RESCUE: Opportunistic Online Scheduling of Model Retraining on Underutilized Edges

Jianping Huang[1,*], Xiang Liu[1,2,*], Feng Shan[1,†]
[1] Southeast University, China [2] The Chinese University of Hong Kong, Hong Kong
Email: {jphuang, xiangliu, shanfeng}@seu.edu.cn

***Abstract*—The proliferation of Artificial Intelligence (AI) applications on edge devices requires frequent edge-assisted model retraining to mitigate data drift. This creates significant resource contention on edge servers, exacerbated by pre-committed high-priority reserved tasks. Instead of investing in costly dedicated servers, we identify a key and overlooked opportunity lying in Underutilized Edge Computing (UEC) resources, which arise from the over-provisioning and fragmentation of these reserved tasks. However, existing scheduling paradigms, designed for dedicated resources, are ineffective at leveraging transient and fragmented UEC environment. To bridge this gap, we introduce RESCUE, a novel online framework that unlocks the UEC's value for opportunistic retraining. RESCUE enables real-time decision-making under uncertainties in task arrival and retraining duration. It facilitates heterogeneous resource-task assignments and co-schedules reserved tasks to maximize expected profit. We formulate this challenge as a stochastic joint-optimization problem and propose a two-stage approach: the offline stage establishes an optimal benchmark and value functions, while the online stage uses this guidance to make irrevocable scheduling decisions in an uncertain environment. We prove RESCUE achieves a competitive ratio of $1/2$. Extensive simulations consistently yield a high empirical competitive ratio between 0.51–0.87, and an average profit increase of over 60% compared to baseline methods. These results indicate RESCUE's robustness and scalability, establishing its effectiveness for real-world edge systems.**

I. Introduction

The widespread adoption of Artificial Intelligence (AI) on edge devices has enabled many applications, including augmented reality [1], autonomous driving [2], and intelligent video analytics [3]. However, these deployed models, often compressed with fewer parameters [4], are vulnerable to performance degradation due to shifts in real-world data distributions, a phenomenon known as *data drift* [5]. For instance, intelligent video analytics [6], [7], used in street cameras and smart cars, face diverse scenes over time, like variations in lighting, weather, and crowd densities. To maintain accuracy, edge-assisted model retraining offers a promising solution with low latency and enhanced data privacy [3], [8], as devices can send new data to nearby servers for retraining.

Yet, rising retraining demands create intense contention on limited edge resources, a situation worsened by pre-committed and high-priority *reserved tasks*, such as lightweight backend computations or periodic data processing [9]. Unlike prior work [6], [10], which relies on costly dedicated retraining servers that incur high hardware, maintenance, and centralized communication overheads [11], as well as contribute to carbon footprint [12], we identify a critical and overlooked opportunity: a considerable portion of existing edge server capacity remains chronically underutilized, referred to as *Underutilized Edge Computing* (UEC).

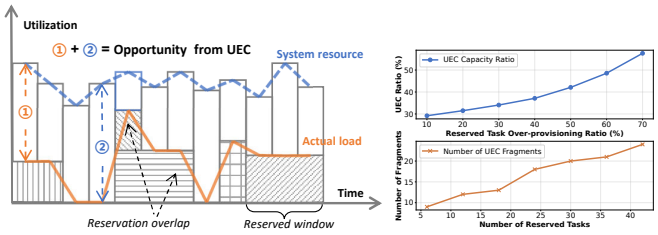

Fig. 1. Opportunity from UEC. *(Left Panel)* UEC is the dynamic gap between statically reserved system resources (blue dashed line) and their actual, often lower, loads (orange line). This gap stems from two primary sources: ① over-provisioning to ensure QoS for individual tasks, and ② fragmentation, where idle slots between reservations become too small to be individually useful. Collectively, they form a volatile yet valuable pool of resources. *(Right Panel)* Our empirical analysis confirms the scale and complexity of UEC. The top plot shows that UEC capacity can reach 30% of total resources with only 10% over-provisioning. The bottom plot reveals that this capacity becomes increasingly fragmented as the system hosts more tasks, complicating its effective utilization.

As depicted in Fig. 1, UEC resources mainly arise from two factors. First (①), to ensure quality of service (QoS) for reserved tasks, resources are over-provisioned to potential peak loads, particularly under fixed reservation policies [13]. This often exceeds routine resource demands, thereby causing UEC gaps. While auto-scaling [14], [15] and overbooking [16] mitigate this, they either react slowly to dynamic UEC or risk pre-committed reserved task performance [17]. Second (②), resource fragmentation creates UEC gaps from varying task demands and static scheduling, which is hard to aggregate for new tasks even if the total idle capacity is adequate. Such resource underutilization is a significant economic burden even in mature cloud ecosystems [18], [19], where waste rates can reach one-third [20]. Our empirical simulations (right part of Fig. 1) reveal that in edges, the UEC ratio can achieve about 30% even with a 10% over-provisioning ratio of reserved tasks; furthermore, resource fragmentation escalates rapidly with the rise in reserved tasks. Edges suffer even higher waste than clouds due to their inherently volatile and heterogeneous resources. Thus, repurposing UEC for model retraining can reduce waste, cut costs, and boost edge efficiency and sus-

* Equal contribution. † Corresponding author.

tainability, which highlights the need for efficient resource allocation in scalable edge intelligence.

However, current paradigms fail to effectively utilize UEC resources for model retraining. First, both general-purpose resource schedulers [14], [15], [21] and specialized retraining schedulers [8], [11] are built on dedicated resources, which are by design incompatible with transient and fragmented UEC slots. This limitation results in ineffective gap-filling [22] and scheduling failures. Second, much current research focuses on technical trade-offs, such as balancing performance between inference and retraining [23], [24], or managing model evolution [3], [25]. These goals, while relevant in resource-abundant environments, overlook the value optimization in UEC, which may waste resources on low-profit tasks. Importantly, most solutions are heuristic-based [6], [25], lacking theoretical performance guarantees for reliable real-time decision-making under uncertainties of task arrival and retraining duration.

Motivated by this, we aim to design an *online scheduling framework* that maximizes the total expected profit from model retraining tasks, by opportunistically exploiting these UEC resources. Addressing this problem presents several challenges:

- Making real-time and irrevocable scheduling decisions for dynamically arriving tasks, poses challenges due to limited knowledge of future arrivals, stochastic retraining durations, and fluctuating UEC availability. It must balance the immediate profit from current tasks against the uncertain but potentially larger value of future tasks. This classic dilemma often leads to the failure of simplistic, short-sighted strategies.
- Assigning the most suitable resource for each retraining task is crucial for effective UEC utilization. However, this is complicated by server heterogeneity, which presents varying capacities, and the diverse task requirements, such as delay constraints [24] and accuracy gains [25]. Suboptimal assignments waste valuable UEC resources and may negatively block future opportunities.
- Scheduling opportunistic retraining tasks directly coupled with the allocation for high-priority reserved tasks, which have specific and pre-committed resource guarantees. The core challenge is to design a unified framework that co-schedules both task types, not just to meet the demands for reserved tasks, but to strategically shape UEC slots for retraining tasks to maximize the total system profit.

To address these challenges, we introduce a novel framework for opportunistic online scheduling of model retraining on underutilized edges, which we call RESCUE (REtraining SCheduling on Underutilized Edges). Specifically, rather than reactive gap-filling [22], RESCUE employs a proactive UEC resource shaping, functioning in a two-stage manner: First, the offline stage formulates the scheduling as a stochastic optimization that models both high-priority reserved tasks and opportunistic retraining tasks. By solving the linear programming (LP) relaxation of this problem, RESCUE establishes an optimal benchmark and critical value functions that estimate the future profit of resources. Second, guided by these offline values, RESCUE online assigns each arriving retraining task to an appropriate *server-service profile pairs (SSPs)*, where service profile sets the retraining duration. It compares the immediate profit of accepting the task against the long-term value of reserving resources for potentially more valuable future tasks, thereby navigating trade-offs in online scheduling under uncertainty. Our contributions are summarized as follows:

- We are the first to formulate the problem of maximizing profit from *online* model retraining as a stochastic joint-optimization problem, which co-schedules opportunistic retraining tasks within UEC resources created by high-priority reserved tasks. This unified model addresses a key limitation of prior works, which typically assume dedicated resource pools and fail to systematically transform wasted UEC resources into valuable opportunities.
- We design RESCUE, a novel *online scheduling framework*, which includes: (1) The offline stage solves an LP relaxation and calculates value functions that price future resource availability; (2) The online stage then uses the offline guidance to make irrevocable scheduling decisions, achieving a provable $1/2$-competitive ratio.
- We conduct extensive experiment evaluations on realistic workloads, showcasing that RESCUE maintains a high empirical competitive ratio of 0.51–0.87 against the offline LP-based upper bound, while achieving an average profit increase of over 60% compared to the baselines.

