheme enables a significant reduction of the total cost, and meanwhile maintains the feasibility of the solution.

Our main contributions are highlighted as follows.

- We cast the joint problem of configuration adaption, model selection, and resource provisioning for edge DNN inference serving as a mixed integer linear programming, which judiciously arbitrates the three-way tradeoff between resource cost, performance (i.e., service latency), and inference accuracy.
- We propose a regularization-based algorithm to address the challenge of uncertain future information by time-coupling switching cost. Through a carefully designed randomized dependent rounding scheme, the feasibility of the solution is maintained without introducing expensive cloud outsourcing costs.
- We rigorously analyze and derive a parameterized competitive ratio for our proposed online algorithms, by incorporating the competitive ratio of the online regularization method as well as the rounding gap of the randomized dependent rounding scheme. Extensive experiments driven by realistic workload traces further verify that the empirical value of the parameterized competitive ratio is no larger than 1.40 in typical scenarios.

The rest parts of the paper are organized as follows. Section 2 reviews the related literature on edge intelligence model inference. Section 3 introduces the system model and problem formulation for cost-efficient edge DNN inference serving at scale. Section 4 presents an online optimization algorithm and we analyze its worst-case performance rigorously in Section 5. Section 6 conducts extensive trace-driven simulations to empirically assess the performance of the proposed online optimization algorithm. Finally, Section 7 concludes this paper.

2 Related Work

To push the AI capabilities from the centralized cloud to the network edge which is in closer proximity to the widespread IoT users and devices, an essential problem is how to facilitate low-latency and energy-efficient DNN model inference at edge nodes that are typically limited by both energy and resource capacity. To answer this problem in the positive, Neurosurgeon [17] attempts to partition the DNN model layers into two parts and offloading the more resource-demanding part from the IoT device to the powerful cloud for inference acceleration. Following Neurosurgeon, Edgent [10] augments model right-sizing to model partitioning to further reduce the inference latency. By

TABLE 1
Main Notations

Notation	Describtion
$\mathcal{I}, \mathcal{J}, \mathcal{K}, \mathcal{T}$	sets of applications, models, configurations, and time slots
i, j, k, t	indexes of applications, models, configurations, and time slots
$A_i(t), L_i$	workload and average latency requirement for application i in time slot t
e_j, E_j	workload capacity and resource capacity for DNN model j
a_{ijk}	accuracy loss for application i served by model j with configuration k
d_{jk}, f_{jk}	inference latency and resource consumption for model j serving with configuration k
$b_j(t), c_i, s_j$	instance operational cost, unit cloud outsourcing cost, and switching cost

pre-training a master AI model embedding with multiple sizes (each size corresponds to a branch of the master DNN), Edgent adaptively selects the best model size to optimize the latency-accuracy tradeoff against varying computation resources and network bandwidth.

For model partitioning approaches, the vulnerable wireless network link between the device and the edge server may become the performance bottleneck of model inference. To cope with this issue, the recent works CLIO [11] and SPINN [12] further compress the DNN layer corresponds to the partitioning points, and thus reduce the transmission latency of the intermediate model data transfer between the device and the edge server.

Different from the above research attempts which relieves resource mismatch at the model serving side, another stream of recent efforts exploit configuration adaption at the application request side to accelerate DNN inference. For computer vision-based applications such as video analytics and image recognition, the application configuration — frame rates and resolution ratio for videos, compression ratio and frame size for images — can be adaptively optimized to balance the tradeoff between inference latency, accuracy, and resource quota [18]. For large-scale edge-coordinated video analytics, Yang *et al.* [19] jointly optimize the video configuration and edge resource allocation to maximize the long-term inference accuracy, while satisfying the real-time delay requirement of various video streams, by applying deep reinforcement learning (DRL). For the real-world implementation, Crankshaw *et al.* [20] propose a inference serving system with a two-layer architecture, which contains the model selection and the model abstraction to achieve low latencies, high throughputs, and improved accuracy. And Francisco *et al.* [21] propose a distributed inference serving system, which selects appropriate model variants for each query with different service level objectives (SLOs), i.e., latency requirements, to achieve a trade-off between accuracy, latency and cost.

While existing efforts on edge intelligence has mostly focused on energy-efficient DNN inference serving for a single user or DNN application, the problem of scaling edge DNN inference serving to multi-user and multi-application deployment has been largely overlooked. Recently, Wu *et al.* [22] investigate collaborative DNN inference between device and edge server for multiple AI services. To minimize the service delay under a time-averaged accuracy requirement, the authors apply DRL method to jointly optimize the knobs of task sampling rate selection, task offloading, and edge computing resource allocation. Our work is different from [22], since the resource allocation decision is formulated as a set of continuous rather than discrete variables, and thus the optimization problem does not necessarily to be NP-hard. Besides, the DRL method is difficult to converge during the training phase, especially for continuous variables with a large action space, which is still an open problem at present. Our online algorithm can directly obtain the online decision without the training phase and provide performance guarantees through rigorous theoretical analysis compared to DRL method. The most related work to us is probably [23], in which Wang *et al.* leverage Lyapunov optimization to jointly optimize the video configuration and network bandwidth allocation to balance the three-way tradeoff between accuracy, latency and energy consumption. Our work is different from and complementary to [23] in at least the following three aspects. First, we further consider model selection and model sharing to optimize the holistic system cost. Second, unlike [23] assumes fixed amount of computation resource, we further considers the problem of dynamical resource provisioning. Finally, in [23], the challenge of time-coupling is incurred by the long-term time-averaged latency constraint, while in our work, the challenge of time-coupling is incurred by the switching cost of dynamically resource provisioning. Note that however, Lyapunov optimization is unable to address the time-coupling term in the objective function and thus is not applicable to our problem.

3 System Model and Problem Formulation

In this section, we present the system model for multi-application and multi-model edge inference serving at scale, and formulate the problem which navigates the three-way tradeoff between inference accuracy, latency, and resource cost. Table 1 lists the main notations in our paper.

3.1 Overview of Edge DNN Inference Serving System

We consider a realistic edge intelligence paradigm as illustrated in Fig. 2. The edge server which is typically attached to an access point serves diverse emerging AIoT applications as exemplified by smart retail, intelligent transportation and smart factory. These AIoT applications persistently sense the surroundings of IoT devices and generate huge amounts of multi-modal data such as streams of videos, audios and images. To extract knowledge from the sensed data, it is required to perform model inference (such as VGG [4], YOLO [24] for object recognition in intelligent transportation). Considering the resource scarcity of the IoT devices, such DNN model inference service is deployed at

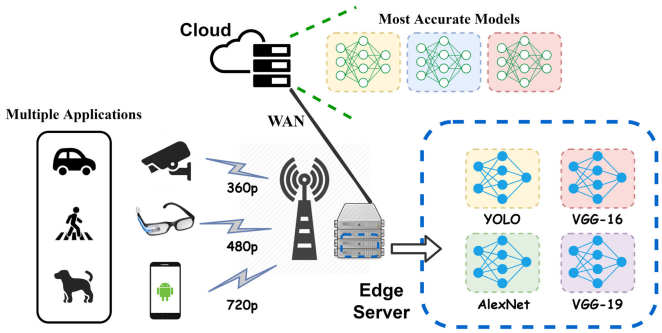

Fig. 2. An illustration of multi-application and multi-model edge DNN inference serving system.

the edge servers (e.g., Huawei Atlas 500 Edge Server) which is equipped with moderate computing capacity and AI accelerators such as GPU and NPU (neural processing units). To fully utilize the expensive edge server resources, an edge server is typically virtualized into a set of containers or virtual machines (VM) to co-locate multiple DNN models to serve diverse AIoT applications.

Based on the above setup, we use a set $\mathcal{I} = \{1, 2, \ldots, I\}$ to denote diverse AIoT applications served by the edge server. Considering the fact that the inference requests of some diverse applications can share the same DNN model, and the inference requests of a specific application can be even served by different DNN models, we use another set $\mathcal{J} = \{1, 2, \ldots, J\}$ to denote the DNN models held on the edge server. The model set $\mathcal{J}$ contains different types of DNN models (e.g., VGG [4], ResNet [5]), as well as model variants with different sizes (e.g., VGG-16 and VGG-19), as a result of the aforementioned model compression technique in the second paragraph of Section 1. Without loss of generality, to capture the system dynamics such as time-varying inference request arrivals, we assume that the edge DNN inference serving system works in a time-slotted fashion within a large time span of T. Each time slot of $t \in \mathcal{T} = \{1, 2, \ldots, T\}$ represents a decision interval that matches the change of the system dynamics. In practice, the length of a time slot (i.e., 10 minutes) is much longer than a typical end-to-end delay of DNN model inference. Then we use $A_i(t)$ to denote the number of inference requests generated by the AIoT application $i \in \mathcal{I}$. For each DNN model $j \in \mathcal{J}$ we assume that each DNN model $j \in \mathcal{J}$ is instantiated in a VM or container on the edge server, and this VM or container running a DNN model is referred to a DNN model instance. Determined by the resource capacities of the underlying physical resources (e.g., CPU, GPU and memory), the workload capacity of each instance of DNN model $j \in \mathcal{J}$ in each time slot is denoted as e_j, i.e., the amount of CPU FLOPS can be offered by an instance of DNN model $j \in \mathcal{J}$ in each time slot. Considering the limited amount of the resources on edge server, we further use E_j to denote the resource capacity, i.e., the maximal number of available instances of DNN model $j \in \mathcal{J}$ on the edge server.

In serving a specific inference request, different DNN models offer varying resource-latency-accuracy tradeoff. Therefore, for each inference request, we should choose the best model to improve the accuracy with low inference latency and resource footprint. In addition to model selection, application configuration adaption [18] is also widely applied to flexibly navigate the resource-latency-accuracy tradeoff in edge computing paradigm. For example, for inference request with image input, the resolution can be adapted (e.g., from 1080p to 480p) to tune the resource consumption, inference latency and accuracy. While for inference request with video input, the frame rate can be further adjusted (e.g., from 30fps to 10fps) to balance the resource-latency-accuracy tradeoff. In this paper, we use a set $\mathcal{K} = \{1, 2, \ldots, K\}$ to denote the set of configurations can be chosen for each application. Then, to model the resource-latency-accuracy tradeoff, we use a_{ijk}, d_{jk} and f_{jk} to denote the accuracy loss, inference latency and resource consumption (i.e., the CPU FLOPS) of using DNN model $j \in \mathcal{J}$ to serve the inference request of application $i \in \mathcal{I}$ with configuration $k \in \mathcal{K}$.

For the envisioned edge DNN inference serving system as shown in Fig. 2, it also incorporates collaborative model inference between the edge server and the remote public cloud, i.e., an inference request can be served either by the edge server or the public cloud, due to the following considerations. First, the price of the edge resource typically fluctuates over time, since the time-varying electricity price dominates the operational cost. This indicates that when the edge resource price peaks, we can offload the inference request to the public cloud for processing. Second, the arrival of inference requests also fluctuates over time, which may overwhelm the capacity of the edge server. For each inference request offloaded to the cloud, it is often served by the most accurate but resource-demanding model, due to the powerful computing capacity of the cloud.

3.2 Decision Variables

For the edge DNN inference serving system, its primary objective is to jointly optimize the inference accuracy, latency and monetary cost. Towards this goal, it should judiciously exploit the model heterogeneity as well as resource heterogeneity (i.e., edge versus cloud) to make the following decisions: *(i)* application configuration adaption, for each application type of inference requests, which configurations (e.g., the resolution of a input image) it should choose? *(ii)* DNN model selection, for the inference requests of each application, which set of DNN models should be chosen to serve them? *(iii)* resource provisioning, for each DNN model, how many instances should be provisioned for it? We now formulated the above decisions in the following.

