Online and Predictive Coordinated Cloud-Edge Scrubbing for DDoS Mitigation

Ruiting Zhou, *Member, IEEE,* Yifan Zeng, Lei Jiao, *Member, IEEE,* Yi Zhong, and Liujing Song

Abstract—To mitigate Distributed Denial-of-Service (DDoS) attacks towards enterprise networks, we study the problem of scheduling DDoS traffic through on-premises scrubbing at the local edge and on-demand scrubbing in the remote clouds. We model this problem as a nonlinear mixed-integer program, which is characterized by the inputs of arbitrary dynamics and the trade-offs between staying at suboptimal scrubbing locations and using different best locations with switching overhead. We first design a prediction-oblivious online algorithm which consists of a carefully-designed fractional algorithm to pursue the long-term total cost minimization but avoid excessive switching overhead over time, and a randomized rounding algorithm to derive the flow-based, integral decisions. We next design a prediction-aware online algorithm which leverages the predicted inputs and can make even better scheduling decisions through invoking our prediction-oblivious online algorithm and improving its solutions via re-solving the original problem slice over each prediction window. We further extend our study to prioritize local scrubbing, and adapt our algorithms to this case correspondingly. Then, we rigorously prove the worst-case, constant competitive performance guarantees of our online algorithms. Finally, we conduct extensive evaluations and validate the superiority of our approach over multiple existing alternatives approaches.

Index Terms—DDoS mitigation, cloud scrubbing, edge scrubbing, online optimization, predictive control

1 Introduction

Scrubbing suspicious traffic through cloud scrubbing centers has been increasingly adopted recently to battle Distributed Denial-of-Service (DDoS) attacks [2]. In this approach, suspicious traffic is routed into the dedicated scrubbing centers of DDoS protection services (e.g., Cloudflare [3]) for investigation, where the malicious traffic is dropped and the legitimate traffic is injected back to the network and continues to flow to the destination. This approach has multiple advantages: scrubbing centers are often geo-distributed and widely available; scrubbing is on-demand and pay-per-use, with "unlimited" capacity; it has no upfront cost, relatively easy to manage.

However, cloud scrubbing is not a panacea, and can actually exhibit drawbacks in some scenarios such as defending enterprise networks. First, routing traffic to scrubbing centers actually deviates from the normal network path and increases the delay [2], which will impact the contained legitimate traffic towards the enterprise as well.

- *This work was supported in part by the National Natural Science Foundation of China under Grants 62072344, U20A20177, and 62232004, and in part by the U.S. National Science Foundation under Grants CNS-2047719 and CNS-2225949. A preliminary version of this work appeared in the Proceedings of the 19th IEEE International Conference on Sensing, Communication, and Networking (SECON), September 20-23, 2022 [1] (Corresponding author: Lei Jiao).*
- *R. Zhou is with the School of Computer Science and Engineering, Southeast University, Nanjing, China (e-mail: ruitingzhou@seu.edu.cn).*
- *Y. Zeng, and Y. Zhong are with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, China (e-mail: {yifanzeng, yizhong}@whu.edu.cn).*
- *L. Jiao is with the Department of Computer Science, University of Oregon, Eugene, OR, USA (e-mail: jiao@cs.uoregon.edu).*
- *L. Song is with the Computer Network Information Center, University of Chinese Academy of Sciences, Beijing, China (e-mail: songliujing@cnic.cn).*

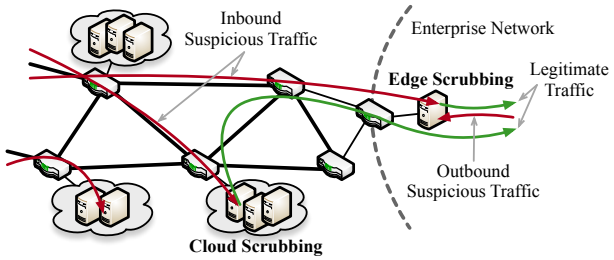

Fig. 1: DDoS mitigation via joint cloud and edge scrubbing

Second, a considerable amount of today's DDoS traffic has low volume, short duration, and is even "hit-and-run" [4], which often costs long time for the remote scrubbing centers to detect, or makes them unable to respond in time. Third, cloud scrubbing is mostly for inbound traffic, and does not help with outbound malicious traffic from within the enterprise networks [5]. To overcome these disadvantages, local edge scrubbing on enterprise premises is needed. Yet, edge scrubbing facilities often have insufficient capacity in face of volumetric DDoS flows, and therefore cloud scrubbing is still indispensable. Fig. 1 illustrates this scenario.

Operating a hybrid approach of on-demand cloud scrubbing and on-premises edge scrubbing optimally is non-trivial and entails making scheduling decisions with challenging requirements. First, the DDoS traffic and other related inputs are often time-varying, highly dynamic, and require scheduling decisions to be made repetitively in an online manner to accommodate such uncertainties with complex long-term total cost management at both local and remote scrubbing locations. Second, the scrubbing performance matters. The best location to scrub a specific flow may change as time goes; yet, switching among different scrubbing centers requires the installation and update of the routing rules (e.g., via Border Gateway Protocol) over

the networks [6], which often takes time and incurs additional delay. The scheduling decisions thus need to strike the dynamic balance between staying stably at suboptimal scrubbing locations and switching back and forth to the best locations with the switching overhead. Third, it is further preferred that scrubbing scheduling should be proactive, rather than responsive, to handle both inbound and outbound suspicious traffic. The enterprise often has the unique contextual awareness of its service and application behaviors, and can potentially better detect or predict abnormal traffic in advance [7]. Such predicted information should be exploited for further cost and performance optimization.