The remaining part of this paper is organized as follows. Related work is described in Section II. The system model is presented in Section III. Then we propose our online framework in Section IV and theoretical analysis in Section V. The experimental results are reported in Section VI. The work is concluded in Section VII.

II. Related Work

A. Online Scheduling for Edge-based Model Retraining

The need to address data drift has made online model retraining essential in edge AI applications [3], [8]. As the number of retraining tasks increases, effective scheduling of these tasks becomes crucial; thereby, a significant body of research optimizes this scheduling process. For instance, some work design systems for balancing resources between inference and retraining [6], [23], and other work [25], [26] integrates model retraining into complex workflows, maximizing model accuracy and meeting delay constraints.

Although these studies enhance edge intelligence, they often rely on simplifying assumptions. First, they tend to model a dedicated and non-interfering resource pool for scheduling [6], [23], neglecting the fragmented and transient resource gaps created by pre-committed and high-priority reserved tasks. Second, many assume fixed or predictable retraining durations [8], [11], failing to schedule tasks with dynamic edge resources and stochastic execution times in practical scenarios. Motivated by this, we aim to develop an online scheduler that directly addresses dynamic resource availability and uncertainties in task arrival and retraining duration.

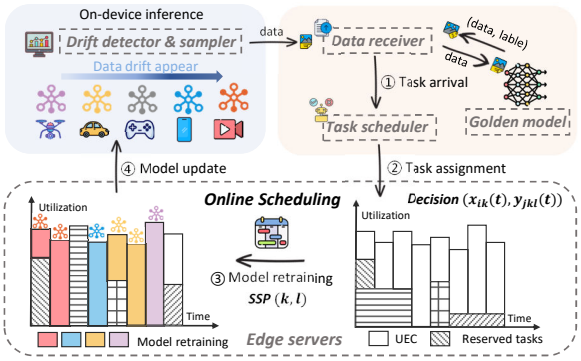

Fig. 2. System model and workflow.

B. Resource-aware Task Scheduling

To enhance system resource utilization, existing task schedulers use various advanced methods. Proactive strategies, like overbooking [16], enable resource bookings to exceed supply; resource profiling [21] establishes optimal resource ratio profiles; and contention-aware allocation [27] optimizes task placement to avoid resource contention. While reactive approaches adjust allocation based on demand fluctuations through auto-scaling [14], [15], fill idle gaps with small tasks using backfilling [22], and estimate resource-performance mappings via online feedback schemes [28].

However, these solutions are not suitable for the UEC scenario. First, they often lack fine-grained mechanisms to proactively exploit UEC resources while maintaining the performance of reserved tasks. Second, their objectives misalign with our profit-driven context, neglecting the characteristics of retraining tasks, such as delay sensitivity [24] and variable accuracy gains [25]. Moreover, many approaches rely on heuristics without offering theoretical performance guarantees, limiting their applicability in real-time decision-making, and highlighting the need for a provably optimal online framework.

III. System Model and Problem Formulation

A. System Overview

We consider an edge computing system, as illustrated in Fig. 2, which mainly comprises a set of devices $\mathcal{J} = \{1, 2, \cdots, J\}$, a set of edge servers $\mathcal{K} = \{1, 2, \cdots, K\}$, and a central online task scheduler. The system operates over a time horizon $\mathcal{T} = \{1, 2, \cdots, T\}$, divided into T time slots.

System Workflow. The system operates in a four-step cycle: ① *Task Arrival*. Upon detecting data drift, a device triggers a retraining task with sampled data and pseudo-labels from a golden model. ② *Task Assignment*. The central scheduler receives the task and decides whether to accept it. If accepted, it assigns the task to an optimal edge server with a retraining profile, which we later define as a *server-service profile pair*. ③ *Model Retraining*. The assigned server performs retraining using the selected profile, opportunistically leveraging UEC resources without interfering with high-priority reserved tasks. ④ *Model Update*. Upon completion, the updated model is deployed back to the device for enhanced inference.

Online Arriving Retraining Tasks. As introduced in the workflow, retraining tasks are triggered on the device side. Specifically, each device $j \in \mathcal{J}$ is equipped with a specific compressed DNN model for inference, which may suffer from performance degradation due to data drift. To counteract this issue, each device runs an adaptive data drift detector and a drift-aware data sampling module [3]. When the inference accuracy of the model is detected to fall below a threshold, the device triggers a retraining task r_j with selected new data samples φ_j to enhance model accuracy. Task r_j arrives at the online scheduler at time t following a known adversarial distribution (KAD) [29], [30], represented by $\{\{p_j(t)\}_j\}_t$. It is well-grounded in practice that the time-varying task arrival probability can be estimated from historical logs reflecting foreseeable operational patterns [31]. While our framework assumes stationarity for analysis, it can adapt to non-stationary environments by periodically re-estimating probabilities in the offline stage. In addition, a "golden model" is employed at the edge, pre-trained on a large dataset for high accuracy. Due to its high deployment cost on devices, the model is used only to generate pseudo-labels for each incoming retraining task. [6].

UEC from Prioritized Reserved Tasks. On the edge side, each server $k \in \mathcal{K}$ has heterogeneous computing resources, denoted as $c_k(t)$, measured in teraflops per time slot. The edge environment is complicated by a set of high-priority *reserved tasks*, denoted as $\mathcal{H}_k$, which are essential for business operations like periodic data analytics and critical control loops. Each reserved task $i \in \mathcal{H}_k$ has a reserved window $[s_i^k, d_i^k]$ and a total computing demand ϕ_{ik}, which must be satisfied. Typically, the resources available within a task's window exceed its demand, *i.e.*, , $\phi_i^k \leq \sum_{t \in [s_i^k, d_i^k]} c_k(t)$, a common practice of over-provisioning that leads to resource underutilization. This, along with fragmentation between task windows, creates idle resources termed UEC. For flexible scheduling, let $x_{ik}(t)$ be the fraction of $c_k(t)$ allocated to reserved task i at t, allowing non-contiguous execution.

Retraining Service Profile and Scheduling. The online scheduler confronts an irrevocable decision for each arriving retraining task: whether to accept it and, if so, how to schedule it. This assignment is formalized by selecting an optimal *server-service profile pair* (SSP), (k, l), represented by the binary variable $y_{jkl}(t)$. The server k is from set $\mathcal{K}$, while the service profile l is selected from an ordered set $\mathcal{L} = \{1, 2, \ldots, L\}$ [6], [23]. Each profile $l \in \mathcal{L}$ defines a computational workload (*e.g.,* a specific number of training epochs), with a stochastic retraining duration d_l^r. This duration follows a known distribution D_l, affected by factors like communication jitter and CPU/GPU scaling [32]. Critically, higher-indexed profiles have longer expected durations, such that if $l' < l$, then $\mathbb{E}[d_{l'}^r] < \mathbb{E}[d_l^r]$. Selecting a higher-indexed profile may improve accuracy, but its prolonged use of UEC resources risks blocking future tasks, and this decision must ensure sufficient resources for high-priority reserved tasks.