Joint Configuration Adaption and Model Selection. While it is feasible to introduce two independent variables of configuration adaption and model selection for each inference request, it would greatly complicate our problem. Instead, a more straightforward and simple way is to use a joint configuration adaption and model selection variable $x_{ijk}(t)$ to denote the amount of inference request of application $i \in \mathcal{I}$ and with configuration $k \in \mathcal{K}$ to be served by DNN model $j \in \mathcal{J}$ at time slot t. Note that this simplification benefits from the identity of different inference requests by a given application type. Furthermore, given the potentially huge amount of inference request arrival $A_i(t)$, the decision variable $x_{ijk}(t)$ can be relaxed to a continuous variable. Such relaxation is in consistent with the recent literature on data center power management [25]. Let $\mathbf{x}(t) = \left(x_{ijk}(t)\right)_{\forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T}}$ be the

set of joint configuration adaption and model selection variables, then it satisfies:

$$\mathcal{X} = \Big\{\mathbf{x}(t) \,|\, \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} x_{ijk}(t) \le A_i(t) \text{ and } x_{ijk}(t) \ge 0, \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall k \in \mathcal{K}, \forall t \in \mathcal{T}\Big\}. \quad (1)$$

Here $x_{ijk}(t) \le A_i(t)$ indicates that part of the inference requests can be offloaded to the cloud.

Resource Provisioning. We use $y_j(t)$ to denote the number of model instances launched for DNN model $j \in \mathcal{J}$ at time slot t. Since the resource capacity on edge server is highly limited, it is impractical to relax the non-negative integer $y_j(t)$ to a continuous one. Instead we enforce that $y_j(t) \in \{0, 1, \ldots, E_j\}$, where E_j is the aforementioned maximal number of available instances for DNN model $j \in \mathcal{J}$. Let $\mathbf{y}(t) = \big(y_j(t)\big)_{\forall j\in\mathcal{J}, \forall t\in\mathcal{T}}$ be the set of feasible resource provisioning variables, then it satisfies:

$$\mathcal{Y} = \Big\{\mathbf{y}(t) \,|\, y_j(t) \in \{0, 1, \ldots, E_j\}, \forall j \in \mathcal{J}, \forall t \in \mathcal{T}\Big\}. \quad (2)$$

3.3 Cost Structure

Given the above decision variables, we are now ready to formulate the overall cost incurred by the edge DNN inference serving system, which includes the edge operational cost, cloud outsourcing cost and instance switching cost.

Edge Operational Cost. Edge service providers rent server resources to run DNN model instances, and multiple inference requests in the same time slot will be processed concurrently in a model instance. For an edge server, it is widely recognized that its operational cost is dominated by the energy cost and the amortized capital expenditure [26]. Moreover, since the electricity price in the real-time electricity market typically fluctuates over time, we use $b_j(t)$ to denote the operational cost of running an instance of DNN model $j \in \mathcal{J}$ at time slot t. It contains the processing cost of multiple inference requests and is jointly determined by DNN size and electricity price. Then the total edge operational cost in each time slot t is given by:

$$C_{EO}(\mathbf{y}(t)) = \sum_{j\in\mathcal{J}} b_j(t) y_j(t). \quad (3)$$

Cloud Outsourcing Cost. As we discussed in Section 3.1, the edge server itself is unable to serve all the inference request arrival when the latter burst suddenly, or un-economical to serve all the inference request arrival when the real-time electricity price peaks and overweighs the cost of the cloud. In these two cases, the edge DNN inference serving system can exploit the cloud resource to improve robustness and cost-efficiency. Following the recent trend on large-scale DNN model inference serving with serverless resource in the cloud [27], we also consider serving the outsourced inference request with serverless (e.g., AWS Lambda and Google Cloud Functions). A salient feature of serverless is that it is charged in a "pay-as-you-go" manner, i.e., we pay for the actual resource usage rather than the resource occupation (like VM). Considering the powerful computing capacity of the cloud, we assume that the outsourced inference requests are served by the most accurate model under the highest application configuration. Here we use c_i to denote the aggregated cost of outsourcing one inference request of application $i \in \mathcal{I}$ from the edge server to the remote central cloud. The unit cloud outsourcing cost c_i unifies the processing cost (resource usage) of the remote cloud, the transmission cost (bandwidth usage) and the performance penalty incurred by the latency of the WAN connecting to the remote cloud. Measurements by internet giants suggest that there is the numerical relationship between service latency and price [28]. Given the amount $A_i(t) - \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} x_{ijk}(t)$ of inference requests of application $i \in \mathcal{I}$ outsourced to the public cloud from edge server, the total cloud outsourcing cost in time slot t can be computed by:

$$C_{CO}(\mathbf{x}(t)) = \sum_{i\in\mathcal{I}} c_i \left[A_i(t) - \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} x_{ijk}(t)\right]. \quad (4)$$

Instance Switching Cost. Note that when we launch new models instance when updating the instance provisioning decisions, another type of cost which is referred to as instance switching cost would be incurred. Specifically, launching a new instance in practical systems (e.g., Docker) involves first loading the image file and booting it to a new VM instance, and then loading the model parameters to the instance. This process not only incurs additional delay and energy consumption for preparing resources but also brings hardware wear-and-tear which is often very costly. In practice, the instance switching cost can be on a par with the edge operation cost [29]. We use s_j to denote the cost of launching a new instance of DNN model $j \in \mathcal{J}$, then the total instance switching cost at time slot t is given by:

$$C^t_{IS}(\mathbf{y}(t), \mathbf{y}(t-1)) = \sum_{j\in\mathcal{J}} s_j \big[y_j(t) - y_j(t-1)\big]^+, \quad (5)$$

where $\big[y_j(t) - y_j(t-1)\big]^+ = \max\big\{y_j(t) - y_j(t-1), 0\big\}$, denoting the number of newly launched instances of DNN model $j \in \mathcal{J}$ in time slot t. Without loss of generally, we let $y_j(0) = 0$.

In summary, the overall cost incurred in time slot t in the system is:

$$\mathbb{C}(\mathbf{x}(t), \mathbf{y}(t)) = C_{EO}(\mathbf{y}(t)) + C_{CO}(\mathbf{x}(t)) + C^t_{IS}(\mathbf{y}(t), \mathbf{y}(t-1)). \quad (6)$$

3.4 Inference Accuracy and Latency

Application configuration adaption and DNN model selection pose a three-way tradeoff among inference accuracy, latency and resource consumption. Recall that in Section 3.1 we use a_{ijk} and d_{jk} to denote the accuracy loss and inference latency when using DNN model $j \in \mathcal{J}$ to serve the inference request of application $i \in \mathcal{I}$ with configuration $k \in \mathcal{K}$. Moreover, since we assume that the outsourced inference requests are served by the most accurate model under the highest configuration, their accuracy loss is thus zero. Then the total accuracy loss of all inference request at each time slot t can be expressed as:

$$Acc(t) = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} a_{ijk} x_{ijk}(t). \quad (7)$$

In practice, requests' latency typically follows the "long-tail" effect [30], [31], and a predefined hard-deadline for each request is non-trivial to satisfy, due to the unpredictable reliability issues of the underlying hardware platform as well as the operating system. At the edge, the latency distribution has a longer tail than the cloud, and it has shown that given the same latency constraints, the average latency guarantee is more beneficial to improve CPU utilization [32]. For different AIoT applications, they usually have different latency requirements. For example, object detection tasks for dynamic recognition (e.g., pedestrian or vehicle) often have a high real-time requirement. Therefore, we use L_i to denote the average inference latency requirements of each applications $i \in \mathcal{I}$ in each time slot. Therefore, the latency constraint condition on edge server can be expressed as $\frac{\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} x_{ijk}(t)d_{jk}}{\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} x_{ijk}(t)} \leq L_i, \forall i \in \mathcal{I}$. For inference requests processed by cloud outsourcing, the latency constraint is implicitly concluded in the unit cost c_i. Specifically, applications with a stricter latency requirements have a higher unit cost.

3.5 Problem Formulation

By jointly consider the holistic cost, inference accuracy and latency, we aim at minimizing the long-term holistic cost and inference accuracy loss over the time horizon T, and under the real-time latency requirement L_i. Formally, this problem can be cast as:

$$
\begin{aligned}
\mathsf{P}: \quad \min \quad & \sum_{t=1}^{T}\Big(\mathbb{C}(\mathbf{x}(t),\mathbf{y}(t)) + \omega \cdot Acc(t)\Big), \\
\text{s.t.} \quad & \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} x_{ijk}(t) \leq A_i(t), && \forall t, \forall i, && \text{(8a)} \\
& \sum_{i\in\mathcal{I}}\sum_{k\in\mathcal{K}} x_{ijk}(t) f_{jk} \leq y_j(t) e_j, && \forall t, \forall j, && \text{(8b)} \\
& \frac{\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} x_{ijk}(t) d_{jk}}{\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} x_{ijk}(t)} \leq L_i, && \forall t, \forall i, && \text{(8c)} \\
& x_{ijk}(t) \geq 0, && \forall t, \forall i, \forall j, \forall k, && \text{(8d)} \\
& y_j(t) \in \{0, 1, \ldots, E_j\}, && \forall t, \forall j. && \text{(8e)}
\end{aligned}
$$

The weighted parameter ω controls the tradeoff between resource cost and inference accuracy. Constraint (8a) ensures that the amount of requests processed on edge server do not exceed the total number of inference requests received in each time slot. Constraint (8b) is the workload capacity constraint which enforces that for each DNN model j, the number of inference requests processed by DNN model j does not exceed the provisioned processing capacity $y_j(t)e_j$. Constraint (8c) enforces that the average edge-side latency of each application i does not exceed the performance threshold L_i. Constraint (8d) is the non-negative constraint for the decision variables. Finally, constraint (8e) is the integer constraint for the number of launched DNN model instances.

Solving the above optimization problem faces the following challenges: *1)* The long-term optimization problem P is a time-coupling problem that involves future system dynamics which typically fluctuates over time and thus hard to precisely estimate. *2)* Even if the future information is known as a priori knowledge, the minimization problem P is also NP-hard. Specifically, we can reduce the classical *minimum knapsack problem* (MKP) [33] which is known to be NP-hard to our problem. The detailed reduction from the MKP can be found in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2022.3189186.
These challenges call for an online approach that can jointly optimize configuration adaption, model selection and resource provision for application requests without requiring future information.

4 Online Algorithm Design

We now present the online optimization framework of EdgeAdaptor. Specifically, to address the dual challenges of time-coupling effect and NP-hardness of long-term objective value minimization problem P, we first relax the integer variables $y_j(t)$ and regularize the switching cost C_{IS}^t, then decouple the long-term problem into a series of single-shot fractional problems. These problems can be easily solved through our fractional algorithm in Section 4.1. In order to satisfy the integer constraints in reality, we design a rounding algorithm in Section 4.2 to obtain a near-optimal solution to the original problem with a bounded optimality gap. Finally, we use the characteristics of the rounding algorithm to obtain the re-selected model variables to meet the capacity constraints changed after the rounding scheme in Section 4.3.

4.1 Fractional Algorithm for Joint Configuration Adaption and Model Selection

In order to overcome the challenge of the NP-hardness of mixed-integer programming problem, we first relax the integer constraint (8e), obtaining the fractional optimization problem $\mathsf{P_R}$ as follow:

$$
\begin{aligned}
\mathsf{P_R}: \quad \min \quad & \sum_{t\in\mathcal{T}}\Big[\omega \cdot Acc(t) + C_{EO}(t) + C_{CO}(t) + C_{IS}^t\Big], \\
\text{s.t.} \quad & \text{Constraint (8a) to (8d)}, \\
& y_j(t) \in [0, E_j], \ \forall t \in \mathcal{T}, \forall j \in \mathcal{J}.
\end{aligned}
$$

To solve the above long-term fractional problem $\mathsf{P_R}$, a natural online approach would be greedily adopting the best decision for the relaxed problem in each independent time slot. However, the instance switching cost C_{IS}^t temporally couples $y_j(t)$ across the time span, this naïve approach perhaps deviates greatly from the global optimum for the long-term cost.