Unfortunately, this problem has never been investigated so far to the best of our knowledge. Existing research on DDoS mitigation with cloud/edge [2], [4], [5], [8], [9] often treats cloud and edge separately and lack systematic, performance-guaranteed algorithms for the dynamic orchestration of traffic scrubbing. Other related research includes network flow scheduling [10], [11], [12], [13], [14] and network resource allocation [15], [16], [17], [18], [19]. Yet, they neither target DDoS specifically nor capture the unique features such as bidirectional attacks, traffic redirection, and scrubbing management; their solution techniques are also inapplicable, as they fail to simultaneously address the challenges of switching cost, integer decisions, and prediction awareness, and may also miss rigorous analysis. See detailed discussions in Section 2.

In this paper, first, we model and present a mathematical formulation of the scrubbing scheduling problem. Our formulation accounts for a range of inputs, including inbound and outbound suspicious traffic for the enterprise, scrubbing cost of different remote cloud scrubbing centers, and operational cost of the local edge scrubbing facility. Our models are general and make no assumption on traffic dynamism, and each flow in our model can be appearing or disappearing over time, long-term or "hit-and-run", volumetric or low-volume, etc. We minimize the long-term total cost of scrubbing all the suspicious traffic over time, while capturing the performance overhead (i.e., switching delay) of changing scrubbing locations. Our problem turns out to be an NP-hard, nonlinear mixed-integer program.

To design polynomial-time online algorithms to solve our problem, we next study the setting where in each time slot we observe the current inputs and are required to make irrevocable decisions in response on the fly. To address the challenge of determining whether to use a better but different scrubbing location at the cost of incurring the switching cost, we relax the problem to the real domain, and design an online algorithm which introduces a carefully-designed logarithmic term [20] and uses it to perform "mild" and conservative switches over time, based on the current inputs and the previous decisions. We also propose a smart rounding algorithm to convert the real-valued solutions to integers [21], while still satisfying all constraints of the problem after rounding. We prove that our approach has a guaranteed competitive ratio parameterized by the inputs, characterizing the multiplicative gap between the cost of our online decisions and that of the offline optimum.

We then study the setting where in each time slot we have access to the predicted inputs (e.g., provided by the enterprise networks via dedicated methods such as time series analysis or recurrent neural networks) for a specified prediction window of several future time slots, so that we can proactively make even better scrubbing scheduling decisions leveraging such predictions in advance. To outperform our aforementioned prediction-free online approach, we design another novel online algorithm which, in each prediction window, invokes our prediction-free approach sequentially until the last time slot of the window, and then fixes that solution for the last time slot as the "anchor" and improves the solutions before it through re-solving the original problem slice over the prediction window.

The novelty and superiority of our algorithms are that we creatively incorporate the regularization theory into the design of our predictive online algorithm to address the scenario where the future inputs in the moving prediction window are available. Our prediction-aware approach attains a competitive ratio which is at least as good as that of our prediction-free approach, and is often better in practice.

Further, we extend our study to the case of prioritizing local scrubbing and only resorting to remote scrubbing when necessary. This approach attempts to maximally avoid the drawbacks of cloud scrubbing and make the upfront cost (e.g., one-time equipment purchase) of edge scrubbing well-spent. To that end, we revise our formulation through penalizing the waste of local scrubbing capacity and introducing additional control variables. Accordingly, by adapting our existing algorithms, we design the waste-penalty-aware prediction-free and predictive online algorithms. Different from the previous design, the new algorithm needs to round two more sets of fractions into integers. Then it puts them back to the problem, and re-uses the updated regularized fractional algorithm to update the fractional decisions. We prove the competitive ratios and show that the competitive ratio of the latter is no worse than that of the former.

Finally, we conduct extensive evaluations to validate the practical performance of our algorithms. Utilizing the public *CICDDoS2019* data [22], Amazon GuardDuty prices [23], US industrial electricity prices [24], and other realistic inputs, we run our evaluations for a 1000-second time period of DDoS attacks and observe multiple results, including the following: (1) Our prediction-free online algorithm (PFO) saves the long-term total cost by up to 28% compared to the greedy approach of always using the cheapest scrubbing location and the random approach for selecting local and remote scrubbing locations, and attains an empirical competitive ratio of less than 3.2 compared to the offline optimum; (2) Our prediction-aware online algorithm (PAO) can further reduce the total cost, saving $15\%\sim24\%$ cost over an existing sophisticated predictive online algorithm; (3) Our prediction-aware approach is robust for inaccurate predictions, e.g., achieving $17\%\sim28\%$ cost reduction when the predicted inputs have $2\%\sim6\%$ random noise for the prediction window of two time slots; (4) Our algorithms run fast, and take up to around 90 seconds cumulatively for the entire time period under consideration on a typical off-the-shelf desktop; (5) As the waste of local scrubbing capacity becomes more important, our waste-penalty-aware algorithm is more advantageous in saving total cost; (6) Our additional evaluations of the prediction-free online algorithm with waste penalty (PFOW) and the prediction-aware online algorithm with waste penalty (PAOW) observe

results of similar trends compared to PFO and PAO.