Decision Variable. (1) $x_{ik}(t)$: Represents the fraction of server k's capacity allocated to reserved task i at time t. (2) $y_{jkl}(t)$: A binary variable where $y_{jkl}(t) = 1$ indicates that r_j is

assigned to SSP (k, l) at arrival time t; otherwise, $y_{jkl}(t) = 0$.

B. Profit Model

We quantify the benefit of scheduling a retraining task by modeling its profit based on expected accuracy gains, stochastic retraining duration, and system constraints.

Delay Model. In practice, to ensure the retraining quality of service, retraining tasks are often constrained by a Service Level Agreement (SLA) deadline, denoted as d_{max}. We analyze the end-to-end delay of the model training task, which comprises data transmission over wireless networks and the retraining process. Based on the Shannon formula, the transmission rate between device j and edge server k at time t is modeled as $r_{jk} = B_{jk} \log(1+\varrho_j g_{jk}^t/N_0)$, where ϱ_j represents the transmission power of device j, g_{jk}^t is the channel gain between device j and server k at time t, and N_0 is the noise power. The transmission delay thus for uploading data φ_j and downloading the model φ_j' is $d_{jk}^t = (\varphi_j + \varphi_j')/r_{jk}(t)$. This results in a remaining time budget for model retraining, given by $\overline{d_l^r} = d_{max} - d_{jk}^t$. Hence, profit from retraining is generated only if the retraining duration satisfies $d_l^r \leq \overline{d_l^r}$.

Accuracy Gain Model. We denote the accuracy gain of each retraining task r_j with an SSP (k, l) at time t as $\Delta\alpha_{jkl}(t)$, combining the model's potential for improvement and the gain from computational work. First, The potential for improvement is quantified by an accuracy gap factor, denoted as $a_j(t) = 1 - \alpha_j(t)/\alpha_{max}$, where $\alpha_j(t)$ is the current model accuracy and α_{max} is the theoretical maximum achievable accuracy. Here, the accuracy $\alpha_j(t)$ decays over time since its last update [33], captured by $\alpha_j(t) = \alpha_j(t_0)e^{-\gamma_j(t-t_0)}$, where t_0 is the last retraining timestamp and γ_j is the decay rate. This factor reflects that a model with a lower current accuracy has a greater potential to improve towards its maximum achievable accuracy. Second, for a retraining duration d under profile l, the computational work on server k starting at t is given by $U_{kl}(t, d) = \sum_{\tau=t}^{t+d-1} c_k(\tau)$. Then, a *resource-to-performance* function $F_j[\cdot]$ [34] is used to represent the accuracy gain from computational work, typically exhibiting a non-linear and concave form (*e.g.,* $F_j[U] = a_j \log(1 + b_j U)$ detailed in Section VI-A), showing practical diminishing returns [35].

Combining these components and considering the stochastic retraining duration, we calculate the expected accuracy gain $\mathbb{E}[\Delta\alpha_{jkl}(t)]$ by summing the gains from all possible satisfied retraining durations, *i.e.,* $d_l^r \leq \overline{d_l^r}$, weighted by their probabilities following the known distribution D_l as $\mathbb{E}[\Delta\alpha_{jkl}(t)] = a_j(t)\sum_{d=1}^{\overline{d_l^r}} \Pr(d_l^r = d) F_j[U_{kl}(t, d)]$. The expected profit $R_{jkl}(t)$ is then the weighted gain:

$$R_{jkl}(t) = w_j \mathbb{E}[\Delta\alpha_{jkl}(t)], \tag{1}$$

where w_j is a service prioritization factor of task r_j. This profit metric guides online decisions, linking accuracy gains to economic incentives in UEC.

C. Problem Formulation

Our objective is to find an optimal scheduling policy for both opportunistic retraining tasks, characterized by the variables $\{y_{jkl}(t)\}$, and pre-committed reserved tasks, denoted by $\{x_{ik}(t)\}$. This policy aims to maximize the total expected profit from retraining tasks, while ensuring that all high-priority reserved tasks are completed within their specified time windows and resource constraints. We formulate this challenge as a stochastic integer program, termed Problem P1:

$$\begin{aligned}
\text{(P1)} \quad \max \; & \sum_{t,j,k,l} p_j(t) y_{jkl}(t) R_{jkl}(t) \\
\textit{s.t.:} \; & \sum_{k,l} y_{jkl}(t) \leq 1, \quad \forall j, t, && \text{(2a)} \\
& \underbrace{\sum_{j,l} \sum_{t' \leq t} p_j(t') y_{jkl}(t') \Pr(d_l^r \geq t - t' + 1)}_{\text{Expected fraction for retraining tasks}} \\
& \quad + \underbrace{\sum_{i \in \mathcal{H}_k} x_{ik}(t)}_{\text{Fraction for reserved tasks}} \leq 1, \quad \forall k, t, && \text{(2b)} \\
& y_{jkl}(t) \in \{0, 1\}, \quad \forall j, k, l, t, && \text{(2c)} \\
& \sum_{t=s_i^k}^{d_i^k} c_k(t) x_{ik}(t) \geq \phi_i^k, \quad \forall k, i, && \text{(2d)} \\
& 0 \leq x_{ik}(t) \leq 1, \quad \forall k, i, t, && \text{(2e)} \\
& x_{ik}(t) = 0, \quad \forall k, i, t \notin [s_i^k, d_i^k]. && \text{(2f)}
\end{aligned}$$

Here, Eq. (2a) ensures each arriving task is assigned to only one SSP. Eq. (2b) maintains that, for any server k at any time t, the *expected* resources used by active retraining tasks, along with those allocated to reserved tasks, do not exceed the server's total capacity. Eq. (2d) guarantees that each reserved task receives its total required computational resources ϕ_i^k within its designated time window, utilizing fractional resource allocation. Finally, Eqs. (2c), (2e), (2f) define the domains of our decision variables.

Problem Challenges. This formulation presents a challenging online stochastic integer program. The core difficulties stem from: (1) *Online Decision-Making under Uncertainty*: Decisions on binary variables $y_{jkl}(t)$ must be made irrevocably at time t with limited foresight into future task arrivals and stochastic retraining durations d_l^r, which introduce uncertainty in resource occupation constrained by SLA deadlines. (2) *Tight Temporal Coupling in Resource Constraints*: Eq. (2b) imposes a time-extended coupling among variables, where any assignment $y_{jkl}(t) = 1$ occupies resources over an uncertain future interval, impacting the subsequent feasibility and optimality. This structure makes myopic approaches highly suboptimal.

IV. RESCUE Framework Design

To tackle the challenges of Problem P1, we now propose RESCUE, an online scheduling framework that bridges long-term optimization with immediate but irrevocable decisions. We first present its high-level design overview as follows.

A. Design Overview

The core of RESCUE is an LP-guided threshold-based online scheduling algorithm that seamlessly integrates offline

Algorithm 1: RESCUE Framework

Input: $\{p_j(t)\}, \mathcal{K}, \mathcal{J}, \mathcal{L}, \mathcal{T}$

// **Offline Planning Stage**

Calculate $\{R_{jkl}(t)\}$, $\forall j, k, l, t$ by Eq. (1);

$y^* \leftarrow$ Solve(P1-LP);

Calculate $\{\mathcal{A}_{jkl}(t)\}, \{\mathcal{B}_k(t)\}$ by Algorithm 2;

// **Online Decision Stage**

for $t \leftarrow 1$ *to* T **do**

  if *task* r_j *arrives at* t **then**

    Select an SSP (k, l) with probability $y^*_{jkl}(t)/p_j(t)$;

    if $\mathcal{A}_{jkl}(t) > \mathcal{B}_k(t+1)$ *and* k *is available* **then**

      Assign r_j on server k with profile l;

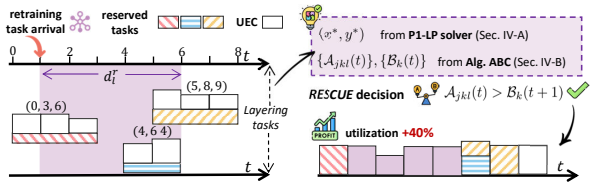

Fig. 3. An example of RESCUE. An edge server has committed resources to three reserved tasks, *e.g.,* task $(0, 3, 6)$ needs resources of 6 between $t = 0$ and $t = 3$, which creates a UEC environment. When a retraining task r_j arrives at $t = 1$, RESCUE uses its value functions to weigh the trade-off of accepting it versus reserving resources for the future. By scheduling the new task into the UEC gaps, it boosts overall system utilization by 40%.

planning with real-time decision-making. Specifically, it is built upon a two-stage approach: (1) In the offline planning stage, RESCUE solves the LP relaxation of the stochastic optimization problem (termed P1-LP), which provides a provable upper bound on the optimal solution of P1 and represents a globally optimized long-term task scheduling. This solution is then used to compute a set of value functions via Dynamic Programming (DP), which estimate the future profitability of various system states and decisions, This computation is detailed in Algorithm 2. (2) In the online decision stage, RESCUE operates with limited foresight into future arrivals. For each incoming task, it first leverages the offline optimal value with probability to select a promising SSP candidate. Then, it applies the value functions to decide whether to accept the task with this SSP or reserve resources for potentially higher-value future opportunities. This online procedure is outlined in Algorithm 1, with Fig. 3 illustrating a toy example of the key concepts behind RESCUE.