To decouple the term of switching cost without drastic shifts in the solution between time slot t and $t-1$, and thus to obtain a good and provable competitive bound (through the primal-dual approach in Section 5.2), we exploit the algorithmic technique of regularization in online learning [16]. Regularization is a way to stabilize the solution and allows the online algorithm to avoid blindly following the best decision given the past data [34]. The basic idea of regularization is to solve the relaxed problem $\mathsf{P_R}$ with a smooth convex function to substitute the intractable $\big[y_j(t) - y_j(t-1)\big]^+$, we employ the widely adopted convex relative

entropy function [16] as follow:

$$\Delta(y_j(t)||y_j(t-1)) = y_j(t)\ln\frac{y_j(t)}{y_j(t-1)} + y_j(t-1) - y_j(t). \quad (9)$$

This smooth convex function is the sum of the relative entropy term $y_j(t)\ln\frac{y_j(t)}{y_j(t-1)}$ and a linear term denoting the movement cost $y_j(t-1) - y_j(t)$. To ensure that the fraction is still valid when no instance of DNN model j is deployed at time slot $t-1$ (i.e., $y_j(t-1) = 0$), we add a positive constant term ϵ to both $y_j(t)$ and $y_j(t-1)$ in (9). Moreover, we define an approximation weight factor $\eta_j = \ln(1 + \frac{E_j}{\epsilon})$ and multiply the improved relative entropy function with $\frac{1}{\eta_j}$ to normalize the switching cost by regularization.

Let $\mathsf{P_{Rr}}$ represent the relaxed and regularized problem by using the above enhanced regularizer $\Delta(y_j(t)||y_j(t-1))$ to approximate the time-coupling term $[y_j(t) - y_j(t-1)]^+$ in (5). Though $\mathsf{P_{Rr}}$ is still time-coupling, the *Karush-Kuhn-Tucker* (KKT) optimality conditions [35] of the regularized problem in each time slot yield a lower bound on the performance of the online algorithm for solving a series of single-shot problems. So we temporally decouple $\mathsf{P_{Rr}}$ into a series of single-shot convex programs ${\mathsf{P_{Rr}}}^t$, which can be solved in each individual time slot t based the solution obtained from the previous time slot $t-1$. Specifically, the decomposed subproblem ${\mathsf{P_{Rr}}}^t$ for each time slot $t \in \mathcal{T}$ can be denoted as follow:

$$\begin{aligned}
{\mathsf{P_{Rr}}}^t: \quad \min \quad & \omega \cdot Acc(t) + C_{EO}(t) + C_{CO}(t) + \\
& \sum_{j\in\mathcal{J}} \frac{s_j}{\eta_j}\left(\big(y_j(t)+\epsilon\big)\ln\frac{y_j(t)+\epsilon}{y_j(t-1)+\epsilon} - y_j(t)\right), \\
\text{s.t.} \quad & \text{Constraint (8a) to (8d)}, \\
& y_j(t) \in [0, E_j],\ \forall j \in \mathcal{J}.
\end{aligned}$$

We design FAAS - *Fractional Algorithm for joint configuration Adaption and model Selection* as shown in Algorithm 1. Our Algorithm 1 FAAS solves a standard convex problem ${\mathsf{P_{Rr}}}^t$ in each slot and outputs the fractional solution $\widetilde{\mathbf{x}}(t), \widetilde{\mathbf{y}}(t)$. Therefore, the time complexity analysis of Algorithm 1 is equivalent to the time complexity of solving the convex problem ${\mathsf{P_{Rr}}}^t$. Since the problem ${\mathsf{P_{Rr}}}^t$ is a standard convex optimization with linear constraints, it can be optimally solved in polynomial time by taking existing convex optimization techniques such as interior-point method [36]. And the complexity analysis of interior-point method is given in Section 11.5 of reference [35].

In our online scenario, future information includes the fluctuating operational cost $b_j(t)$ caused by real-time electricity price and inference request arrivals $A_i(t)$. At the same time, the online algorithm also utilizes the solution of the previous time slot. Therefore, FAAS first observes $\boldsymbol{A}(t), \boldsymbol{b}(t)$ and $\widetilde{\mathbf{y}}(t-1)$ at each time slot $t \in \mathcal{T}$, where $\widetilde{\mathbf{y}}(t-1)$ has been obtained when solving ${\mathsf{P_{Rr}}}^{t-1}$ at time slot $t-1$. Then, FAAS computes the optimal fractional solution $\big(\widetilde{\mathbf{x}}(t), \widetilde{\mathbf{y}}(t)\big)$ for the current time slot t. Obviously, the optimal solution of the relaxed and regularized problem ${\mathsf{P_{Rr}}}^t$ is a feasible solution of the relaxed problem $\mathsf{P_R}$. Later, we will derive the competitive ratio of FAAS in Section 5.2.

Algorithm 1. Fractional Algorithm for Joint Configuration Adaption and Model Selection — FAAS

Input: $\mathcal{I}, \mathcal{J}, \mathcal{K}, \boldsymbol{E}, \boldsymbol{L}, \boldsymbol{a}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, \boldsymbol{f}, \boldsymbol{s}, \boldsymbol{\eta}, \epsilon, \omega$
Output: $\widetilde{\mathbf{x}}(t), \widetilde{\mathbf{y}}(t)$
1: Initialization: $\widetilde{\mathbf{x}}(0) = \mathbf{0}, \widetilde{\mathbf{y}}(0) = \mathbf{0}$;
2: **for** each time slot $t \in \mathcal{T}$ **do**
3: Observe $\boldsymbol{A}(t), \boldsymbol{b}(t)$ and $\widetilde{\mathbf{y}}(t-1)$;
4: Invoke the *interior-point method* to solve the regularized problem ${\mathsf{P_{Rr}}}^t$;
5: **return**the optimal fractional solution $\widetilde{\mathbf{x}}(t), \widetilde{\mathbf{y}}(t)$;
6: **end for**

4.2 Rounding Algorithm for Resource Provisioning

The Algorithm 1 obtains a fractional solution $\big(\widetilde{\mathbf{x}}(t), \widetilde{\mathbf{y}}(t)\big)$ of problem ${\mathsf{P_{Rr}}}^t$, where the integer constraint (8e) is relaxed to $y_j(t) \in [0, E_j], \forall t \in \mathcal{T}, \forall j \in \mathcal{J}$. In order to satisfy the practical physical meaning of the variable $y_j(t)$, i.e., the number of model instances launched for DNN model j, we now design a rounding algorithm that rounds the optimal fractional solution $\widetilde{\mathbf{y}}(t)$ to an integer solution $\overline{\mathbf{y}}(t)$. A straightforward solution is the independent randomized rounding scheme [37], whose basic idea is to round up each fractional $\widetilde{\mathbf{y}}(t)$ to the nearest integer $\overline{\mathbf{y}}(t) = \lceil\widetilde{\mathbf{y}}(t)\rceil$ with the probability of $\widetilde{\mathbf{y}}(t) - \lfloor\widetilde{\mathbf{y}}(t)\rfloor$, i.e., $Pr\{\overline{\mathbf{y}}(t) = \lceil\widetilde{\mathbf{y}}(t)\rceil\} = \widetilde{\mathbf{y}}(t) - \lfloor\widetilde{\mathbf{y}}(t)\rfloor$, and round down $\widetilde{\mathbf{y}}(t)$ to the nearest integer $\overline{\mathbf{y}}(t) = \lfloor\widetilde{\mathbf{y}}(t)\rfloor$ with the probability of $\lceil\widetilde{\mathbf{y}}(t)\rceil - \widetilde{\mathbf{y}}(t)$, i.e., $Pr\{\overline{\mathbf{y}}(t) = \lfloor\widetilde{\mathbf{y}}(t)\rfloor\} = \lceil\widetilde{\mathbf{y}}(t)\rceil - \widetilde{\mathbf{y}}(t)$.

Although the above independent rounding scheme can always generate a feasible integer solution, since the central cloud can cover all the unserved requests incurred by rounding down $\widetilde{y}_j(t)$, directly applying this scheme may incur high edge operational cost (round up all fractional $\widetilde{y}_j(t)$) or cloud outsourcing cost (round down all fractional $\widetilde{y}_j(t)$). To solve the above challenge, we develop a randomized and dependent pairwise rounding scheme [38] which can exploit the inherent dependence of the variables $\widetilde{y}_j(t)$. The basic idea is that, each rounded-down variable will be compensated by another rounded-up variable, ensuring that all requests received by the edge server could be fully processed even after the rounding phase. Such a dependent pairwise rounding scheme can more reasonably round variables to satisfy the edge server capacity constraint, reducing the cost of the expensive cloud outsourcing or launching an excessive amount of DNN instances at the edge.

Based on the above analysis, we design RARP - *Rounding Algorithm for Resource Provisioning* which rounds the fractional solution and is shown in Algorithm 2, it runs in each time slot and returns the integer result of the fractional solution $\widetilde{\mathbf{y}}(t)$ obtained by FAAS. First, we introduce two index sets $\mathcal{J}_t^+ = \big\{j \,|\, \widetilde{y}_j(t) \in \mathbb{Z}\big\}$ and $\mathcal{J}_t^- = \big\{j \,|\, \widetilde{y}_j(t) \in \mathbb{R}^+\big\}$ to denote the set of DNN models with integral $\widetilde{y}_j(t)$ and the set of DNN models with fractional $\widetilde{y}_j(t)$ respectively for each time slot, so that we have $\mathcal{J}_t^+ \bigcup \mathcal{J}_t^- = \mathcal{J}$ in each time slot. For each element $j \in \mathcal{J}_t^-$, we further introduce a probability coefficient p_j and a weight coefficient ω_j associated with it. Here we define $p_j = \widetilde{y}_j(t) - \lfloor\widetilde{y}_j(t)\rfloor$ and $\omega_j = e_j, \forall j \in \mathcal{J}_t^-$. Next, the algorithm runs a series of rounding iterations. At each iteration, it randomly selects two elements j_1 and j_2 from $\mathcal{J}_t^-$, and let the probability of these two elements

round to 0 or 1, which is decided by the coupled coefficient φ_1 and φ_2. Finally, if $\mathcal{J}_t^-$ has only one element in the last iteration, we directly round it up to integer.

Algorithm 2. Rounding Algorithm for Resource Provisioning — RARP

Input: $\mathcal{J}, e, \widetilde{\mathbf{y}}(t)$
Output: $\overline{\mathbf{y}}(t)$
1: Let $\mathcal{J}_t^+ = \{j \,|\, \widetilde{y}_j(t) \in \mathbb{Z}\}, \mathcal{J}_t^- = \{j \,|\, \widetilde{y}_j(t) \in \mathbb{R}^+\}$;
2: **for** each edge DNN model $j \in \mathcal{J}_t^+$ **do**
3: Set $\overline{y}_j(t) = \widetilde{y}_j(t)$;
4: **end for**
5: **for** each DNN model $j \in \mathcal{J}_t^-$ **do**
6: Let $p_j = \widetilde{y}_i(t) - \lfloor \widetilde{y}_j(t) \rfloor, \omega_j = e_j$;
7: **end for**
8: **while** $|\mathcal{J}_t^-| > 1$ **do**
9: Randomly select two elements j_1, j_2 from $\mathcal{J}_t^-$;
10: Define $\varphi_1 = \min\{1 - p_{j_1}, \frac{\omega_{j_2}}{\omega_{j_1}} p_{j_2}\}$, $\varphi_2 = \min\{p_{j_1}, \frac{\omega_{j_2}}{\omega_{j_1}}(1 - p_{j_2})\}$;
11: With the probability $\frac{\varphi_2}{\varphi_1 + \varphi_2}$ set
12: $p_{j_1} = p_{j_1} + \varphi_1, p_{j_2} = p_{j_2} - \frac{\omega_{j_1}}{\omega_{j_2}} \varphi_1$;
13: With the probability $\frac{\varphi_1}{\varphi_1 + \varphi_2}$ set
14: $p_{j_1} = p_{j_1} - \varphi_2, p_{j_2} = p_{j_2} + \frac{\omega_{j_1}}{\omega_{j_2}} \varphi_2$;
15: If $p_{j_1} \in \{0, 1\}$, then set $\overline{y}_{j_1}(t) = \lfloor \widetilde{y}_{j_1}(t) \rfloor + p_{j_1}$,
16: $\mathcal{J}_t^+ = \mathcal{J}_t^+ \cup \{j_1\}, \mathcal{J}_t^- = \mathcal{J}_t^- \setminus \{j_1\}$;
17: If $p_{j_2} \in \{0, 1\}$, then set $\overline{y}_{j_2}(t) = \lfloor \widetilde{y}_{j_2}(t) \rfloor + p_{j_2}$,
18: $\mathcal{J}_t^+ = \mathcal{J}_t^+ \cup \{j_2\}, \mathcal{J}_t^- = \mathcal{J}_t^- \setminus \{j_2\}$;
19: **end while**
20: **if** $|\mathcal{J}_t^-| = 1$ **then**
21: Set $\overline{y}_i(t) = \lceil \widetilde{y}_i(t) \rceil$ for the only element $j \in \mathcal{J}_t^-$;
22: **end if**

The proposed pairwise rounding scheme would not aggressively launch new DNN instances or outsource unserved requests to the central cloud, and this is achieved by maintaining three desirable properties in the main loop of each iteration.