2 Related Work

DDoS Mitigation at Cloud and Edge: Zilberman et al. [2] examined the optimal placement of cloud scrubbing centers by considering factors such as traffic footprint, link load, and network latency. Myneni et al. [5] proposed a distributed defense solution against DDoS attacks, which is deployed at both customer and provider edges, utilizing neural network techniques. Bhardwaj et al. [4] employed edge functions to enhance the efficiency of DDoS attack detection. Zhou et al. [9] proposed a DDoS mitigation scheme which utilizes a flexible allocation of traffic to distributed locations in close proximity to the sources of the attacks. You et al. [8] investigated the scheduling of DDoS flows through auction mechanisms, involving the utilization of scrubbing centers. In an effort to address the Edge DDoS Mitigation (EDM) problem, He et al. [25] proposed a method that utilizes integer programming to derive optimal solutions for small-scale EDM problems and game theory to achieve sub-optimal solutions for large-scale problems. Additionally, IR et al. [26] proposed a novel defense mechanism that employs a combination of fog computing and deep learning techniques to mitigate the issue of domain mismatch and effectively handle real-time attacks within a cloud environment. Kumar et al. [27] separated benign requests and malicious requests at the container level to serve the former without interruptions. MEC-enabled VANET is vulnerable to DDoS attacks. To cope with it, Deng et al. [28] proposed a graph neural network (GNN)-based collaborative deep reinforcement learning model to generate the resource provisioning and mitigating strategy.

Previous studies have investigated cloud and edge DDoS defenses independently, rather than exploring the potential for coordination and integration between the two. Additionally, these studies have not considered the orchestration of mitigation efforts with performance guarantees. Our work differs from these previous efforts by approaching the problem from the perspective of the victim, and by taking into account both incoming and outgoing DDoS traffic in our analysis.

Network Flow Scheduling: Liu et al. [10] employed a priority-based flow scheduling approach in decentralized federated graph learning systems. Li et al. [11] proposed a unified dynamic flow and function scheduling method for addressing real-time security function enforcement issues. Lan et al. [12] examined wireless networks with continuous and dynamic flows and developed online and adaptive scheduling algorithms. Gu et al. [13] utilized Lyapunov optimization to maximize network utility while satisfying unpredictable network traffic and fairness resource allocation requirements. Gushchin et al. [14] explored deadline-constrained flow scheduling through both offline and online algorithms in their research. Tsanikidis et al. [29] designed online admission, routing, and scheduling algorithms for deadline-constrained packets under an interference graph model of wireless networks. Mao et al. [30] took into account the concept of service function chains (SFCs) and developed a method for traffic-sensitive online placement and flow routing of SFCs. Aiming at TSN chain flows, Gong et al. [31] presented a heuristic scheduling approach based on the chain flow constraints, which is utilized to globally schedule chain flows in the multi-level topology. Chung et al. [32] designed two centralized dynamic parallel flow scheduling algorithms for recursively defined network structures to find the least congested path for each flow.

These studies do not specifically address malicious or DDoS traffic, and may not be directly applicable to the problem of DDoS attacks, which have unique characteristics such as the need to consider the costs of switching traffic routes and the use of scrubbing across multiple locations. These considerations fundamentally alter the problem and have a significant impact on the design of appropriate algorithms to address it.

Resource Allocation: Liu et al. [33] proposed an online task offloading and resource allocation approach for edge-cloud orchestrated computing to minimize the latency of tasks. Fan et al. [34] proposed a collaborative scheme for service placement, task scheduling, computing resource allocation, and transmission rate allocation to minimize latency in a multi-task and multi-service scenario. Farhadi et al. [35] proposed a two-time-scale framework for jointly considering service placement and request scheduling for data-intensive applications. Wang et al. [36] presented a mixed integer linear programming (MILP) model and a three-phase heuristic for virtual machine placement and workload assignment in cooperative edge-cloud computing over backhaul networks. Alsurdeh et al. [37] provided a resource estimation and task scheduling framework to run hybrid workflows on edge and cloud computing systems. Additionally, there are other researches that consider the impact of switching cost on scheduling. Ouyang et al. [15] examined the placement and migration of services in edge environments through the use of multi-armed bandits. Gao et al. [16] proposed laziness-inspired approaches to balance access, switching, and communication delay. Wang et al. [17] developed regularization-based online methods for allocating edge resources to mobile users. Krishnasamy et al. [18] optimized wireless network energy by managing base station status switching through Markov-chain-based approaches. Chen et al. [19] used game theory to investigate drone network interference and channel switching mitigation. Li et al. [38] studied online optimization with switching costs for predicted inputs from a purely algorithmic perspective. Yang et al. [39] defined the problem of delay-aware network function placement and routing and proposed a randomized approximation algorithm to solve it. Ren et al. [40] proposed a solution for the problem of jointly orchestrating services and managing requests while ensuring compliance with service agreements. Liu et al. [41] developed a model for computing resource allocation, request assignment, and VM instance deployment to minimize total latency in completing requests.

These studies employ a variety of solution techniques, but do not often consider *all* of the challenges present in our problem, such as the intractability of integer variables, the need for online solutions, the importance of prediction-awareness, and the competitive analysis for predictive algorithms. These challenges are non-trivial and unique to our problem.