Remark. Our framework strategically decouples the offline planning (*i.e.,* LP and DP) from the highly efficient online execution. While the offline phase has polynomial-time complexity, scaling with the size of $\mathcal{J}, \mathcal{K}, \mathcal{L}, \mathcal{T}$, its structure allows for standard decomposition techniques, such as column generation [36], to handle large-scale instances. The resulting policy is then employed in the online stage, where decisions are made with minimal complexity, involving only a lookup and comparison and spreading heavy computation costs across numerous rapid, real-time scheduling decisions.

B. Offline Optimal Value

Given a problem instance, a sequence of online retraining tasks follows the arrival probability distribution $\mathcal{I}$. An offline algorithm, with complete knowledge of the task arrival sequence, can calculate the maximum profit for any I. The expected profit over the distribution $\mathcal{I}$, denoted as $\mathbb{E}_{I\sim\mathcal{I}}[\text{OPT}(I)]$, provides the theoretical upper bound for any online algorithm's performance, which corresponds to the optimal solution of the stochastic integer program P1.

Due to the complexity of P1, we formulate the LP relaxation of P1, denoted as P1-LP, to establish a computable benchmark. In P1-LP, the binary decision variable $y_{jkl}(t)$ is relaxed to be a fractional value in $[0, 1]$, which also represents the *joint probability* that task r_j arrives at time t *and* is assigned to SSP (k, l). The variable $x_{ik}(t)$ remains fractional as defined in P1. The resulting P1-LP formulation is as follows:

$$\text{(P1-LP)} \quad \max \sum_{t,j,k,l} y_{jkl}(t) R_{jkl}(t)$$

$$s.t.: \quad \sum_{k,l} y_{jkl}(t) \le p_j(t), \quad \forall j, t, \tag{3a}$$

$$\sum_{j,l} \sum_{t' \le t} y_{jkl}(t') \Pr(d_l^r \ge t - t' + 1) + \sum_{i \in \mathcal{H}_k} x_{ik}(t) \le 1, \quad \forall k, t, \tag{3b}$$

$$y_{jkl}(t) \ge 0, \quad \forall j, k, l, t, \tag{3c}$$

$$Eqs.\ (2d) - (2f).$$

Eq. (3a) ensures that the total assignment probability of a task does not exceed its arrival probability, and the other constraints directly mirror those in P1.

Remark. We use $\text{OPT}(LP)$ to denote the optimal value of P1-LP, defined by a pair of decisions (x^*, y^*), where $x^* = \{x^*_{ik}(t)\}$ is the optimal fractional allocation for reserved tasks, shaping the UEC resource view, and $y^* = \{y^*_{jkl}(t)\}$ is the optimal probabilistic policy for accepting retraining tasks *within* this UEC. Based on Eq. (3b), y^* inherently captures the resource trade-offs imposed by x^*. This enables us to utilize y^* as a reliable guide for real-time decision-making.

Lemma 1. *$OPT(LP) \ge \mathbb{E}_{I\sim\mathcal{I}}[OPT(I)]$.*

Proof Sketch. The proof is based on showing that any feasible solution to the stochastic integer program P1 can be converted into a feasible solution for its LP relaxation, P1-LP, with an identical objective value. Let (x^*, y^*) be an optimal solution for P1, and set $\bar{x}_{ik}(t) = x^*_{ik}(t)$, $\bar{y}_{jkl}(t) = p_j(t) y^*_{jkl}(t)$, which directly aligns with the probabilistic definitions of the variables. It can be verified that $(\bar{x}, \bar{y})$ is feasible for P1-LP. For example, $\sum_{k,l} \bar{y}_{jkl}(t) \le p_j(t)$ in P1-LP holds because $\sum_{k,l} y^*_{jkl}(t) \le 1$ in P1. The other constraints are satisfied due to the structural similarities between the two problems. Finally, the objective value in P1-LP is $\sum_{t,j,k,l} \bar{y}_{jkl}(t) R_{jkl}(t) = \sum_{t,j,k,l} p_j(t) y^*_{jkl}(t) R_{jkl}(t)$, which is exactly the optimal value of P1. Since a feasible solution for P1-LP that achieves the optimal value of P1, *i.e.,* $\text{OPT}(LP) \ge \mathbb{E}_{I\sim\mathcal{I}}[\text{OPT}(I)]$. □

Algorithm 2: Activation-Baseline Computation (ABC)

Input: $y^*, \{R_{jkl}(t)\}$
for $j \in \mathcal{J}, k \in \mathcal{K}, l \in \mathcal{L}$ **do**
  Calculate $\mathcal{A}_{jkl}(T)$ and $\mathcal{B}_k(T)$ by Eq. (6);
for $t \leftarrow T-1$ *to* 1 **do**
  for $j \in \mathcal{J}, k \in \mathcal{K}, l \in \mathcal{L}$ **do**
    Calculate $\mathcal{A}_{jkl}(t)$ by Eq. (4);
  Calculate $\mathcal{B}_k(t)$ by Eq. (5);
Output: $\{\mathcal{A}_{jkl}(t)\}, \{\mathcal{B}_k(t)\}$

C. Offline Value Function Computation

To translate the offline optimal insight from P1-LP into an effective online policy, we employ DP to pre-compute two key value functions, as detailed in Algorithm 2. These functions address the core challenges of uncertainty and temporal resource coupling, as discussed in **Problem Challenges** of P1. They quantify the trade-off between the expected profit of immediate actions and the strategic value of reserving resources for future opportunities, enabling online algorithms to make proactive and farsighted decisions.

Specifically, the *activation function* $\mathcal{A}_{jkl}(t)$ quantifies the total expected profit from accepting task r_j with SSP (k,l) at time t. It combines the immediate profit $R_{jkl}(t)$ with the expected future value of server k after the task's completion. Importantly, it considers the stochastic retraining duration d_l^r to embed future uncertainty into the present decision:

$$\mathcal{A}_{jkl}(t) = R_{jkl}(t) + \sum_{d=1}^{T-t} Pr(d_l^r = d)\mathcal{B}_k(t+d). \quad (4)$$

Here, $\mathcal{B}_k(t)$ is the *baseline function*, which represents the maximum expected profit achievable from server k from time t onwards, assuming no new retraining task is scheduled. This function reflects the "value of waiting" by reserving the server's capacity for future opportunities. Guided by the offline probabilities y^*, it recursively computes this value by weighing the option of accepting a new task against the benefits of waiting, thus capturing the temporal coupling where each decision influences future resource availability:

$$\mathcal{B}_k(t) = \sum_{j,l} y^*_{jkl}(t) \max\{\mathcal{A}_{jkl}(t), \mathcal{B}_k(t+1)\} + (1 - \sum_{j,l} y^*_{jkl}(t))\mathcal{B}_k(t+1). \quad (5)$$

These functions are computed via backward induction from the horizon T down to 1. At $t = T$, we have:

$$\begin{cases} \mathcal{A}_{jkl}(T) = R_{jkl}(T), & \forall j,k,l, \\ \mathcal{B}_k(T) = \sum_{j,l} y^*_{jkl}(T) R_{jkl}(T), & \forall k. \end{cases} \quad (6)$$

Afterward, for each $k \in \mathcal{K}, l \in \mathcal{L}, j \in \mathcal{J}, t \in \mathcal{T}$, we fill in $\mathcal{A}_{jkl}(t)$ and $\mathcal{B}_k(t)$ recursively with Eqs. (4) and (5).