1) *Continuous Reduction Property.* At least one of two selected variables $\widetilde{y}_{j_1}(t)$ and $\widetilde{y}_{j_2}(t)$ is rounded into integer. For example, if $\varphi_1 = 1 - p_{j_1}$ and $\varphi_2 = p_{j_1}$ (line 10), then $p_{j_1} = 1$ if line 12 is executed or $p_{j_1} = 0$ if line 14 is executed. In both two cases, $\widetilde{y}_{j_1}(t)$ will be rounded to an integer.

2) *Weight Conservation Property.* After the main loop of each iteration, the total weighted resource capacity of the selected two elements (i.e., DNN instances) remains unchanged, in other words, the sum of $\widetilde{y}_{j_1}(t)e_{j_1} + \widetilde{y}_{j_2}(t)e_{j_2}$ stays constant. For example, if line 11 is executed, we have $\big[\widetilde{y}_{j_1}(t) + \varphi_1\big]e_{j_1} + \big[\widetilde{y}_{j_2}(t) - \frac{e_{j_1}}{e_{j_2}}\varphi_1\big]e_{j_2} = \widetilde{y}_{j_1}(t)e_{j_1} + \widetilde{y}_{j_1}(t)e_{j_2}$. Similarly, if line 13 is executed, we also have $\big[\widetilde{y}_{j_1}(t) - \varphi_2\big]e_{j1} + \big[\widetilde{y}_{j_2}(t) + \frac{e_{j_1}}{e_{j_2}}\varphi_2\big]e_{j2} = \widetilde{y}_{j_1}(t)e_{j_1} + \widetilde{y}_{j_2}(t)e_{j_2}$. This property indicates that additional cloud outsourcing cost with capacity reduction would not be incurred after rounding the fractional solution.

3) *Marginal Distribution Property.* In each iteration of the main loop, the probability of rounding up or down each element $j \in \mathcal{J}_t^-$ is determined by the fractional part $\widetilde{y}_j(t) - \lfloor \widetilde{y}_j(t) \rfloor$ of the fractional solution $\widetilde{y}_j(t)$. More specifically, $Pr\{\overline{y}_j(t) = \lceil \widetilde{y}_j(t) \rceil\} = \widetilde{y}_j(t) - \lfloor \widetilde{y}_j(t) \rfloor$ and $Pr\{\overline{y}_j(t) = \lfloor \widetilde{y}_j(t) \rfloor\} = 1 - \big(\widetilde{y}_j(t) - \lfloor \widetilde{y}_j(t) \rfloor\big)$. Based on this marginal distribution property which has been proven in [38], we can further derive:

$$\begin{aligned}\mathbb{E}\{\overline{y}_j(t)\} &= \big(\widetilde{y}_j(t) - \lfloor \widetilde{y}_j(t) \rfloor\big)\lceil \widetilde{y}_j(t) \rceil \\ &\quad + \big[1 - (\widetilde{y}_j(t) - \lfloor \widetilde{y}_j(t) \rfloor)\big]\lfloor \widetilde{y}_j(t) \rfloor \\ &= \big(\widetilde{y}_j(t) - \lfloor \widetilde{y}_j(t) \rfloor\big)(\lfloor \widetilde{y}_j(t) \rfloor + 1) \\ &\quad + \big[1 - (\widetilde{y}_j(t) - \lfloor \widetilde{y}_j(t) \rfloor)\big]\lfloor \widetilde{y}_j(t) \rfloor = \widetilde{y}_j(t).\end{aligned}$$

The above equation shows that the expectation of each rounded solution is exactly the value of the original fractional solution. This property indicates that additional costs with launching new DNN instances would not be incurred after rounding the fractional solution.

Our Algorithm 2 RARP rounds the fractional solution $\widetilde{\mathbf{y}}(t)$ obtained by Algorithm 1 to the integer in each slot through the iterations of the main loop. The main loop of RARP in each time slot needs to be executed at most $|\mathcal{J}_t^-|$ times, and $|\mathcal{J}_t^-| \leq J$. Thus, the time-complexity of Algorithm 2 is $O(J)$, which is independent of the scale of the requests arrival. Later, we will discuss the optimality gap between the rounded solution and the optimal fractional solution in Section 5.3.

4.3 Fractional Solution Relocation

After performing the Algorithm 2 at each time slot t, the resource provision decision $\overline{\mathbf{y}}(t)$ produced by RARP together with the model selection decision $\widetilde{\mathbf{x}}(t)$ may well not be a feasible solution of the original problem P. This means that we further need to modify the fractional model selection decision $\widetilde{\mathbf{x}}(t)$ to maintain the solution feasibility.

In order to obtain the model selection decision after rounding fractional solution $\widetilde{\mathbf{y}}(t)$, a naive approach is to bring the feasible rounded solution $\overline{\mathbf{y}}(t)$ back to the original problem P, and solve the degraded relocation problem to obtain a complete feasible solution $\big(\overline{\mathbf{x}}(t), \overline{\mathbf{y}}(t)\big)$. After bringing the rounded solution $\overline{\mathbf{y}}(t)$, the edge operational cost and instance switching cost in the objective function of the original problem become constant. However, if we further consider the weight conservation property maintained by the rounding scheme in RARP, we could obtain a complete feasible solution more simply. Specifically, the weight conservation property ensures that the total resource capacity of the rounded solution $\sum_j e_j \overline{y}_j(t)$ is equal to that of the fractional solution $\sum_j e_j \widetilde{y}_j(t)$. Therefore, we can solve the following simplified relocation problem without considering the cloud outsourcing cost to obtain $\overline{\mathbf{x}}(t)$.

$$\mathsf{P_{Re}}^t: \quad \min \quad Acc(t)$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} \overline{x}_{ijk}(t) \leq A_i(t), \qquad \forall i, \tag{10a}$$

$$\sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{K}} \overline{x}_{ijk}(t) f_{jk} \leq \overline{y}_j(t) e_j, \qquad \forall j, \tag{10b}$$

$$\frac{\sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} \overline{x}_{ijk}(t) d_{jk}}{\sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} \overline{x}_{ijk}(t)} \leq L_i, \qquad \forall i, \tag{10c}$$

$$\overline{x}_{ijk}(t) \geq 0, \qquad \forall i, \forall j, \forall k. \tag{10d}$$

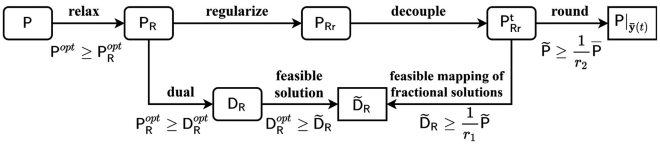

Fig. 3. An illustration of the basic idea of the performance analysis.

Since the above problem is a linear programming, therefore its optimal solution can be readily obtained in polynomial time. Our relocation scheme can work as a bridge to analyze the performance bound of the aforementioned naïve approach, as we will discuss later in Section 5. The rounded solution $\overline{\mathbf{y}}(t)$ obtained by RARP and the relocation solution obtained by solving the above problem together constitute the final solution of our online approach. In the next section, we will discuss the competitive ratio between the proposed online approach and the theoretical optimal solution in offline case.

5 Performance Analysis

In this section, we rigorously analyze the theoretical performance of the proposed online framework of EdgeAdaptor via competitive analysis. As a standard approach to quantifying the performance of online algorithms [39], the basic idea of the competitive analysis is to compare the performance of the online algorithm to the theoretical optimal solution in the offline case, where all the future information is given as prior knowledge. In particular, we prove that the proposed online algorithm has guaranteed performance, which is quantified by a worst-case parameterized competitive ratio.

5.1 The Basic Idea

As illustrated in Fig. 3, we introduce the dual problem of the relaxed problem D_R to act as the bridge that connects the original problem and regularized problem. We formally prove that the objective value of the original problem evaluated with solutions produced by our proposed online approach, which is denoted as $\overline{P}$, is upper-bounded by a parameterized constant multiple the offline optimum of the original problem. We will establish the following chain of inequalities:

$$\begin{aligned}
&\overline{\mathsf{P}}\big(\overline{\mathbf{x}}(t), \overline{\mathbf{y}}(t)\big) && \text{(11a)}\\
&\leq r_2 \widetilde{\mathsf{P}}\big(\widetilde{\mathbf{x}}(t), \widetilde{\mathbf{y}}(t)\big) && \text{(11b)}\\
&\leq r_1 r_2 \widetilde{\mathsf{D}}_\mathsf{R}\big(\nu_j(t), \alpha_i(t), \beta_j(t), \gamma_i(t), \lambda_j(t)\big) && \text{(11c)}\\
&\leq r_1 r_2 \mathsf{P}_\mathsf{R}^{opt} && \text{(11d)}\\
&\leq r_1 r_2 \mathsf{P}^{opt}, && \text{(11e)}
\end{aligned}$$

where $r = r_1 r_2$ is the overall *competitive ratio*. Here P^{opt} and $\mathsf{P}_\mathsf{R}^{opt}$ denote the objective value achieved by the optimal solution of the original problem P and the relaxed problem P_R, respectively. And $\widetilde{\mathsf{D}}_\mathsf{R}$ denotes the objective value of P_R's dual problem D_R, achieved by a feasible solution mapped from the optimal fractional solution and dual solution of the single-shot regularized problem ${\mathsf{P}_{\mathsf{Rr}}}^t$. Finally, $\widetilde{\mathsf{P}}$ and $\overline{\mathsf{P}}$ denote the objective value of the original problem P achieved by the optimal fractional solution and the rounded solution after relocation, respectively.

We first prove (11d) $\leq$ (11e) in Theorem 1. Next, the proof of (11b)$\leq$(11c)$\leq$(11d) is based on the online primal-dual techniques used in competitive analysis [40]. Specifically, the inequality $\mathsf{P}_\mathsf{R}^{opt} \geq \mathsf{D}_\mathsf{R}^{opt}$ holds due to the celebrated *Weak Duality* in convex optimization theory [35], where $\mathsf{D}_\mathsf{R}^{opt}$ denotes the objective value of the dual problem achieved by the optimal solution. To connect the dual problem D_R and the regularized problem P_{Rr}, we construct a feasible solution for D_R mapped from the optimal fractional solution for ${\mathsf{P}_{\mathsf{Rr}}}^t$. Based on such mapping, we can derive the competitive ratio r_1 by applying *Karush-Kuhn-Tucker* (KKT) conditions [35] in Section 5.2. Finally, based on the three desirable properties maintained by the rounding scheme and the constraints in relocation problem P_{Re}, we can further prove (11a) $\leq$ (11b) and characterize the competitive ratio r_2 between the optimal fractional solution and the rounded solution after relocation in Section 5.3.

Theorem 1. *$\mathsf{P}_\mathsf{R}^{opt} \leq \mathsf{P}^{opt}$ since P_R is a relaxed problem after relaxing the integer constraint in the original minimization problem P.*

Proof. See Appendix B, available in the online supplemental material. □

5.2 Competitive Ratio of FAAS

We establish (11b) $\leq$ (11c) and derive the competitive ratio r_1 of FAAS in this subsection. Specifically, we first transform the relaxes problem equivalently to drive its Lagrange problem. Then, we characterize the optimality conditions for a series of regularized problems ${\mathsf{P}_{\mathsf{Rr}}}^t$. Finally, we construct a feasible solution for D_R mapped from the optimal primal and dual solutions for ${\mathsf{P}_{\mathsf{Rr}}}^t$.