3 Model and Formulation

3.1 System Modeling

Suspicious Flows: We consider an enterprise that utilizes traffic scrubbing to mitigate DDoS attacks. There are incoming suspicious flows that originate from external attackers. There can also exist outgoing suspicious flows, coming from compromised servers within the enterprise or from internal attackers toward outside the enterprise. We study the entire system over a series of consecutive time slots $\mathcal{T}$, and consider a set $\mathcal{J}$ of incoming suspicious flows. For $j \in \mathcal{J}$ and $t \in \mathcal{T}$, we use $\lambda_{jt} \in \{1, 0\}$ to denote whether the flow j appears in the system at the time slot t, and use σ_{jt} to denote the traffic volume of the flow j at the time slot t. To reflect arbitrary flow dynamism, we make no assumption on how λ_{jt} and σ_{jt} vary across j and t; we capture the outgoing suspicious flow dynamism below.

Edge Scrubbing: This enterprise has deployed a local, on-premises scrubbing facility (e.g., server cluster) for scrubbing the flows. While the enterprise always uses the local facility to scrub the outgoing flows, we denote by C_t, $\forall t \in \mathcal{T}$ the residual available capacity of the local scrubbing facility that can be used for scrubbing incoming flows at the time slot t. We use d to denote the operational cost (e.g., electricity price) as using the local facility to scrub one unit of traffic.

Cloud Scrubbing: We consider a set $\mathcal{I}$ of remote cloud scrubbing centers that are often geographically distributed and operated by one or multiple operator(s). The enterprise can redirect any incoming suspicious flows to the scrubbing centers through deploying BGP rules in the networks, so that after scrubbing, the clean flows are routed back to the enterprise. For $i \in \mathcal{I}$ and $t \in \mathcal{T}$, we use b_{it} to denote the price that needs to be paid by the enterprise to the scrubbing center operator for scrubbing one unit traffic at the scrubbing center i at the time slot t. For $i \in \mathcal{I}$ and $j \in \mathcal{J}$, we use c_{ij} to denote the routes setup overhead (e.g., the amount of time it takes to propagate BGP rules across routers in wide area networks) for redirecting the flow j to the scrubbing center i.

Control Decisions: The enterprise needs to make the following scheduling decisions as time goes: $y_{ijt} \in \{1, 0\}$, denoting whether or not to use the remote scrubbing center i to scrub the incoming flow j at time t, and $z_{jt} \in [0, 1]$, denoting the proportion of the local scrubbing capacity allocated for scrubbing the incoming flow j at time t.

We have summarized all such notations in Table 1.

TABLE 1: Notations and Descriptions

Inputs	Description
λ_{jt}	whether incoming flow j appears at time t
σ_{jt}	traffic volume of flow j at time t
C_t	available local scrubbing capacity at time t
d	operational cost of scrubbing one unit traffic locally
b_{it}	price of scrubbing one unit traffic at scrubbing center i at t
c_{ij}	overhead of setting up routes to redirect flow j to scrubbing center i
a	penalty of wasting one unit local scrubbing capacity
Outputs	**Description**
x_{jt}	whether to use local scrubbing facility for flow j at t
y_{ijt}	whether to use remote scrubbing center i for flow j at t
z_{jt}	proportion of local scrubbing capacity allocated for flow j at t
w_t	proportion of local scrubbing capacity wasted at t

Total Cost: Having all these notations, we can now represent the different components of the total cost of the system over the entire time horizon: $\sum_t \sum_j dC_t z_{jt}$ is the operational cost for scrubbing incoming flows locally; $\sum_t \sum_i \sum_j b_{it}\sigma_{jt}y_{ijt}$ is the cost of scrubbing incoming flows in the remote cloud scrubbing centers; and $\sum_t \sum_i \sum_j c_{ij}(y_{ijt} - y_{ijt-1})^+$, where $(\cdot)^+ \stackrel{\text{def}}{=} \max\{\cdot, 0\}$, is the "switching cost" of setting up routes to change the target cloud scrubbing centers. Via our residual-capacity-based modeling, we do not need to consider the cost of scrubbing outgoing flows, since scrubbing them locally is the default action of the enterprise and cannot be optimized.

Prediction Window: We also study the setting where the inputs can be predicted over the prediction window. That is, as time goes to the time slot t, we have access to the inputs for all the w time slots of $\{t, t+1, ..., t+w-1\}$, where w is the length of prediction window, even though we still cannot access the inputs beyond the prediction window. Here, "inputs" refer to λ_{jt}, σ_{jt}, C_t, and so on. Such predicted inputs can be provided by the enterprise via statistical or machine learning techniques [7]. While focusing on accurate predictions in our algorithm design and theoretical analysis, we also experiment with inaccurate predictions later.