D. Online Scheduling and Decision-Making

The online phase of RESCUE is executed upon the arrival of a retraining task r_j at time t. The decision process is twofold: (1) Instead of a brute-force search of SSPs, the algorithm leverages the offline optimal y^* to probabilistically identify a promising SSP. Using a form of randomized rounding [37], it selects SSP (k,l) with a probability of $y^*_{jkl}(t)/p_j(t)$. This step ensures that choices align with the globally-aware and long-term strategy derived from the LP relaxation. (2) For the selected SSP (k,l), it performs a threshold-based check by comparing $\mathcal{A}_{jkl}(t)$ (the profit from accepting the task) against $\mathcal{B}_k(t+1)$ (the profit from reserving the server for the future). A task is accepted only if the server k is available and $\mathcal{A}_{jkl}(t) > \mathcal{B}_k(t+1)$. Otherwise, the task is rejected, preserving resources for potential future tasks. When task r_j is assigned to an SSP (k,l), server k becomes blocked for a stochastic duration $d_l^r \sim D_l$ of profile l.

According to the definition of $\mathcal{B}_k(t)$ and Algorithm 1, we have the following observation.

Observation 1. *$\mathcal{B}_k(t)$ is computed via backward induction, where $\mathcal{B}_k(T+1) \triangleq 0$. This recursive calculation allows $\mathcal{B}_k(1)$ to capture the maximum value achievable on server k over the entire horizon, effectively reflecting the opportunity cost of utilizing its resources from the beginning.*

V. Theoretical Analysis

In this section, we first construct an *auxiliary instance* to derive performance bounds in Lemmas 2 and 3. Then, we prove that RESCUE achieves a tight competitive ratio of $1/2$ against the optimal offline solution in Theorems 1 and 2.

A. Auxiliary Instance

For each server, we define an independent auxiliary instance by assuming the arrival of *synthetic* tasks, aggregating tasks for each service profile, which is non-repeating in service profile $l \in \mathcal{L}$. At each time t, a synthetic task r'_{kl} arrives with probability $p'_{kl}(t)$ and yields a profit of $R'_{kl}(t)$, which are derived from the optimal solution y^* of P1-LP:

$$p'_{kl}(t) = \sum_j y^*_{jkl}(t), p'_{kl}(t)R'_{kl}(t) = \sum_j y^*_{jkl}(t)R_{jkl}(t). \quad (7)$$

The task r'_{kl} can only be scheduled on server k with profile l at time t. Similarly, we define an activation function $\tilde{\mathcal{A}}_{kl}(t)$ and a baseline function $\tilde{\mathcal{B}}_k(t)$ for this auxiliary instance:

$$\begin{cases} \tilde{\mathcal{A}}_{kl}(t) & = R'_{kl}(t) + \sum_{d=1}^{T-t} Pr(d_l^r = d)\tilde{\mathcal{B}}_k(t+d), \\ \tilde{\mathcal{B}}_k(t) & = \sum_l p'_{kl}(t) \max\{\tilde{\mathcal{A}}_{kl}(t), \tilde{\mathcal{B}}_k(t+1)\} \\ & \quad + (1 - \sum_l p'_{kl}(t))\tilde{\mathcal{B}}_k(t+1). \end{cases} \quad (8)$$

Note that we set $\tilde{\mathcal{B}}_k(t) = 0$ for $t > T$. Since $\sum_{l \in \mathcal{L}} p'_{kl}(t) \leq \sum_{j,l} y^*_{jkl}(t) \leq \sum_j p_j(t) \leq 1$, the probabilities are valid, making the auxiliary instance well defined.

B. Competitive Analysis

Now we present the two key lemmas that connect the P1 problem, the auxiliary instance, and the offline optimal value.

Lemma 2. *$\forall k \in \mathcal{K}, \forall t \in \mathcal{T}$, we have $\mathcal{B}_k(t) \geq \tilde{\mathcal{B}}_k(t)$.*

Proof. We prove this lemma by backward induction on t.

Base Case ($t = T$)**.** The property holds with equality. By their definitions, both $\mathcal{B}_k(T)$ and $\tilde{\mathcal{B}}_k(T)$ simplify to $\sum_{j,l} y^*_{jkl}(T)R_{jkl}(T)$, since future profits beyond T are 0.

Induction Step. Assume that $\mathcal{B}_k(t') \geq \tilde{\mathcal{B}}_k(t')$. $\forall t' > t$, $\mathbb{E}[\mathcal{B}_k(t + d^r_l)] \geq \mathbb{E}[\tilde{\mathcal{B}}_k(t + d^r_l)]$. By applying Eq. (5), we have

$$\begin{aligned}\mathcal{B}_k(t) &\geq \sum_l \max\{\sum_j y^*_{jkl}(t)\mathcal{A}_{jkl}(t), p'_{kl}(t)\mathcal{B}_k(t+1)\} \\ &\quad + (1 - \sum_l p'_{kl}(t))\mathcal{B}_k(t+1) \\ &\geq \sum_l p'_{kl}(t)\max\{\tilde{\mathcal{A}}_{kl}(t), \tilde{\mathcal{B}}_k(t+1)\} \\ &\quad + (1 - \sum_l p'_{kl}(t))\tilde{\mathcal{B}}_k(t+1) = \tilde{\mathcal{B}}_k(t), \end{aligned} \tag{9}$$

where the last inequality holds because the aggregated activation value is larger, *i.e.,* $\sum_j y^*_{jkl}\mathcal{A}_{jkl} \geq p'_{kl}\tilde{\mathcal{A}}_{kl}$ and from the induction hypothesis $\mathcal{B}_k(t+1) \geq \tilde{\mathcal{B}}_k(t+1)$. □

Lemma 3. $\tilde{\mathcal{B}}_k(1) \geq \frac{1}{2}\sum_t \sum_{j,l} y^*_{jkl}(t)R_{jkl}(t)$.

Proof Sketch. The proof relies on a primal-dual analysis. According to Eq. (8), we have $\tilde{\mathcal{B}}_k(t) \geq \max\{\tilde{\mathcal{B}}_k(t+1), \tilde{\mathcal{B}}_k(t+1) + \sum_l p'_{kl}(t)(\tilde{\mathcal{A}}_{kl}(t) - \tilde{\mathcal{B}}_k(t+1))$. Then, such $\tilde{\mathcal{B}}_k(t)$ can be formulated as an LP problem to find the minimum value of $\tilde{\mathcal{B}}_k(1)$ subject to the recursive constraints. By the principle of weak duality, the objective value of any feasible solution to the corresponding dual LP provides a lower bound on the optimal primal value. We associate dual variables, $\lambda_k(t) \geq 0$ and $\mu_k(t) \geq 0$, interpreted as "shadow prices", with each primal constraint. Here, $\lambda_k(t)$ corresponds to $\tilde{\mathcal{B}}_k(t) \geq \tilde{\mathcal{B}}_k(t+1)$ and represents the marginal value of maintaining this non-increasing property of the baseline function. $\mu_k(t)$ corresponds to $\tilde{\mathcal{B}}_k(t) \geq \tilde{\mathcal{B}}_k(t+1) + \sum_l p'_{kl}(t)(\tilde{\mathcal{A}}_{kl}(t) - \tilde{\mathcal{B}}_k(t+1))$ and signifies the importance or marginal value of the expected profit achievable at time t. Then, the objective of the dual LP is to maximize $\sum_{t=1}^{T} \mu_k(t)(\sum_l p'_{kl}(t)R'_{kl}(t))$.