An Equivalent Problem Transformation. Due to the time-coupling switching cost C_{IS}^t and the boxing constraints of decision variables $y_j(t)$, directly deriving the Lagrange dual problem D_R for the relaxed problem P_R is not straightforward. In response, we first introduce a set of auxiliary variables $w_j(t)$ which satisfy $w_j(t) \geq y_j(t) - y_j(t-1),\ \forall j \in \mathcal{J}, \forall t \in \mathcal{T}$ to replace the time-coupling term $\big[y_j(t) - y_j(t-1)\big]^+$. The transformed switching cost $C_{IS}^t = \sum_j s_j \big[y_j(t) - y_j(t-1)\big]^+$ is equivalent to the original expression since the additional constraints $w_j(t) \geq 0, \forall j \in \mathcal{J}, \forall t \in \mathcal{T}$. For the boxing constraints $y_j(t) \in [0, E_j]$, as suggested by the literature [16], we replace it by a set of *knapsack cover* (KC) constraints $\sum_{j' \in \mathcal{J}} y_{j'}(t) e_{j'} - y_j(t) e_j \geq \sum_i \sum_j \sum_k x_{ijk}(t) f_{jk} - E_j e_j, \forall j \in \mathcal{J}, \forall t \in \mathcal{T}$. With the above transformations, we rewrite the relaxed problem in the following equivalent form.

$$\begin{aligned}
\min\quad & \sum_{t\in\mathcal{T}} \Big\{ \omega \cdot Acc(t) + C_{EO}(t) + C_{CO}(t) + \sum_j s_j w_j(t) \Big\} && \\
\text{s.t.}\quad & w_j(t) \geq y_j(t) - y_j(t-1),\ \forall t, j, && \text{(12a)}\\
& \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} x_{ijk}(t) - A_i(t) \leq 0,\ \forall t, i, && \text{(12b)}\\
& \sum_{i\in\mathcal{I}} \sum_{k\in\mathcal{K}} x_{ijk}(t) f_{jk} \leq y_j(t) e_j,\ \forall t, j, && \text{(12c)}\\
& \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} x_{ijk}(t) d_{jk} \leq \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} x_{ijk}(t) L_i, \forall t, i, && \text{(12d)}\\
& \sum\nolimits_i \sum\nolimits_j \sum\nolimits_k x_{ijk}(t) f_{jk} + y_j(t) e_j && \\
& - \sum\nolimits_{j'\in\mathcal{J}} y_{j'}(t) e_{j'} - E_j e_j \leq 0,\ \forall t, j, && \text{(12e)}\\
& x_{ijk}(t), y_j(t), w_j(t) \geq 0,\ \forall t \in \mathcal{T}, j \in \mathcal{J}. && \text{(12f)}
\end{aligned}$$

Deriving the Lagrange Dual Problem. We use $\nu_j(t)$, $\alpha_i(t)$, $\beta_j(t)$, $\gamma_i(t)$, $\lambda_j(t)$ to denote the corresponding dual variables for the constraints (12a) to (12e). Now we can derive the Lagrange dual problem $\mathsf{D_R}$ of the above transformed problem (and also equally the problem $\mathsf{P_R}$) as follows.

$$\mathsf{D_R}: \max \quad \sum_{t\in\mathcal{T}}\Big\{\sum_{i\in\mathcal{I}} A_i(t)\big(c_i-\alpha_i(t)\big)-\sum_{j\in\mathcal{J}}\lambda_j(t)E_je_j\Big\}$$

$$\text{s.t.} \quad s_j-\nu_j(t)\geq 0,\ \forall t\in\mathcal{T}, j\in\mathcal{J}, \tag{13a}$$

$$a_{ijk}-c_i+\alpha_i(t)+\beta_j(t)f_{jk} + \gamma_i(t)(d_{jk}-L_i)+\lambda_j(t)f_{jk}\geq 0,\forall t,i,j,k, \tag{13b}$$

$$b_j(t)+\nu_j(t)-\nu_j(t+1)-\beta_j(t)e_j - e_j\Big[\textstyle\sum_{j'}\lambda_{j'}(t)-\lambda_j(t)\Big]\geq 0,\forall t,j,$$

$$\text{All the dual variables}\geq 0. \tag{13c}$$

Characterizing the Optimality of the Regularized Problem. Recall that in Section 4.1, we regularize the relaxed problem $\mathsf{P_R}$ to a solvable convex problem $\mathsf{P_{Rr}}$, so that the optimal fractional solution $\big(\widetilde{\mathbf{x}}(t),\widetilde{\mathbf{y}}(t)\big)$ obtained by Algorithm 1 FAAS satisfies the *Karush-Kuhn-Tucker* (KKT) conditions, i.e., the first-order necessary conditions for optimality. Here we use $\big(\widetilde{\alpha}_i(t),\widetilde{\beta}_j(t),\widetilde{\gamma}_i(t),\widetilde{\lambda}_j(t)\big)$ to denote the optimal solution to the dual problem of the regularized problem ${\mathsf{P_{Rr}}}^t$, corresponding to constraints (12b) to (12e) respectively. To simplify the presentation, we write some KKT conditions in the disjunctive form, where $a\perp b$ is equivalent to $a,b\geq 0$ and $ab=0$. Therefore, the KKT conditions of the optimal fractional solution can be readily obtained as follows.

$$\widetilde{\alpha}_i(t)\perp\Big(A_i(t)-\sum\nolimits_j\sum\nolimits_k\widetilde{x}_{ijk}(t)\Big),\forall t,i \tag{14a}$$

$$\widetilde{\beta}_j(t)\perp\Big(\widetilde{y}_j(t)e_j-\sum\nolimits_i\sum\nolimits_k\widetilde{x}_{ijk}(t)f_{jk}\Big),\forall t,j \tag{14b}$$

$$\widetilde{\gamma}_i(t)\perp\Big(\sum\nolimits_j\sum\nolimits_k(L_i-d_{jk})\widetilde{x}_{ijk}(t)\Big),\forall t,i \tag{14c}$$

$$\widetilde{\lambda}_j(t)\perp\Big(\sum\nolimits_{j'}\widetilde{y}_{j'}(t)e_{j'}+E_je_j-\widetilde{y}_j(t)e_j - \sum\nolimits_i\sum\nolimits_j\sum\nolimits_k\widetilde{x}_{ijk}(t)f_{jk}\Big),\forall t,j \tag{14d}$$

$$a_{ijk}-c_i+\widetilde{\alpha}_i(t)+\widetilde{\beta}_j(t)f_{jk}+\widetilde{\gamma}_j(t)(d_{jk}-L_i) +\widetilde{\lambda}_j(t)=0,\forall t,i,j,k \tag{14e}$$

$$b_j(t)+\frac{s_j}{\eta_j}\ln\frac{\widetilde{y}_j(t)+\epsilon}{\widetilde{y}_j(t-1)+\epsilon}-\widetilde{\beta}_j(t)e_j - e_j\Big(\sum\nolimits_{j'}\widetilde{\lambda}_{j'}(t)-\widetilde{\lambda}_j(t)\Big)=0,\forall t,j \tag{14f}$$

Constructing the Mapping. After characterizing the optimality conditions for a series of regularized problems ${\mathsf{P_{Rr}}}^t$, we now construct a mapping to jointly map ${\mathsf{P_{Rr}}}^t$'s optimal primal and dual solutions to a feasible solution of the relaxed dual problem $\mathsf{D_R}$ as follows:

$$\mathbf{\Pi}\big\{\widetilde{\mathbf{x}}(t),\widetilde{\mathbf{y}}(t),\widetilde{\alpha}_i(t),\widetilde{\beta}_j(t),\widetilde{\gamma}_i(t),\widetilde{\lambda}_j(t)\big\} = \big(\nu_j(t),\alpha_i(t),\beta_j(t),\gamma_i(t),\lambda_j(t)\big),$$

in which we let

$$\nu_j(t)=\frac{s_j}{\eta_j}\ln\frac{E_j+\epsilon}{\widetilde{y}_j(t-1)+\epsilon},\ \alpha_i(t)=\widetilde{\alpha}_i(t),$$
$$\beta_j(t)=\widetilde{\beta}_j(t),\quad \gamma_i(t)=\widetilde{\gamma}_i(t),\ \lambda_j(t)=\widetilde{\lambda}_j(t).$$

Lemma 1. *The mapping* $\mathbf{\Pi}\big\{\widetilde{\mathbf{x}}(t),\widetilde{\mathbf{y}}(t),\widetilde{\alpha}_i(t),\widetilde{\beta}_j(t),\widetilde{\gamma}_i(t),\widetilde{\lambda}_j(t)\big\}$ *obtains a feasible solution of the relaxed dual problem* $\mathsf{D_R}$.

Proof. See Appendix C, available in the online supplemental material. □

Due to this feasibility, we know that the objective value $\widetilde{\mathsf{D}}_\mathsf{R}$ of the relaxed dual problem $\mathsf{D_R}$ is no larger than the optimal objective value $\mathsf{D}_\mathsf{R}^{opt}$. Recall the inequality $\mathsf{P}_\mathsf{R}^{opt}\geq\mathsf{D}_\mathsf{R}^{opt}$ based on the *Weak Duality* in convex optimization theory, we can derive (11c) $\leq$ (11d). Meanwhile, through the KKT conditions, we can also derive the optimality gap between $\widetilde{\mathsf{D}}_\mathsf{R}$ and $\widetilde{\mathsf{P}}$. To further obtain the competitive ratio r_1 of the proposed Algorithm 1 FAAS, we first show that the total non-switching cost (i.e., the objective value excludes the switching cost) is upper bounded by the objective value $\widetilde{\mathsf{D}}_\mathsf{R}$ of the relaxed dual problem $\mathsf{D_R}$, as shown by the following Lemma 2.

Lemma 2. *The sum of edge operational cost, cloud outsourcing cost and cumulative accuracy loss achieved by Algorithm 1 FAAS is no larger than the objective value of the relaxed dual problem* $\mathsf{D_R}$ *achieved by the constructed feasible solution* $\mathbf{\Pi}$, *i.e.,*

$$\sum_{t\in\mathcal{T}}(Acc(t)+C_{EO}(t)+C_{CO}(t))\leq\widetilde{\mathsf{D}}_\mathsf{R}.$$

The instance switching cost is also bounded, as shown by the following Lemma 3.

Lemma 3. *The total instance switching cost* $\sum_t C_{IS}^t$ *of the fractional solution achieved by Algorithm 1 FAAS is no larger than* $\big[\ln\big(1+\frac{E_{max}}{\epsilon}\big)+\frac{E_{max}}{\delta}\big]$ *times of* $\widetilde{\mathsf{D}}_\mathsf{R}$, *where* $E_{max}=\max_j E_j$ *and* $\delta=\min_{t,j}\big\{\widetilde{y}_j(t),\widetilde{y}_j(t)\ >\ 0\big\}$

$$\sum_{t\in\mathcal{T}}C_{IS}^t\leq\left[\ln\big(1+\frac{E_{max}}{\epsilon}\big)+\frac{E_{max}}{\delta}\right]\widetilde{\mathsf{D}}_\mathsf{R}.$$

We give a detailed proof of Lemmas 2 and 3 in Appendix D and Appendix E respectively, available in the online supplemental material.