3.2 Problem Formulation and Challenges

Problem Formulation: We formulate the total cost minimization problem $\mathbf{P}$ in the following:

$$
\begin{aligned}
\min \quad & P = \textstyle\sum_t \sum_j dC_t z_{jt} + \sum_t \sum_i \sum_j b_{it}\sigma_{jt}y_{ijt} \\
& \quad + \textstyle\sum_t \sum_i \sum_j c_{ij}(y_{ijt} - y_{ijt-1})^+ \\
\text{s.t.} \quad & z_{jt} + \textstyle\sum_i y_{ijt} \geqslant \lambda_{jt}, && \forall j, \forall t, && \text{(1a)} \\
& z_{jt} + \textstyle\sum_i y_{ijt} \leqslant 1, && \forall j, \forall t, && \text{(1b)} \\
& C_t z_{jt} + \sigma_{jt} \textstyle\sum_i y_{ijt} \geqslant \sigma_{jt}, && \forall j, \forall t, && \text{(1c)} \\
& \textstyle\sum_j z_{jt} \leqslant 1, && \forall t, && \text{(1d)} \\
& y_{ijt} \in \{0, 1\}, && \forall i, \forall j, \forall t, && \text{(1e)} \\
& z_{jt} \geqslant 0, && \forall j, \forall t. && \text{(1f)}
\end{aligned}
$$

Constraints (1a) and (1b) ensure that every single incoming flow is scrubbed and must be scrubbed in either the local scrubbing facility or a remote scrubbing center. Constraint (1c) ensures that the traffic of each incoming flow is scrubbed completely. Constraint (1d) ensures that the allocated local capacity cannot exceed the residual available capacity of the local scrubbing facility. Constraints (1e) and (1f) enforce the domains of the control variables.

Settings and Challenges: We face fundamental challenges when solving the above problem. The first challenge is *online decision-making*. In this paper, we consider two settings of "prediction-free" and "prediction-aware", respectively, depending on the availability of predictions. In the former setting, our algorithms run as time goes and make control decisions irrevocably on the fly for each time slot by observing the inputs for only that time slot and no inputs of the future. This imposes a challenge because, for example, at the time slot $t-1$, to minimize the switching cost $\sum_i \sum_j c_{ij}(y_{ijt} - y_{ijt-1})^+$, it would not be straightforward to determine y_{ijt-1} as we do not know y_{ijt}; y_{ijt} should be considered as an input to the time slot $t-1$, but will only be determined at the next time slot t. In the latter setting, at each time slot t, we have access to future inputs within the prediction window beyond t. It is yet non-trivial to exploit such future inputs to make provably better decisions than in

Algorithm 1: Prediction-Free Online Control Algorithm (PFO)

1 **for** $t = 1, 2, 3, ...$
2 With $\hat{\mathbf{y}}_{\mathbf{t-1}}$ as input, invoke **Algorithm 2** to solve $\hat{\mathbf{P}}_{\mathbf{t}}$ to get the solution $\{\hat{\mathbf{y}}_{\mathbf{t}}, \hat{\mathbf{z}}_{\mathbf{t}}\}$;
3 Invoke **Algorithm 3** to round $\hat{\mathbf{y}}_{\mathbf{t}}$ to $\bar{\mathbf{y}}_{\mathbf{t}}$;
4 With $\bar{\mathbf{y}}_{\mathbf{t-1}}$ and $\bar{\mathbf{y}}_{\mathbf{t}}$ as input, invoke **Algorithm 2** to solve $\hat{\mathbf{P}}_{\mathbf{t}}$ to get the solution $\{\bar{\mathbf{y}}_{\mathbf{t}}, \hat{\mathbf{z}}^*_{\mathbf{t}}\}$;

Algorithm 2: Regularized Fractional Algorithm, $\forall t$

Solve $\hat{\mathbf{P}}_{\mathbf{t}}$ using any standard convex program solver:

$$\begin{aligned} \min \quad & \hat{P}_t = \textstyle\sum_i \sum_j b_{it}\sigma_{jt}y_{ijt} + \sum_j dC_t z_{jt} \\ & + \textstyle\sum_i \sum_j \frac{c_{ij}}{\delta}((y_{ijt}+\varepsilon)\ln\frac{y_{ijt}+\varepsilon}{y_{ijt-1}+\varepsilon} - y_{ijt}) \\ \text{s.t.} \quad & \text{(1a)} \sim \text{(1d)}, \text{ without } \text{“}\forall t\text{”}; y_{ijt} \geqslant 0, z_{jt} \geqslant 0. \end{aligned}$$

the former setting with no future inputs. The second challenge is *intractability*. Our problem is essentially a nonlinear mixed-integer program. Even in the offline setting where all the inputs are observable all at once before the start of the time horizon, the problem is NP-hard. Solving it online faces only escalated difficulty.

4 Online Algorithms

To overcome all the aforementioned challenges, we firstly design Algorithm 1, which invokes Algorithms 2 and 3, for the prediction-free setting; based on this approach, we further design Algorithm 4 for the prediction-aware setting.

4.1 Algorithm for the Prediction-Free Setting

Algorithm 1: Algorithm 1 is the overall control algorithm for the prediction-free setting. At t, Algorithm 1 takes the fractional solution $\hat{\mathbf{y}}_{\mathbf{t-1}}$ from $t-1$ as input and invokes Algorithm 2 to get the fractional solution $\{\hat{\mathbf{y}}_{\mathbf{t}}, \hat{\mathbf{z}}_{\mathbf{t}}\}$ for t. Next, it invokes Algorithm 3 to round $\hat{\mathbf{y}}_{\mathbf{t}}$ to integers $\bar{\mathbf{y}}_{\mathbf{t}}$. Further, since $\hat{\mathbf{y}}_{\mathbf{t}}$ has been updated to $\bar{\mathbf{y}}_{\mathbf{t}}$, it needs to update $\hat{\mathbf{z}}_{\mathbf{t}}$ as well—it puts $\bar{\mathbf{y}}_{\mathbf{t}}$ back into the problem and re-invokes Algorithm 2 to get the updated fractional solution $\hat{\mathbf{z}}^*_{\mathbf{t}}$. That is, eventually, the solution for t is $\{\bar{\mathbf{y}}_{\mathbf{t}}, \hat{\mathbf{z}}^*_{\mathbf{t}}\}$.