The core of our proof is to construct such a feasible dual solution by setting the dual variables $\mu_k(t) = 1/2$ for all $t \in [1, T]$. With this assignment, the remaining dual variables, $\lambda_k(t)$, can be recursively set to satisfy all dual constraints. Substituting our feasible solution (*i.e.,* $\mu_k(t) = 1/2$) into the dual objective function directly yields the lower bound: $\tilde{\mathcal{B}}_k(1) \geq \sum_{t=1}^{T} \frac{1}{2}(\sum_l p'_{kl}(t)R'_{kl}(t)) = \frac{1}{2}\sum_t \sum_{j,l} y^*_{jkl}(t)R_{jkl}(t)$. □

Definition 1 (Competitive Ratio)**.** *Let $ALG(I)$ be the expected profit of an online algorithm ALG when the input sequence is I. We say that an online algorithm is* ***c-competitive*** *if $\mathbb{E}_{I\sim\mathcal{I}}[ALG(I)] \geq c \cdot \mathbb{E}_{I\sim\mathcal{I}}[OPT(I)], \forall I$.*

Theorem 1. *RESCUE can achieve a $1/2$-competitive ratio against the offline optimal solution.*

Proof. Let $\mathbb{E}_{I\sim\mathcal{I}}[\text{ALG}(I)]$ be the total expected profit from RESCUE. The expected profit of the offline optimal algorithm, $\mathbb{E}_{I\sim\mathcal{I}}[\text{OPT}(I)]$, is proved to be upper-bounded by the optimal value of P1-LP in Lemma 1, *i.e.,* $\text{OPT}(LP) \geq \mathbb{E}_{I\sim\mathcal{I}}[\text{OPT}(I)]$. Thus, combining Lemmas 2 and 3, we have:

$$\begin{aligned}\mathbb{E}_{I\sim\mathcal{I}}[\text{ALG}(I)] &= \sum_k \mathcal{B}_k(1) \geq \sum_k \tilde{\mathcal{B}}_k(1) \\ &\geq \frac{1}{2}\sum_t \sum_{j,k,l} y^*_{jkl}(t)R_{jkl}(t) \\ &= \frac{1}{2}\text{OPT}(LP) \geq \frac{1}{2}\mathbb{E}_{I\sim\mathcal{I}}[\text{OPT}(I)]. \end{aligned} \tag{10}$$

□

Theorem 2. *The $1/2$-competitive ratio of RESCUE is tight.*

Proof. We construct an instance: a single server over $T = 2$. At $t = 1$, task r_1 arrives with profit $R_1 = 1$. At $t = 2$, either task r_2 with probability ϵ and $R_2 = 1/\epsilon$, or task r_3 with probability $1 - \epsilon$ and $R_3 = 0$. Assume a single service profile with a fixed duration of $d^r_l = 2$, so accepting any task occupies the server for the entire horizon.

An optimal offline algorithm, knowing the full arrival sequence in advance, would reject r_1 to accept r_2 (profit $1/\epsilon$) when it appears, and would accept r_1 (profit 1) otherwise. Its total expected profit is $\mathbb{E}[\text{OPT}] = \epsilon \cdot (1/\epsilon) + (1-\epsilon) \cdot 1 = 2 - \epsilon$. In contrast, any online algorithm must make an irrevocable decision on r_1 at $t = 1$. If it accepts r_1, its profit is exactly 1. If it rejects r_1, its expected profit from future tasks is $\epsilon \cdot (1/\epsilon) + (1 - \epsilon) \cdot 0 = 1$. Thus, the maximum expected profit for any online algorithm is $\max\{1, 1\} = 1$. The resulting competitive ratio is $\mathbb{E}[ALG]/\mathbb{E}[\text{OPT}] = 1/2 - \epsilon$. As $\epsilon \to 0$, this ratio approaches $1/2$, which shows that no online algorithm can achieve a better competitive ratio. □

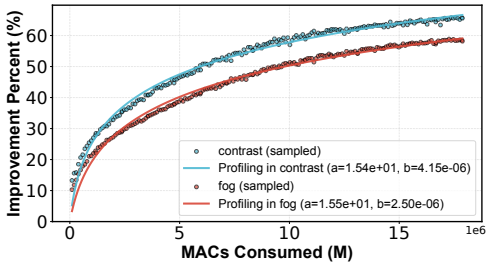

Fig. 4. Example of resource-to-performance profiling.

VI. Experiment Evaluations

A. Evaluation Settings

System Setup. We configure our evaluation within $[2, 7]$ edge servers and $[10, 18]$ devices. To model hardware heterogeneity, each server's computational capacity, $c_k(t)$, ranges from 5 to 10 tera MACs per time slot. Network parameters are configured based on common models [13], [24], where the wireless bandwidth is 3 MHz, the transmission power ϱ_j is 100 mW, the noise power N_0 is -40 dBm, and the channel gain g^t_{jk} is determined by a path loss model with device-server distances randomly ranging from 0.15 to 0.45 km.

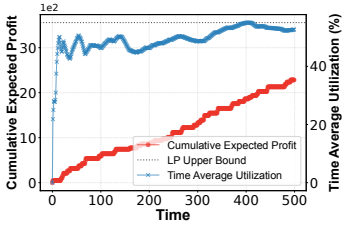

Fig. 5. Overall Performance over time.

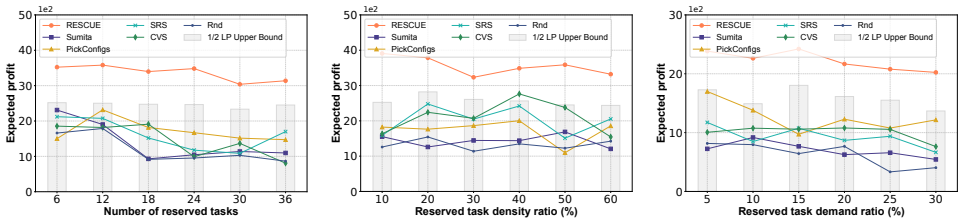

Fig. 6. Expected profit under different reserved task number/density/demand.

UEC and Reserved Tasks. To simulate different UEC environments, we first set the number of total reserved tasks to 6 to 36 dispersed in a given number of servers, and then we also introduce two key factors: a density factor to manage the total occupation ratio of reserved tasks within the range of $[10\%, 60\%]$, and a demand factor to adjust the total computational requirement ratio from $[5\%, 30\%]$. These factors create UEC resources with diverse scales and fragmentation levels.

Online Retraining Tasks. To simulate task arrival probabilities in practice, we account for fluctuations between daytime and nighttime by setting the average arrival probability within the range of $[20\%, 70\%]$, modeled using a periodic sinusoidal function. We assume the data amount uploaded to and downloaded from edge-side randomly varies between 8–32MB and 1–4MB, respectively. The average SLA deadline varies from 18 to 28 time slots, while the profit weight is randomly assigned within $[0.5, 10]$ to reflect real-world varying profits. In addition, we configure between 3–8 service profiles ($\mathcal{L}$), and the expected retraining duration for each profile follows normal, bimodal, uniform, and heavy-tail distributions to capture real-world server capacity uncertainties.

Dataset and Model Profiling. We use MobileNetV2 [4] for compressed DNNs on devices and ResNet50 [38] as the "golden model" on the edge side to generate pseudo-labels. Our experiments focus on the CIFAR-10-C dataset [39], which introduces 19 corruption types to the standard CIFAR-10 to simulate data drift. To realistically estimate the retraining profit, we perform empirical *resource-to-performance profiling* [34]. We treat each corruption type as a distinct data drift scenario and conduct extensive retraining experiments. By adjusting the number of training epochs and measuring the corresponding accuracy gain $\Delta\alpha$, we vary the computational work. This yields a set of empirical data points $(U, \Delta\alpha)$ for each drift type. Following [35], [40], which indicates diminishing returns on computational investment, we then apply a non-negative least squares (NNLS) solver [41] to fit these data points to the logarithmic function $F_j[U] = a_j \log(1 + b_j U)$. An example of the model profiling is illustrated in Fig. 4.