Theorem 2. *The objective value* $\widetilde{\mathsf{P}}$ *of problem* P *through the optimal fractional solution achieved by our proposed Algorithm 1 FAAS is no larger than* r_1 *times of the offline optimum* P^{opt}, *where* r_1 *is given by:*

$$r_1=\ln\left(1+\frac{E_{max}}{\epsilon}\right)+\frac{E_{max}}{\delta}+1.$$

Proof. Since the objective value of the original problem P is the sum of edge operational cost, cloud outsourcing, instance switching cost and cumulative accuracy loss. According to Lemmas 2 and 3, we can directly derive Theorem 2. □

5.3 Integrality Gap of RARP and Relocation

In this subsection, we establish (11a) $\leq$ (11b) and study the rounding gap incurred by Algorithm 2 RARP and relocation, in terms of competitive ratio r_2 of the objective value $\overline{\mathsf{P}}$ achieved by the final rounded solution to the objective value $\widetilde{\mathsf{P}}$ achieved by the fractional solution. Recall the relationship between $\widetilde{\mathbf{y}}(t)$ and $\overline{\mathbf{y}}(t)$, which has been characterized by the weight conservation property in Section 4.2. Specifically, after the main loop of each iteration, the total resource capacity of the two selected elements $j_1, j_2 \in \mathcal{J}_t^-$ keeps unchanged, i.e., $\widetilde{y}_{j_1}(t)e_{j_1} + \widetilde{y}_{j_2}(t)e_{j_2} = \overline{y}_{j_1}(t)e_{j_1} + \overline{y}_{j_2}(t)e_{j_2}$. Based on this equation, we know that after the execution of RARP, for the integral elements $j \in \mathcal{J}_t^+$, we have the deterministic equation

$$\sum_{j\in\mathcal{J}_t^+} \overline{y}_j(t)e_j = \sum_{j\in\mathcal{J}_t^+} \widetilde{y}_j(t)e_j.$$

Then, we further take this connection as a bridge to bound the terms in the objective function of the original problem P. There is at most one element remaining in the fractional set $\mathcal{J}_t^-$ after the execution of RARP, i.e., $|\mathcal{J}_t^-| \leq 1$. Specifically, if $|\mathcal{J}_t^-| = 1$, for the only element in $\mathcal{J}_t^-$, by defining $\kappa_t = \frac{\max_{j\in\mathcal{J}} E_j e_j}{\sum_i \sum_j \sum_k \widetilde{x}_{ijk}(t)}$, we have

$$\begin{aligned}\sum_{j\in\mathcal{J}_t^-} \overline{y}_j(t)e_j &\leq \max_{j\in\mathcal{J}} E_j e_j \\ &= \kappa_t \sum_i \sum_j \sum_k \widetilde{x}_{ijk}(t) \leq \kappa_t \sum_j \widetilde{y}_j(t)e_j.\end{aligned}$$

Note that when there is no element in $\mathcal{J}_t^-$, i.e., $|\mathcal{J}_t^-| = 0$, the above inequality still holds since $\sum_{j\in\mathcal{J}_t^-} \overline{y}_j(t)e_j = 0$. Combining the above equation and inequality, we further have

$$\begin{aligned}\sum_{j\in\mathcal{J}} \overline{y}_j(t)e_j &= \sum_{j\in\mathcal{J}_t^+} \overline{y}_j(t)e_j + \sum_{j\in\mathcal{J}_t^-} \overline{y}_j(t)e_j \\ &\leq \sum_{j\in\mathcal{J}_t^+} \widetilde{y}_j(t)e_j + \kappa_t \sum_{j\in\mathcal{J}} \widetilde{y}_j(t)e_j \\ &\leq \sum_{j\in\mathcal{J}} \widetilde{y}_j(t)e_j + \kappa_t \sum_{j\in\mathcal{J}} \widetilde{y}_j(t)e_j \\ &= (1+\kappa_t) \sum_{j\in\mathcal{J}} \widetilde{y}_j(t)e_j.\end{aligned}$$

By summing over the above inequality over all the time slots $t \in \mathcal{T}$ and defining $\kappa = \max_{t\in\mathcal{T}} \kappa_t$, we finally have

$$\begin{aligned}\sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} \overline{y}_j(t)e_j &\leq \sum_{t\in\mathcal{T}} (1+\kappa_t) \sum_{j\in\mathcal{J}} \widetilde{y}_j(t)e_j \\ &\leq (1+\kappa) \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} \widetilde{y}_j(t)e_j. \end{aligned} \tag{15}$$

Upper Bound of the Edge Operational Cost. Based on the inequality (15), we bound the edge operational cost as follows.

$$\begin{aligned}&\sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} \overline{y}_j(t)b_j(t) = \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} \overline{y}_j(t)e_j \frac{b_j(t)}{e_j} \\ &\leq \max_{t,j} \frac{b_j(t)}{e_j} \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} \overline{y}_j(t)e_j \\ &\leq \max_{t,j} \frac{b_j(t)}{e_j} (1+\kappa) \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} \widetilde{y}_j(t)b_j(t) \frac{e_j}{b_j(t)} \\ &\leq (1+\kappa) \max_{t,j} \frac{b_j(t)}{e_j} \max_{t,j} \frac{e_j}{b_j(t)} \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} \widetilde{y}_j(t)b_j(t)\end{aligned} \tag{16}$$

Upper Bound of the Instance Switching Cost. Based on the inequality (15), we bound the instance switching cost as follows:

$$\begin{aligned}&\sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} s_j \left[\overline{y}_j(t) - \overline{y}_j(t-1)\right]^+ \\ &\leq \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} s_j \overline{y}_j(t) = \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} e_j \overline{y}_j(t) \frac{s_j}{e_j} \\ &\leq \max_j \frac{s_j}{e_j} \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} \overline{y}_j(t)e_j \\ &\leq (1+\kappa) \max_j \frac{s_j}{e_j} \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} \widetilde{y}_j(t)b_j(t) \frac{e_j}{b_j(t)} \\ &\leq (1+\kappa) \max_{t,j} \frac{e_j}{b_j(t)} \max_j \frac{s_j}{e_j} \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} \widetilde{y}_j(t)b_j(t).\end{aligned} \tag{17}$$

Upper Bound of the Cumulative Accuracy Loss. Based on the constraint (10b) in relocation problem Re^t and the inequality (15), the cumulative accuracy loss can be bounded as follows:

$$\begin{aligned}&\sum_t \sum_i \sum_j \sum_k a_{ijk} \overline{x}_{ijk}(t) \\ &\leq \max_{i,j,k} a_{ijk} \sum_t \sum_i \sum_j \sum_k \overline{x}_{ijk}(t) f_{jk} \frac{1}{f_{jk}} \\ &\leq \max_{i,j,k} a_{ijk} \min_{j,k} f_{jk} \sum_t \sum_j \overline{y}_j(t)e_j \\ &\leq (1+\kappa) \max_{i,j,k} a_{ijk} \min_{j,k} f_{jk} \sum_t \sum_j \widetilde{y}_j(t)b_j(t) \frac{e_j}{b_j(t)} \\ &\leq (1+\kappa) \max_{t,j} \frac{e_j}{b_j(t)} \max_{i,j,k} a_{ijk} \min_{j,k} f_{jk} \sum_t \sum_j \widetilde{y}_j(t)b_j(t).\end{aligned} \tag{18}$$

Theorem 3. *The objective value $\overline{\mathsf{P}}$ of problem P through the final rounded solution achieved by our Algorithm 2 RARP and relocation scheme is no larger than r_2 times of $\widetilde{\mathsf{P}}$ achieved by the optimal fractional solution, where $r_2 = \xi_1 + \xi_2 + \xi_3$, and*

$$\begin{aligned}\xi_1 &= (1+\kappa) \max_{t,j} \frac{b_j(t)}{e_j} \max_{t,j} \frac{e_j}{b_j(t)}, \\ \xi_2 &= (1+\kappa) \max_{t,j} \frac{e_j}{b_j(t)} \max_j \frac{s_j}{e_j}, \\ \xi_3 &= (1+\kappa) \max_{t,j} \frac{e_j}{b_j(t)} \max_{i,j,k} a_{ijk} \min_{j,k} f_{jk}.\end{aligned}$$

Proof. Recall the weight conservation property in Section 4.2, the total resource capacity of the rounded solution is no smaller than that of the optimal fractional solution. Therefore, when relocating the decision variables $\widetilde{\mathbf{x}}(t)$ after

rounding decision variables $\widetilde{\mathbf{y}}(t)$, we do not outsource additional requests to the central cloud, meaning that the cloud outsourcing cost of the final rounded solution is the same with that of the optimal fractional solution. Combining the upper bounds of several other terms in the objective function (16) to (18), we can derive the competitive ratio of the final rounded solution after the rounding scheme and relocation. □

5.4 The Overall Competitive Ratio

Theorem 4. *The objective value $\overline{\mathsf{P}}$ of problem P achieved by the final rounded solution is no larger than $r_1 r_2$ times of the offline optimum, where r_1 and r_2 are derived in Theorem 1 and Theorem 2, respectively. That is, our proposed online scheme for problem P achieves a competitive ratio of $r_1 r_2$.*

Proof. Based on the inequality (11a) to (11e) and the competitive ratio given in Theorem 2 and Theorem 3, we can directly derive the overall competitive ratio of our proposed online approach. □

The overall gap between our proposed online algorithms and the offline optimum mainly comes from two parts: *1)* The long-term optimization problem is solved online in each time slot without future information such as operational cost $b_j(t)$ and inference request arrivals $A_i(t)$, which is reflected in the competition ratio r_1 of Algorithm 1. *2)* Rounding the fractional solution to the integer solution for practical physical meaning incurs additional edge operational cost (round up $\widetilde{y}_j(t)$) or cloud outsourcing cost (round down $\widetilde{y}_j(t)$), which is reflected in the competition ratio r_2 of Algorithm 2. Intuitively, our online approach stabilizes the solution by avoiding drastic shifts in the solution between adjacent time slots, thus addressing the online scenario with the unknown future information. Meanwhile, the online algorithm obtains a good competitive bound as shown in Theorem 4.

For the overall competitive ratio given in Theorem 4, we now discuss some insights as follows: *1)* The final competitive ratio $r_1 r_2$ decreases with the increasing of the parameter ϵ. By increasing ϵ to be large enough, we can obtain the competitive ratio r_1 that is arbitrarily close to $\frac{E_{max}}{\delta} + 1$, but at the same time with the increasing of the time complexity of the regularized problem P_{Rr}. *2)* Compared to the naive approach that directly brings $\overline{\mathbf{y}}(t)$ back to the original problem P to obtain the complete feasible solution, our relocation scheme has lower time complexity but also reduced optimality. However, in a realistic edge-cloud system, the cloud outsourcing cost is far more expensive than edge-based processing.

6 Performance Evaluation

In this section, we evaluate the practical performance of the proposed online optimization framework through trace-driven simulations. The simulations are based on real-world request traces and inference accuracy and delay of object detection tasks.

6.1 Experimental Setup

We simulate an edge server with three DNN models ($J = 3$), corresponding to each model, the input images contain different configuration resolution of 240p, 360p, 540p, and 720p ($K = 4$). According to different types of object detection targets, we define five types of inference requests are people, car, bus, bike, and dog respectively ($I = 5$). For the convex problem $\mathsf{P}_{\mathsf{Rr}}{}^t$ at each time slot, we solve it with CVXPY [41] based on Python 3 by invoking the interior-point solver *MOSEK* [42].

TABLE 2
Accuracy Loss a_{ijk}

$\mathcal{J}$	$\mathcal{I}$ / $\mathcal{K}$	people	car	bus	bike	dog
YOLOv2	240p	0.287	0.329	0.300	0.281	0.216
	360p	0.238	0.283	0.252	0.231	0.163
	540p	0.195	0.242	0.210	0.188	0.116
	720p	0.187	0.235	0.202	0.180	0.107
SSD	240p	0.206	0.239	0.206	0.199	0.130
	360p	0.184	0.201	0.199	0.193	0.134
	540p	0.167	0.185	0.183	0.177	0.116
	720p	0.164	0.182	0.180	0.174	0.112
R-FCN	240p	0.242	0.272	0.267	0.251	0.167
	360p	0.197	0.229	0.224	0.207	0.118
	540p	0.156	0.189	0.184	0.166	0.073
	720p	0.149	0.183	0.178	0.160	0.066

Request Traces. Since edge intelligence is still in a very early stage, there is no publicly accessible inference request trace from the edge server. Moreover, for the protection of user data privacy, there is also no similar request trace with specific task type information. Therefore, in line with the recent work [27] on DNN model inference serving, we adopt workload trace of computing clusters to simulate the number of different types of inference requests arriving at the edge server. Specifically, we randomly select five different jobs from the 24-hour traces of Alibaba production cluster [43], and then expand these five kinds of workload traces in different proportions to a reasonable range according to the actual situation of each type of inference request in the real world.

Accuracy and Delay. To obtain the accuracy and delay parameters of the object detection task at the edge server, we select three currently popular object detection models YOLOv2 [13], SSD [14], R-FCN [15]. Specifically, for each DNN model, we calculate the parameters a_{ijk} based on the accuracy of different targets (i.e., people, car, etc.) and the parameters d_{jk} based on the FPS (frames per second) of images with different configurations (i.e., 240p, 360p, etc.) through the curve fitting method [23]. The service latency constraints L_i of different object detection tasks are increased from 30ms to 150ms according to actual demands. Specific parameters are listed in Tables 2 and 3.