Algorithm 2: Algorithm 2 splits the problem into a series of one-shot problem slices at each corresponding t, introduces a carefully-designed logarithmic term to replace the switching cost in the objective function, and solves this transformed one-shot problem slice at each t, denoted as $\hat{\mathbf{P}}_{\mathbf{t}}$, by only using control decisions of the previous time slot $t-1$. The transformation to $\hat{\mathbf{P}}_{\mathbf{t}}$ allows us to overcome the blindness to future inputs while still making provably good decisions for the current time slot, as shown in our theoretical analysis later.

To derive the concrete formulation of $\hat{\mathbf{P}}_{\mathbf{t}}$ at t, we use the logarithmic term $\frac{1}{\delta}((y_{ijt}+\varepsilon)\ln\frac{y_{ijt}+\varepsilon}{y_{ijt-1}+\varepsilon} - y_{ijt})$ to replace the original switching cost term $(y_{ijt} - y_{ijt-1})^+$ [20], while relaxing $\mathbf{y}_{\mathbf{t}}$ into the real domain, where $\varepsilon > 0$ and $\delta = \ln(1+\frac{1}{\varepsilon})$ are configurable parameters. Note that in Algorithm 2, at t, $\mathbf{y}_{\mathbf{t-1}}$ is no longer a decision variable but an input, i.e., $\hat{\mathbf{y}}_{\mathbf{t-1}}$ or $\bar{\mathbf{y}}_{\mathbf{t-1}}$, as exhibited in Algorithm 1.

Algorithm 3: Algorithm 3 iteratively selects a pair of the fractional decisions and randomly rounds them up and down, respectively, to produce at least one integer in every iteration while guaranteeing their sum does not change after rounding [21]. It also preserves the expectation of the rounded integer to be equal to the corresponding fraction before rounding, a property also useful in our theoretical analysis later.

Algorithm 3: Randomized Rounding Algorithm, $\forall t$

1 $\mathcal{I}'_t \overset{\text{def}}{=} \mathcal{I} \backslash \{i | \hat{\mathbf{y}}_{\mathbf{t}} \in \{0,1\}\}$;
2 **while** $|\mathcal{I}'_t| > 1$
3 Select $i_1, i_2 \in \mathcal{I}'_t$, where $i_1 \neq i_2$;
4 $\rho_1 \overset{\text{def}}{=} \min\{1 - \hat{y}_{i_1jt}, \hat{y}_{i_2jt}\}$;
5 $\rho_2 \overset{\text{def}}{=} \min\{\hat{y}_{i_1jt}, 1 - \hat{y}_{i_2jt}\}$;
6 With the probability $\frac{\rho_2}{\rho_1+\rho_2}$, Set $\hat{y}'_{i_1jt} = \hat{y}_{i_1jt} + \rho_1, \hat{y}'_{i_2jt} = \hat{y}_{i_2jt} - \rho_1$; With the probability $\frac{\rho_1}{\rho_1+\rho_2}$, Set $\hat{y}'_{i_1jt} = \hat{y}_{i_1jt} - \rho_2, \hat{y}'_{i_2jt} = \hat{y}_{i_2jt} + \rho_2$;
7 **if** $\hat{y}'_{i_1jt} \in \{0,1\}$
8 Set $\bar{y}_{i_1jt} = \hat{y}'_{i_1jt}, \mathcal{I}'_t = \mathcal{I}'_t \backslash \{i_1\}$;
9 **else** Set $\hat{y}_{i_1jt} = \hat{y}'_{i_1jt}$;
10 **endif**
11 **if** $\hat{y}'_{i_2jt} \in \{0,1\}$
12 Set $\bar{y}_{i_2jt} = \hat{y}'_{i_2jt}, \mathcal{I}'_t = \mathcal{I}'_t \backslash \{i_2\}$;
13 **else** Set $\hat{y}_{i_2jt} = \hat{y}'_{i_2jt}$;
14 **endif**
15 **endwhile**
16 **if** $|\mathcal{I}'_t| = 1$
17 Set $\bar{y}_{it} = 1$ for the only $i \in \mathcal{I}'_t$;
18 **endif**

This algorithm actually has multiple critical properties: (i) at least one of the two selected fractions is rounded into an integer after every iteration in the loop of Lines 2∼15, i.e., either $\hat{y}_{i_1jt}$, or $\hat{y}_{i_2jt}$, or both are rounded into integer(s); (ii) the weighted sum of the two fractions keeps unchanged before and after rounding i.e., we have $\hat{y}'_{i_1jt} + \hat{y}'_{i_2jt} = \hat{y}_{i_1jt} + \hat{y}_{i_2jt}$, no matter which case we choose in Line 6; (iii) the expectation of the integral $\bar{y}_{ijt}$ equals the fractional $\hat{y}_{ijt}$, i.e., $E(\bar{y}_{ijt}) = \hat{y}_{ijt}, \forall i \in \mathcal{I} \backslash \mathcal{I}'_t$—for example, if $\hat{y}_{i_2jt}$ becomes integral, then $E(\bar{y}_{i_2jt}) = \frac{\rho_2}{\rho_1+\rho_2}(\hat{y}_{i_2jt} - \rho_1) + \frac{\rho_1}{\rho_1+\rho_2}(\hat{y}_{i_2jt} + \rho_2) = \hat{y}_{i_2jt}$; (iv) after the loop, there is at most one fraction left. To satisfy the constraints of our problem, we just round it up, as in Line 17.