Baseline Algorithms. To evaluate RESCUE's performance, we compare it with these baselines. (1) *PickConfigs* [6]: When a retraining task arrives, this algorithm greedily selects the SSP that offers the highest immediate profit among all available options, while ensuring all reserved tasks can be completed. (2) *SRS* [26]: Upon task arrival, this algorithm selects the SSP by ranking priority scores that consider both the expected profit and the expected completion retraining duration. (3) *Sumita* [29]: This algorithm adopts a hybrid way. It first selects the server based on the offline LP solution y^*, similar to RESCUE, but then greedily chooses the service profile that yields the highest profit on that server. (4) *Cost-Value Selection (CVS)*: This algorithm selects the SSP based on the best cost-value ratio, where cost is a function of resource consumption. (5) *Random (Rnd)*: This algorithm randomly decides whether to accept an incoming task and, if accepted, randomly assigns it to a valid SSP.

B. Results

Overall Performance over Time. We begin by illustrating the real-time overall performance of RESCUE. As shown in Fig. 5, the red curve, tracking the actual time-averaged UEC utilization, which is an indicator of resource engagement, quickly increases and stabilizes around 53%. This efficient resource usage results in a near-linear profit accumulation, as shown in the blue curve. Ultimately, this leads to a final profit with an empirical competitive ratio of 0.64 against the LP-derived upper bound, exceeding our theoretical guarantee of $1/2$. This validates that RESCUE's core design, which dynamically balances immediate gains with the future value of resources as stated in Eqs. (4) and (5), is effective in practice.

Robustness to Resource Fragmentation. A key challenge in UEC environments is resource fragmentation. We evaluate the robustness of our framework by varying the total demand of reserved tasks, temporal density and spatial distribution, as shown in Fig. 6. These results highlight a core strength of RESCUE: while baseline performances often fluctuate and decline significantly under high fragmentation, RESCUE exhibits good resilience. Its performance advantage becomes increasingly significant as the UEC resource grows more challenging. As observed in the left part of Fig. 6, under high spatial fragmentation, *i.e.,* the number of tasks, RESCUE generates over 84% more profit than the best-performing baseline. This superiority stems from its LP-guided foresight, enabling it to intelligently navigate a complex resource environment where reactive and greedy baseline strategies inevitably fail.

Performance under Dynamic Loads and Temporal Constraints. We evaluate how algorithms adapt to system dynamics, focusing on varying system loads (via task arrival probability, the left part of Fig. 7) and temporal flexibility (via SLA deadlines, the right part of Fig. 7). Under increasing load, RESCUE's profit advantage widens, achieving 99% more

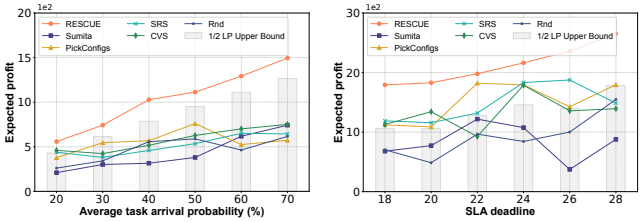

Fig. 7. Expected profit under varying task arrival probability and SLA.

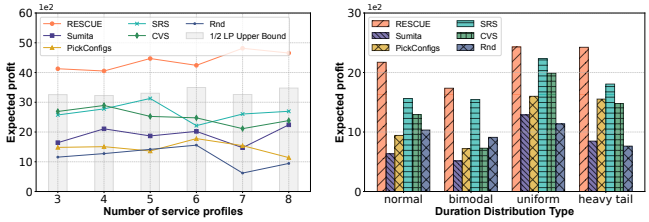

Fig. 8. Expected profit under varying service profile number and distribution.

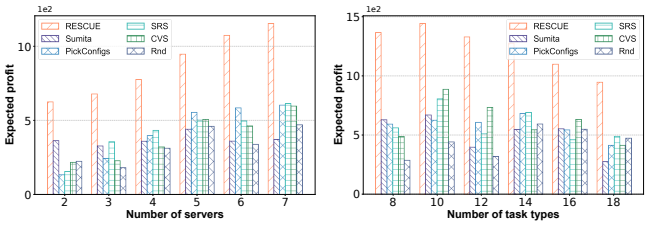

Fig. 9. Scalability on server/task type number.

profit than the next-best at a 70% arrival rate. When temporal constraints are relaxed, RESCUE leverages the increased flexibility, maintaining an empirical competitive ratio of 74%–87%. Baselines, in contrast, perform in fluctuations, which showcases that RESCUE's value-based framework correctly prices both resource competition and temporal flexibility, a crucial capability for real-world online systems.

Adaptability to Service Profile Heterogeneity. We then evaluate performance against service profile heterogeneity, which we introduce by varying the number of available profiles and the stochasticity of their duration distributions, as shown in Fig. 8. The results show that RESCUE is highly adaptive. In the left part of Fig. 8, as profiles increase, baseline performance often degrades due to a more complex decision-making environment, while RESCUE maintains a strong competitive ratio of 0.61–0.87. Its advantage is most evident in complex scenarios; for example, it outperforms the best baseline by 73% when 8 profiles are available and achieves an outstanding 87.5% competitive ratio. In addition, the right part of Fig. 8 also reveals RESCUE's adaptability to different retraining duration distributions, yielding an average of 70% more profit than the baselines. This highlights the strength of its unified decision framework, which effectively evaluates diverse and uncertain options where simple heuristics fail.

Scalability. Finally, we evaluate the scalability of our framework by increasing both resource and complexity. Specifically, we scale the number of servers and diversify task types, as shown in Fig. 9. RESCUE exhibits robust and scalable performance in both scenarios, while baseline approaches struggle to effectively leverage the larger scale. As the system size expands, RESCUE consistently maintains higher profits, achieving 83% greater profit than the best baseline with 6 servers and 98% more with 18 distinct task types. This supports that as edge computing systems grow in size and complexity, proactive and foresightful planning becomes essential. The ability of RESCUE to perform an effective estimate of the continuously expanding resources and complexities highlights its applicability to large-scale real-world applications.

Summary. The simulation results consistently and comprehensively validate the superiority of RESCUE. Quantitatively, RESCUE outperforms all baselines across all evaluated scenarios, achieving an average profit increase of over 60% compared to the next-best performing methods. Furthermore, it yields a high empirical competitive ratio against the offline LP-based upper bound, typically ranging between 0.51–0.87, robustly exceeding its $1/2$ theoretical guarantee. These results stem from a key design advantage: by applying LP-guided value functions, RESCUE replaces short-sighted heuristics with a foresightful mechanism that intelligently prices future opportunities. This core capability drives its performance, robustness, and scalability, making it an effective and reliable solution for scheduling model retraining tasks on UEC.

VII. Conclusion

In this paper, we present RESCUE, a novel online framework that effectively transforms the challenge of Underutilized Edge Computing (UEC) resources into opportunities for model retraining. By formulating the problem as a stochastic joint optimization, our LP-guided two-stage algorithm navigates real-time uncertainties to make robust scheduling decisions, backed by a provable $1/2$-competitive ratio. Extensive evaluations confirm that RESCUE not only achieves a high empirical competitive ratio of 0.51–0.87 but also yields an average profit increase exceeding 60% over baseline algorithms. This work demonstrates that a unified and foresight-driven approach can practically and efficiently unlock the vast potential of UEC, paving the way for more sustainable and powerful edge intelligence. Future work will focus on extending our framework to handle unknown task arrival distributions and validating its performance in a real-world edge testbed.

VIII. Acknowledge

This work is supported in part by National Key R&D Program of China No. 2023YFC3605800, the National Natural Science Foundation of China Grant 62472090, 62402102, Jiangsu Province Frontier Technology R&D Program No. BF2025615, the National Natural Science Foundation of Jiangsu Grants BK20242026, BK20241275, the Jiangsu Provincial Key Laboratory of Network and Information Security Grant BM2003201, and the Collaborative Innovation Center of Novel Software Technology.