Cost Parameters. We simulate the DNN instance running cost $b_j(t)$ through the parameter amount of each DNN model and the electricity price with temporal diversities. As we mentioned in Section 3.3, there is the numerical relationship between service latency and price [28]. On the basis of the price of the serverless scheme (i.e., AWS Lambda [44]), we add the penalty incurred by the latency of data transmission to the unit cloud outsourcing cost c_i, and the value of the penalty depends on the service latency constraints of different object detection tasks. Specifically, every 30ms corresponds to 20% of the base price. Inspired by the experimental setup in [45] and [46], we normalize the unit switching cost s_j by using the mean of the operational cost $b_j(t)$. Specifically, we

TABLE 3
Instance Latency d_{jk}

DNN Model	Configuration			
	240p	360p	540p	720p
YOLOv2	11.0 ms	12.3 ms	16.9 ms	25.0 ms
SSD	21.7 ms	24.6 ms	52.6 ms	80.4 ms
R-FCN	76.9 ms	77.7 ms	85.0 ms	92.7 ms

set $s_j = \mathbb{E}\{b_j(t)\}$, which corresponds to the operational cost to running one model instance in each time slot.

Capacity Constraints. The computing capability of different DNN model instances is difficult to actually measure, so we refer to the model size of three DNN models and design the workload capacity e_j of each DNN model instance based on this ratio in simulation experiment. The intuitive insight is that all containers with different DNN model have the same physical resources (e.g., CPU and memory). Therefore, the larger parameters amount of the loaded model in the container, the fewer requests processed per time slot. The degraded parameter e_j represents the maximum number of requests processed by DNN model j in one time slot. As for E_j, the maximal number of available instances of DNN model j, we set it according to the peak-arrival of request traces at the edge server. Specially, we summary the DNN capacity configurations in Table 4.

Benchmarks. To demonstrate the efficacy of the proposed online framework (FAAS+RARP), we compare it to the *Online Lazy Switching Algorithm* (OLSA) [47]. The basic idea of OLSA is to tolerate as much non-switching cost (i.e., the sum of edge operational cost, cloud outsourcing cost and cumulative accuracy loss) as possible until the non-switching cost significantly exceeds the switching cost. OLSA is a state-of-the-art online algorithm with competitive analysis [48], which has been extensively applied to tackle problems with switching cost in recent literature [49], [50], [51]. However, the characteristic of OLSA is lazy switching, which makes it unable to cope well with the frequently changing request arrivals in the dynamic environment. Furthermore, to demonstrate the efficacy of the rounding scheme RARP, we compare it to another two benchmarks: *1)* EO-Greedy approach directly rounds up all the fractional solutions to process all received inference requests at the edge server. *2)* CO-Greedy approach directly rounds down all the fractional solutions with the help of the central cloud to serve the inference requests which can not be covered by the edge server.

6.2 Evaluation Results

We use the competition ratio to measure the performance of the online algorithm, which is equal to the objective value achieved by the online algorithm divided by the offline optimal value. In addition, we also use the following more intuitive quantitative indicators to measure the performance of our online algorithms: i) *total cost*, i.e., objective value in each time slot, ii) *average total cost*, i.e., total objective value over the entire time horizon averaged by T.

TABLE 4
DNN Model Instance Configuration

DNN Model	Workload Capacity e_j	Resource Capacity E_j
YOLOv2	180	40
SSD	200	30
R-FCN	190	20

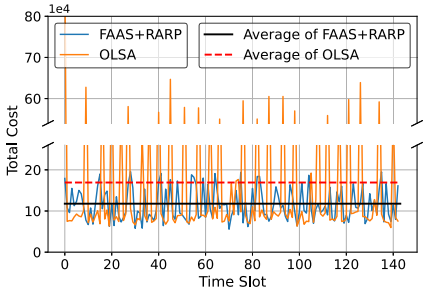

Fig. 4. Total cost versus T.

Efficiency of the Online Framework. We first plot the total cost of the proposed online framework (FAAS+RARP) as well as the benchmark OLSA in each time slot as shown in Fig. 4, and we mark the average total cost of these two methods in 144 time slots. To further show the gap between our online algorithm and the offline optimal solution, we also plot the competitive ratio of the two online approaches in Fig. 5. From these two figures, we observe that: *1)* The proposed online approach (FAAS+RARP) has lower long-term total cost than OLSA, demonstrating the effectiveness of our proposed EdgeAdaptor. *2)* The proposed online algorithm (FAAS+RARP) achieves a competition ratio of no more than 1.4 in different time horizons T, which is significantly lower than that of OLSA, and achieves a maximum reduction of 30.8%. *3)* As the number of total time slots varies, EdgeAdaptor has stable performance against uncertain influence caused by time-varying parameters (i.e., fluctuating workload and electricity price). The total cost of OLSA exhibits violent fluctuations in some time slots due to its lazy switching characteristic. The "laziness" of OLSA limits it to be unable to cope well with the frequently changing request arrivals in the dynamic environment compared to our proposed online approach.

Efficiency of the Rounding Scheme. For a more intuitive comparison of rounding scheme, we plot the average total cost of three different rounding approaches under varying numbers of total time slots in Fig. 6. It demonstrates that the average total cost of our proposed RARP outperforms those of the two benchmarks. However, the performance of RARP is not obvious compared with EO-Greedy. This is mainly because the number of DNN models is only 3 in our simulation setup, even if all the fractional solutions are rounded up, the

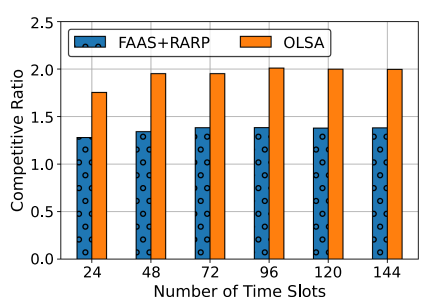

Fig. 5. Competitive ratio versus T.

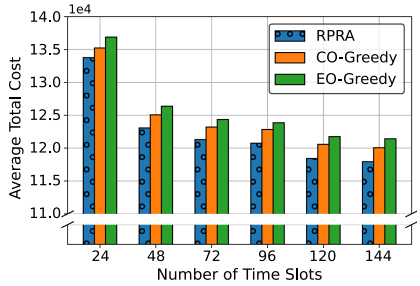

Fig. 6. Average total cost versus T in rounding.

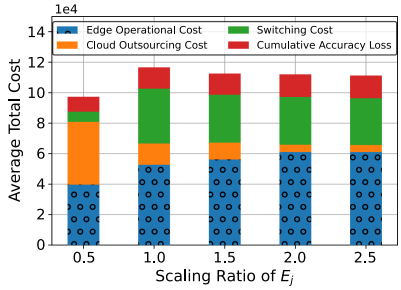

Fig. 8. Various cost versus resource capacity E_j.

additional instance running cost would not be too large. Actually, this gap will inevitably be further enlarged as the scale of edge servers increases. We also note that the average total cost gradually decreases with the increase of the total number of time slots. The reason for this phenomenon is that the decision variables are initialized to 0 in our simulation setup, thus a huge switching cost is generated in the first time slot while algorithm running. In a reality system, we can preheat the system (e.g., implement some instances in advance) to avoid this terrible cold start phenomenon.

Effect of the Switching Cost. We next investigate the effect of the instance switching cost (i.e., s_j in our formulation) on the average total cost of our online approach (FAAS+RARP) and OLSA, by multiplying the switching cost of each DNN instance with a various scaling ratio. The results are plotted in Fig. 7, which shows that as the switching cost increases, the average total cost of both our online approach and OLSA increase dramatically. The rationale is that, by increasing the scaling ratio of switching cost, the algorithms would pay more attention to minimize the total switching cost. However, since the switching cost term is time-coupling and involves future stochastic information, the optimality gap incurred by any online algorithm would increase as the switching cost s_j grows. Nevertheless, we mark the ratio of the average total cost between two online approaches, which indicates that our proposed EdgeAdaptor can suppress these deteriorating increase compared to OLSA until the scaling ratio extends to 10.

Effect of the Edge Capacity. Recall that in Section 3.1, we use E_j to denote the resource capacity at the edge server, i.e., the maximal number of available instances of DNN models. Fig. 8 depicts the impact of E_j on the average total cost under different scaling ratios, especially the impact on edge operational cost, from which we observe that: *1)* Increasing resource capacity of edge server can effectively reduce the proportion of expensive cloud outsourcing cost in the total system cost. *2)* The average total cost of the system is the smallest, when the scaling ratio is 0.5, i.e., the edge resource capacity is further limited. This is because that almost all containers on the edge server are running to provide inference services, so fluctuating workload will not generate switching cost (as shown by the green column in Fig. 8). Despite the high cost of cloud outsourcing, the rapid expansion of serverless technology makes it possible to reduce cold starts. *3)* As the edge server resource capacity increases, the average total cost has a downward trend, however, this decline is quickly filled by increased cumulative accuracy loss. Because compared with cloud outsourcing which does not consider the inference accuracy loss, increasing inference requests served at the edge server will inevitably lead to an increase on the cumulative accuracy loss.

Cost, Latency, and Accuracy. Fig. 9 shows the three-way tradeoff between resource cost, service latency, and inference accuracy. As the scaling ratio of L_i increases (i.e., the tolerance for inference service latency increases), all three curves indicate that the total cost of the system decreases. And as the parameter ω increases (i.e., the accuracy requirement of inference service increases), the total cost of the system increases between three curves. Although the plotted curves are in line with our intuitive perception, there is also an abnormal data point when $\omega = 5$ and $0.5 \times L_i$. The reason may be that the algorithm tends to choose the cloud outsourcing policy under the fluctuating workload due to the dual effect of higher latency requirement and lower accuracy requirement, the expensive switching cost is avoided as a result. The convex relationship between the total cost and the inference service latency also brings inspiration for the deployment of realistic edge inference systems. In the range of 0.1 to 0.3, we can appropriately relax the latency requirement of inference service in exchange for a substantial reduction in system cost, while in the range of 0.5 to 3.0, we can choose a more stringent latency requirement without incurring much cost.

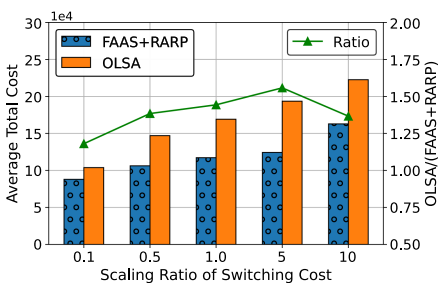

Fig. 7. Average total cost versus scaling ratio of s_j.

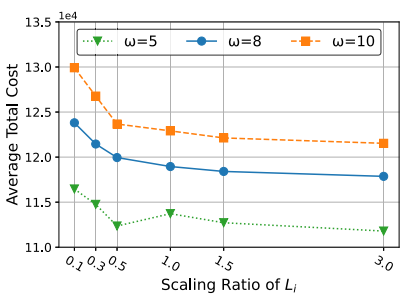

Fig. 9. Cost versus Latency versus Accuracy.

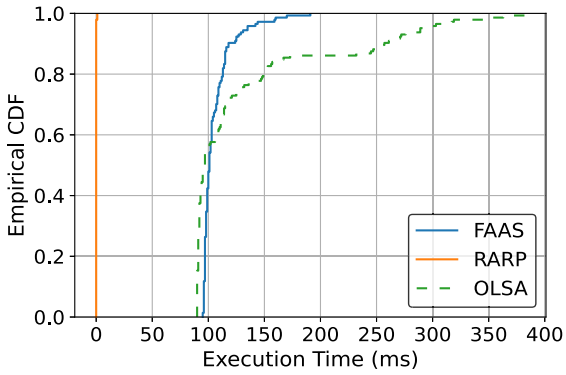

Fig. 10. Distribution of execution time of our proposed online algorithms.

Algorithm Execution Time. Fig. 10 shows the distribution of execution time of our proposed online algorithms and OLSA. We observe that the empirical cumulative distribution function (CDF) curve of RARP is almost a vertical line, because its execution time in each slot is less than 1ms. The execution time of FAAS varies from 90ms to 200ms, and its average execution time over all 144 time slots is about 106ms. Our FAAS takes relatively much longer time to run than the rounding algorithm, because it involves a regularized convex problem to be solved. Our RARP takes negligible time to finish, as it has a linear time complexity by design. Our online approach (FAAS+RARP) is extremely efficient and scalable because the overall running times are much less than the commonly adopted length of 15 minutes of a time slot (recall that our algorithm runs once at each time slot), which indicates the real-time responses of our online algorithms to large-scale requests.