4.2 Algorithm for the Prediction-Aware Setting

Suppose a prediction window contains w time slots, then the entire time horizon can be divided into a series of prediction windows starting at the time slots 1, $w+1$, $2w+1$... At the first time slot of a prediction window, we can observe all the inputs for the prediction window and can thus make control decisions altogether for every single time slot of the prediction window. This is actually the setting of standard Fixed Horizon Control (FHC) [38]; however, as FHC has no explicit consideration for switching cost, it does not have guaranteed performance in our case and thus motivates our Algorithm 4.

Algorithm 4: Algorithm 4 first performs in each prediction window the same way as in the prediction-free setting to reach the very last time slot of the window, then fixes the decisions in this last time slot as the “anchor”, and finally resolves the original problem $\mathbf{P}$ over the prediction window while rounding the fractional decisions. We can thus connect the prediction-aware setting to the prediction-free setting,

Algorithm 4: Prediction-Aware Online Control Algorithm (PAO)

1 **for** $t = 1, w+1, 2w+1, 3w+1, ...$
2 **for** $\tau = t, t+1, ..., t+w-1$
3 With $\hat{\mathbf{y}}_{\tau-1}$as input, use **Algorithm 2** to get $\hat{\mathcal{S}}_\tau$;
4 **endfor**
5 With $\hat{\mathcal{S}}_{t-1}$ and $\hat{\mathcal{S}}_{t+w-1}$ as inputs, use any standard convex solver to minimize $\sum_{\tau=t}^{t+w-1} P_\tau$ and get the solutions $\{\mathcal{S}_t^*, ..., \mathcal{S}_{t+w-2}^*, \hat{\mathcal{S}}_{t+w-1}\}$;
6 **for** $\tau = t, t+1, ..., t+w-2$
7 Invoke **Algorithm 3** to round $\mathcal{S}_\tau^*$ into $\dot{\mathcal{S}}_\tau$;
8 **endfor**
9 Invoke **Algorithm 3** to round $\hat{\mathcal{S}}_{t+w-1}$ into $\dot{\mathcal{S}}_{t+w-1}$;

and beat the latter in terms of the cost evaluated by the objective function of $\mathbf{P}$.

This algorithm executes as follows. We use $\hat{\mathcal{S}}_t$ to denote $\{\hat{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}_\mathbf{t}\}$ obtained from Algorithm 2 at t. For the prediction window $\{t, ..., t+w-1\}$, in Lines 2∼4, we proceed as in the prediction-free setting until we get $\hat{\mathcal{S}}_{t+w-1}$, that is, we get $\{\hat{\mathcal{S}}_t, ..., \hat{\mathcal{S}}_{t+w-2}, \hat{\mathcal{S}}_{t+w-1}\}$. Then, fixing $\hat{\mathcal{S}}_{t+w-1}$, we minimize $\sum_{\tau=t}^{t+w-1} P_\tau$ over the prediction window, subject to all our constraints of (1a)∼(1e), (2a), and (2b), as in Line 5. Finally, we round the fractional solutions $\{\mathcal{S}_t^*, ..., \mathcal{S}_{t+w-2}^*, \hat{\mathcal{S}}_{t+w-1}\}$ into the mixed-integer solutions $\{\dot{\mathcal{S}}_t, ..., \dot{\mathcal{S}}_{t+w-2}, \dot{\mathcal{S}}_{t+w-1}\}$ in Lines 6∼8. Algorithm 4 connects to Algorithm 1, as both algorithms produce exactly the same solutions for the time slots $\{w, 2w, 3w, ...\}$; however, Algorithm 4 is better overall, as it has access to future inputs in the prediction window and utilizes such inputs to improve the solutions for all the other time slots of each prediction window over the time horizon.

Note that how predictions are produced by external predictors, whether they need to be synchronized, and how much time is needed to produce such predictions are out of the scope of our work. The predictions may be generated in real time, or generated offline in advance, or generated in a somewhat hybrid manner, depending on the specific prediction mechanisms and algorithms that are used—all these issues are not the focus of our paper. We only assume the predictions are available and gradually fed to our algorithms as time goes, and our focus is to design online algorithms that can use such predictions on the fly to make better control decisions compared to the case of having no predictions. If the predictions are not available due to prediction lag, synchronization, convergence, and other issues, then our prediction-aware online algorithms will just not apply (but our prediction-oblivious online algorithms will still apply because they do not require predictions).

5 Performance Analysis

5.1 Overview

To characterize the performance of our algorithms, we will rigorously prove that the total cost incurred by the online solutions from our algorithms is no greater than a constant, called the "competitive ratio", times the offline optimal cost. The competitive ratio represents how competitive our online algorithms could be, when compared to the offline optimum. Here, online algorithms can only access the inputs as they are gradually revealed on the fly, and the offline optimum has access to the inputs over the entire time horizon at hindsight.