REFERENCES

[1] Z. Cao, Y. Cheng, Z. Zhou, A. Lu, Y. Hu, J. Liu, M. Zhang, and Z. Li, "Patching in order: Efficient on-device model fine-tuning for multi-dnn vision applications," *IEEE Transactions on Mobile Computing*, 2024.

[2] M. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–37, 2021.

[3] L. Wang, Z. Yu, H. Yu, S. Liu, Y. Xie, B. Guo, and Y. Liu, "Adaevo: Edge-assisted continuous and timely dnn model evolution for mobile devices," *IEEE Transactions on Mobile Computing*, 2023.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[5] Y. Kong, P. Yang, and Y. Cheng, "Edge-assisted on-device model update for video analytics in adverse environments," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 9051–9060.

[6] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, Y. Shu, N. Karianakis, K. Hsieh, P. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2022, pp. 119–135.

[7] Y. Kim, C. Oh, J. Hwang, W. Kim, S. Oh, Y. Lee, H. Sharma, A. Yazdanbakhsh, and J. Park, "Dacapo: Accelerating continuous learning in autonomous systems for video analytics," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2024, pp. 1246–1261.

[8] T. Li, H. Wang, Q. Li, Y. Jiang, and Z. Yuan, "CL-Shield: A continuous learning system for protecting user privacy," *IEEE Transactions on Mobile Computing*, 2024.

[9] S. Jošilo and G. Dán, "Computation offloading scheduling for periodic tasks in mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 667–680, 2020.

[10] M. Khani, G. Ananthanarayanan, K. Hsieh, J. Jiang, R. Netravali, Y. Shu, M. Alizadeh, and V. Bahl, "RECL: Responsive resource-efficient continuous learning for video analytics," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2023, pp. 917–932.

[11] L. Wang, K. Lu, N. Zhang, X. Qu, J. Wang, J. Wan, G. Li, and J. Xiao, "Shoggoth: Towards efficient edge-cloud collaborative real-time video inference via adaptive online learning," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[12] H. Liao, G. Tang, D. Guo, Y. Wang, and R. Cao, "Rethinking low-carbon edge computing system design with renewable energy sharing," in *Proceedings of the 53rd International Conference on Parallel Processing*, 2024, pp. 950–960.

[13] H. Liu, X. Long, Z. Li, S. Long, R. Ran, and H.-M. Wang, "Joint optimization of request assignment and computing resource allocation in multi-access edge computing," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1254–1267, 2022.

[14] Y. Guo, J. Ge, P. Guo, Y. Chai, T. Li, M. Shi, Y. Tu, and J. Ouyang, "Pass: Predictive auto-scaling system for large-scale enterprise web applications," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 2747–2758.

[15] H. Qiu, W. Mao, C. Wang, H. Franke, A. Youssef, Z. T. Kalbarczyk, T. Başar, and R. K. Iyer, "Aware: Automate workload autoscaling with reinforcement learning in production cloud systems," in *2023 USENIX Annual Technical Conference (ATC)*, 2023, pp. 387–402.

[16] M. Liwang and X. Wang, "Overbooking-empowered computing resource provisioning in cloud-aided mobile edge networks," *IEEE/ACM Transactions on Networking*, vol. 30, no. 5, pp. 2289–2303, 2022.

[17] Z. Tang, F. Mou, J. Lou, W. Jia, Y. Wu, and W. Zhao, "Joint resource overbooking and container scheduling in edge computing," *IEEE Transactions on Mobile Computing*, 2024.

[18] H. Tian, S. Li, A. Wang, W. Wang, T. Wu, and H. Yang, "Owl: Performance-aware scheduling for resource-efficient function-as-a-service cloud," in *Proceedings of the 13th Symposium on Cloud Computing*, 2022, pp. 78–93.

[19] Q. Liu, Y. Yang, D. Du, Y. Xia, P. Zhang, J. Feng, J. R. Larus, and H. Chen, "Harmonizing efficiency and practicability: optimizing resource utilization in serverless computing with jiagu," in *2024 USENIX Annual Technical Conference (ATC)*, 2024, pp. 1–17.

[20] DevOps, "The cloud is booming, but so is cloud waste," https://devops.com/the-cloud-is-booming-but-so-is-cloud-waste/, 2025.

[21] W. Cao, J. Gu, Z. Ming, Z. Cai, Y. Wang, C. Ji, Z. Xiao, Y. Feng, Y. Liu, and L.-J. Zhang, "Flexible computing: A new framework for improving resource allocation and scheduling in elastic computing," *IEEE Transactions on Services Computing*, 2024.

[22] M. Zakarya, L. Gillam, M. R. C. Qazani, A. A. Khan, K. Salah, and O. Rana, "Backfillme: an energy and performance efficient virtual machine scheduler for iaas datacenters," *IEEE Transactions on Services Computing*, 2025.

[23] H. Cai, Z. Zhou, and Q. Huang, "Online resource allocation for edge intelligence with colocated model retraining and inference," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2024, pp. 1900–1909.

[24] Y. Zeng, R. Zhou, L. Jiao, Z. Han, J. Yu, and Y. Ma, "Efficient online dnn inference with continuous learning in edge computing," in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*. IEEE, 2024, pp. 1–10.

[25] S. S. Shubha and H. Shen, "Adainf: Data drift adaptive scheduling for accurate and slo-guaranteed multiple-model inference serving at edge servers," in *Proceedings of the ACM SIGCOMM 2023 Conference*, 2023, pp. 473–485.

[26] Y. Zeng, R. Zhou, L. Jiao, and R. Zhang, "Online scheduling of edge multiple-model inference with dag structure and retraining," in *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 2025, pp. 1–10.

[27] Y. Wang, L. Huang, Z. Wang, V. Kalavri, and I. Matta, "CAPSys: Contention-aware task placement for data stream processing," in *Proceedings of the Twentieth European Conference on Computer Systems*, 2025, pp. 654–670.

[28] R. Bhardwaj, K. Kandasamy, A. Biswal, W. Guo, B. Hindman, J. Gonzalez, M. Jordan, and I. Stoica, "Cilantro: Performance-aware resource allocation for general objectives via online feedback," in *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2023, pp. 623–643.

[29] H. Sumita, S. Ito, K. Takemura, D. Hatano, T. Fukunaga, N. Kakimura, and K.-i. Kawarabayashi, "Online task assignment problems with reusable resources," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 5, 2022, pp. 5199–5207.

[30] J. P. Dickerson, K. A. Sankararaman, A. Srinivasan, and P. Xu, "Allocation problems in ride-sharing platforms: Online matching with offline reusable resources," *ACM Transactions on Economics and Computation (TEAC)*, vol. 9, no. 3, pp. 1–17, 2021.

[31] A. Saadallah, L. Moreira-Matias, R. Sousa, J. Khiari, E. Jenelius, and J. Gama, "Bright—drift-aware demand predictions for taxi networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 2, pp. 234–245, 2018.

[32] Q. Wang and X. Chu, "GPGPU performance estimation with core and memory frequency scaling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 12, pp. 2865–2881, 2020.

[33] D. Vela, A. Sharp, R. Zhang, T. Nguyen, A. Hoang, and O. S. Pianykh, "Temporal quality degradation in ai models," *Scientific reports*, vol. 12, no. 1, p. 11654, 2022.

[34] S. Venkataraman, Z. Yang, M. Franklin, B. Recht, and I. Stoica, "Ernest: Efficient performance prediction for large-scale advanced analytics," in *13th USENIX symposium on networked systems design and implementation (NSDI)*, 2016, pp. 363–378.

[35] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[36] G. B. Dantzig, *Linear Programming and Extensions*. Princeton University Press, 1960.

[37] O. D. Anderson, "Kendall's advanced theory of statistics, volume 1: Distribution theory," 1988.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[39] "CIFAR-10-C," https://zenodo.org/records/2535967.

[40] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[41] SciPy, "scipy.optimize.nnls," https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.optimize.nnls.html, 2014.