7 Conclusion

In this work, we propose an online optimization framework EdgeAdaptor for edge DNN inference serving at scale. It jointly optimizes the application configuration adaption, DNN model selection and edge resource provisioning to judiciously navigate the three-way tradeoff between resource cost, inference accuracy and latency. Since the formulated optimization problem is NP-hard and involves future uncertain information, we carefully fuse the power of an online optimization technique and an approximate optimization method. Specifically, with a regularization technique, we first decompose the long-term problem into a series of one-shot fractional problems which can be readily solved. Then, applying a randomized dependent scheme, we round the fractional solutions to a near-optimal feasible solution of the original problem. Rigorous theoretical analysis and extensive trace-driven simulations demonstrate the efficacy of our proposed framework.

References

[1] J. Zhang and D. Tao, "Empowering things with intelligence: A survey of the progress, challenges, and opportunities in artificial intelligence of things," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 7789–7817, May 2021.

[2] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[6] B. Reagen *et al.*, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *Proc. IEEE/ACM 43rd Annu. Int. Symp. Comput. Archit.*, 2016, pp. 267–278.

[7] S. Liu, Y. Lin, Z. Zhou, K. Nan, H. Liu, and J. Du, "On-demand deep model compression for mobile devices: A usage-driven model selection framework," in *Proc. ACM Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2018, pp. 389–400.

[8] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.

[9] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "DeepDecision: A mobile deep learning framework for edge video analytics," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 1421–1429.

[10] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proc. ACM Workshop Mobile Edge Commun.*, 2018, pp. 31–36.

[11] J. Huang, C. Samplawski, D. Ganesan, B. Marlin, and H. Kwon, "CLIO: Enabling automatic compilation of deep learning pipelines across IoT and cloud," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, Art. no. 58.

[12] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "SPINN: Synergistic progressive inference of neural networks over device and cloud," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, Art. no. 37.

[13] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6517–6525.

[14] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.

[15] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," 2016, *arXiv:1605.06409*.

[16] N. Buchbinder, S. Chen, and J. Naor, "Competitive analysis via regularization," in *Proc. 25th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2014, pp. 436–444.

[17] Y. Kang *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proc. 22nd Int. Conf. Architect. Support Program. Lang. Operating Syst.*, 2017, pp. 615–629.

[18] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: Scalable adaptation of video analytics," in *Proc. Conf. ACM Special Int. Group Data Commun.*, 2018, pp. 253–266.

[19] P. Yang, F. Lyu, W. Wu, N. Zhang, L. Yu, and X. S. Shen, "Edge coordinated query configuration for low-latency and accurate video analytics," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4855–4864, Jul. 2020.

[20] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica, "Clipper: A low-latency online prediction serving system," in *Proc. 14th USENIX Symp. Netw. Syst. Des. Implementation*, 2017, pp. 613–627.

[21] F. Romero, Q. Li, N. J. Yadwadkar, and C. Kozyrakis, "INFaaS: Automated model-less inference serving," in *Proc. USENIX Annu. Tech. Conf.*, 2021, pp. 397–411.

[22] W. Wu, P. Yang, W. Zhang, C. Zhou, and X. Shen, "Accuracy-guaranteed collaborative DNN inference in industrial IoT via deep reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4988–4998, Jul. 2021.

[23] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 257–266.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[25] Z. Zhou, F. Liu, R. Zou, J. Liu, H. Xu, and H. Jin, "Carbon-aware online control of geo-distributed cloud services," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2506–2519, Sep. 2016.

[26] Z. Zhou, Q. Wu, and X. Chen, "Online orchestration of cross-edge service function chaining for cost-efficient edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1866–1880, Aug. 2019.

[27] C. Zhang, M. Yu, W. Wang, and F. Yan, "MArk: Exploiting cloud services for cost-effective, SLO-aware machine learning inference serving," in *Proc. USENIX Annu. Tech. Conf.*, 2019, pp. 1049–1062.
[28] A. Singla, B. Chandrasekaran, P. B. Godfrey, and B. Maggs, "The internet at the speed of light," in *Proc. 13th ACM Workshop Hot Topics Netw.*, 2014, pp. 1–7.
[29] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. H. Andrew, "Greening geographic load balancing," in *Proc. ACM SIGMETRICS Joint Int. Conf. Meas. Model. Comput. Syst.*, 2011, pp. 233–244.
[30] J. Dean and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, pp. 74–80, 2013.
[31] L. Suresh, M. Canini, S. Schmid, and A. Feldmann, "C3: Cutting tail latency in cloud data stores via adaptive replica selection," in *Proc. 12th USENIX Symp. Netw. Syst. Des. Implementation*, 2015, pp. 513–527.
[32] A. Ali-Eldin, B. Wang, and P. Shenoy, "The hidden cost of the edge: A performance comparison of edge and cloud latencies," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2021, pp. 1–12.
[33] J. Csirik, "Heuristics for the 0-1 min-knapsack problem," *Acta Cybernetica*, vol. 10, no. 1–2, pp. 15–20, 1991.
[34] A. Rakhlin, J. Abernethy, A. Agarwal, P. Bartlett, E. Hazan, and A. Tewari, "Lecture notes on online learning draft," 2009. [Online]. Available: http://www.mit.edurakhlin/papers/onlinelearning.pdf
[35] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[36] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. Philadelphia, PA, USA: SIAM, 1994.
[37] P. Raghavan and C. D. Tompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, 1987.
[38] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan, "Dependent rounding and its applications to approximation algorithms," *J. ACM*, vol. 53, no. 3, pp. 324–360, 2006.
[39] M. Lin, Z. Liu, A. Wierman, and L. L. H. Andrew, "Online algorithms for geographical load balancing," in *Proc. Int. Green Comput. Conf.*, 2012, pp. 1–10.
[40] N. Buchbinder *et al.*, "The design of competitive online algorithms via a primal–dual approach," *Found. Trends Theor. Comput. Sci.*, vol. 3, no. 2/3, pp. 93–263, 2009.
[41] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 83, pp. 1–5, 2016.
[42] Mosek Optimizer. Accessed: 2022. [Online]. Available: https://www.mosek.com/
[43] Alibaba Production Cluster Trace Data. Accessed: 2022. [Online]. Available: https://github.com/alibaba/clusterdata
[44] AWS Lambda Price. Accessed: 2022. [Online]. Available: https://aws.amazon.com/cn/lambda/pricing/
[45] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1378–1391, Oct. 2013.
[46] Y. Zeng, Y. Huang, Z. Liu, and Y. Yang, "Joint online edge caching and load balancing for mobile data offloading in 5G networks," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 923–933.
[47] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1459–1467.
[48] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. M. Lau, "Moving big data to the cloud: An online cost-minimizing approach," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2710–2721, Dec. 2013.
[49] B. Gao, Z. Zhou, F. Liu, F. Xu, and B. Li, "An online framework for joint network selection and service placement in mobile edge computing," *IEEE Trans. Mobile Comput.*, early access, Mar. 9, 2021, doi: 10.1109/TMC.2021.3064847.
[50] X. Qi, H. Xu, Z. Ma, and S. Chen, "Joint network selection and task offloading in mobile edge computing," in *Proc. IEEE/ACM 21st Int. Symp. Cluster Cloud Internet Comput.*, 2021, pp. 475–482.
[51] Q. Zhang, F. Liu, and C. Zeng, "Online adaptive interference-aware VNF deployment and migration for 5G network slice," *IEEE/ACM Trans. Netw.*, vol. 29, no. 5, pp. 2115–2128, Oct. 2021.

Kongyange Zhao received the BE degree from the South China University of Technology, China, in 2020. He is currently working toward the master's degree with Sun Yat-sen University, China. His research interests include edge computing, edge intelligence, and serverless computing.

Zhi Zhou (Member, IEEE) received the BS, ME, and PhD degrees from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, in 2012, 2014, and 2017, respectively. He is currently an Associate Professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. In 2016, he was a visiting scholar with the University of Göttingen. He was nominated for the 2019 China Computer Federation CCF Outstanding Doctoral Dissertation Award, the sole recipient of the 2018 ACM Wuhan & Hubei Computer Society Doctoral Dissertation Award, and a recipient of the Best Paper Award of IEEE UIC 2018. His research interests include edge computing, cloud computing, and distributed systems.

Xu Chen (Senior Member, IEEE) received the PhD degree in information engineering from the Chinese University of Hong Kong, in 2012. He is a full professor with Sun Yat-sen University, Guangzhou, China, and the vice director of National and Local Joint Engineering Laboratory of Digital Home Interactive Applications. He worked as a postdoctoral research associate with Arizona State University, Tempe, USA, from 2012 to 2014, and a Humboldt scholar fellow with the Institute of Computer Science, University of Goettingen, Germany from 2014 to 2016. He received the prestigious Humboldt research fellowship awarded by the Alexander von Humboldt Foundation of Germany, 2014 Hong Kong Young Scientist Runner-up Award, 2017 IEEE Communication Society Asia-Pacific Outstanding Young Researcher Award, 2017 IEEE ComSoc Young Professional Best Paper Award, Honorable Mention Award of 2010 IEEE international conference on Intelligence and Security Informatics (ISI), Best Paper Runner-up Award of 2014 IEEE International Conference on Computer Communications (INFOCOM), and Best Paper Award of 2017 IEEE Intranational Conference on Communications (ICC). He is currently an area editor of the *IEEE Open Journal of the Communications Society*, an associate editor of the *IEEE Transactions Wireless Communications*, *IEEE Internet of Things Journal* and *IEEE Journal on Selected Areas in Communications* (JSAC) Series on Network Softwarization and Enablers.

Ruiting Zhou (Member, IEEE) received the PhD degree from the Department of Computer Science, University of Calgary, Canada, in 2018. She has been an associate professor with the School of Cyber Science and Engineering, Wuhan University since June 2018. Her research interests include cloud computing, machine learning, and mobile network optimization. She has published research papers in top-tier computer science conferences and journals, including IEEE INFOCOM, ACM MobiHoc, ICDCS, the *IEEE/ACM Transactions on Networking*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Mobile Computing*. She also serves as a reviewer for journals and international conferences such as the *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Wireless Communications*, and IEEE/ACM IWQoS.

Xiaoxi Zhang (Member, IEEE) received the BE degree in electronics and information engineering from the Huazhong University of Science and Technology, in 2013, and the PhD degree in computer science from the University of Hong Kong, in 2017, advised by Prof. Chuan Wu and Prof. Francis C.M. Lau. She is an Associate Professor with the School of Computer Science and Engineering, Sun Yat-sen University. Before joining SYSU, she was a postdoctoral researcher with the Department of Electrical and Computer Engineering, Carnegie Mellon University, advised by Prof. Carlee Joe-Wong. Her research interests lie in the broad area of optimization and algorithm design for networked systems, including edge computing networks, and distributed machine learning systems.

Shuai Yu (Member, IEEE) received the BS degree from the Nanjing University of Post and Telecommunications, Nanjing, China, in 2009, the MS degree from the Beijing University of Post and Telecommunications, Beijing, China, in 2014, and the PhD degree from the University Pierre and Marie Curie (now Sorbonne Université), Paris, France, in 2018. He is currently an associate professor with Sun Yat-sen University, Guangzhou, China. His research interests include edge computing, mobile computing, machine learning, and space-air-ground integrated networks.

Di Wu (Senior Member, IEEE) received the BS degree from the University of Science and Technology of China, Hefei, China, in 2000, the MS degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2003, and the PhD degree in computer science and engineering from the Chinese University of Hong Kong, Hong Kong, in 2007. He was a postdoctoral researcher with the Department of Computer Science and Engineering, Polytechnic Institute of New York University, Brooklyn, NY, USA, from 2007 to 2009, advised by Prof. K. W. Ross. He is currently a professor and the associate dean with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. He was the recipient of the IEEE INFOCOM 2009 Best Paper Award, IEEE Jack Neubauer Memorial Award, and etc. His research interests include edge/cloud computing, multimedia communication, Internet measurement, and network security.