For the prediction-free setting (i.e., Algorithm 1, which invokes Algorithms 2 and 3), we will prove

$$\begin{aligned} & E(P(\{\bar{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}_\mathbf{t}^*, \forall t\})) && \text{(3a)}\\ \leqslant & r_2 P(\{\hat{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}_\mathbf{t}, \forall t\}) && \text{(3b)}\\ \leqslant & r_2 P''(\{\hat{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}_\mathbf{t}, \forall t\}) && \text{(3c)}\\ \leqslant & r_1 r_2 D && \text{(3d)}\\ \leqslant & r_1 r_2 P''_{opt} && \text{(3e)}\\ \leqslant & (r_1 r_2 + M) P_{opt}. && \text{(3f)} \end{aligned}$$

Here, our goal is to connect (3a) to (3f), via utilizing the intermediate steps (3b)∼(3e) as the bridge. In (3a), we have the expectation because of the randomization in our rounding algorithm. In (3a)⩽(3b) which corresponds to Algorithm 3, r_2 characterizes the "loss" of the rounding process. In (3c), P'' is the objective function of the problem $\mathbf{P}''$, where $\mathbf{P}''$ is an equivalent reformulation of the problem $\mathbf{P}'$ by introducing an additional parameter M and $\mathbf{P}'$ is a relaxed problem from our original problem $\mathbf{P}$. Thus, (3b)⩽(3c) naturally holds. In (3c)⩽(3d) which corresponds to Algorithm 2, D is the objective function of the problem $\mathbf{D}$ which is the Lagrange dual problem of the problem $\mathbf{P}''$. r_2 characterizes the connection between P'' and D. Then, note that we automatically have (3d)⩽(3e) due to duality, where P''_{opt} is the offline optimum of the problem $\mathbf{P}''$. Finally, in (3e)⩽(3f), P_{opt} is the offline optimum of our original problem $\mathbf{P}$. Overall, the competitive ratio is $r_1 r_2 + M$ for our prediction-free online approach.

For the prediction-aware setting (i.e., Algorithm 4, which also invokes Algorithms 2 and 3), we will prove

$$\begin{aligned} & E(P(\dot{\mathcal{S}}_1, ..., \dot{\mathcal{S}}_{w-1}, \dot{\mathcal{S}}_w, \dot{\mathcal{S}}_{w+1}, ..., \dot{\mathcal{S}}_{2w-1}, \dot{\mathcal{S}}_{2w}, ...)) && \text{(4a)}\\ \leqslant & r_2 P(\mathcal{S}_1^*, ..., \mathcal{S}_{w-1}^*, \hat{\mathcal{S}}_w, \mathcal{S}_{w+1}^*, ..., \mathcal{S}_{2w-1}^*, \hat{\mathcal{S}}_{2w}, ...) && \text{(4b)}\\ \leqslant & r_2 P''(\mathcal{S}_1^*, ..., \mathcal{S}_{w-1}^*, \hat{\mathcal{S}}_w, \mathcal{S}_{w+1}^*, ..., \mathcal{S}_{2w-1}^*, \hat{\mathcal{S}}_{2w}, ...) && \text{(4c)}\\ \leqslant & r_2 r_3 P''(\hat{\mathcal{S}}_t, \forall t) && \text{(4d)}\\ \leqslant & (r_1 r_2 r_3 + M) P_{opt}. && \text{(4e)} \end{aligned}$$

We highlight that (4a)⩽(4b) is analogous to (3a)⩽(3b), with the same r_2; (4b)⩽(4c) is analogous to (3b)⩽(3c); and (4d)⩽(4e) is analogous to (3c)⩽(3f), with the same r_1 and M. Our only focus here will be (4c)⩽(4d), based on the design of Algorithm 4. r_3 characterizes the benefit of using the predicted inputs in the prediction window. The competitive ratio is $r_1 r_2 r_3 + M$ for our prediction-aware online approach.

Based on the above, we will prove and find out r_1, r_2, r_3, and M, respectively, in the following subsections.

5.2 Analysis of Regularized Fractional Algorithm

We prove (3c)⩽(3d) and find out r_1.

Formulating the Problems $\mathbf{P}'$ and $\mathbf{P}''$. We firstly relax our original problem $\mathbf{P}$ into the following problem $\mathbf{P}'$:

$$\begin{aligned} \min \quad & P' = \textstyle\sum_t P_t \\ & = \textstyle\sum_t \sum_i \sum_j b_{it}\sigma_{jt} y_{ijt} + \sum_t \sum_i \sum_j c_{ij} u_{ijt} \\ & + \textstyle\sum_t \sum_j d C_t z_{jt} \\ \text{s.t.} \quad & \text{(1a)}\sim\text{(1d)}, \\ & u_{ijt} \geqslant y_{ijt} - y_{ijt-1}, \qquad \forall i, \forall j, \forall t, \qquad \text{(5a)} \end{aligned}$$

$$u_{ijt} \geqslant 0, y_{ijt} \geqslant 0, z_{jt} \geqslant 0, \qquad \forall i, \forall j, \forall t. \quad (5b)$$

$\mathbf{P}'$ relaxes the control variables of $\mathbf{P}$ to the real domains and also introduces the auxiliary variables $\boldsymbol{u}_t$ with the constraints $u_{ijt} \geqslant y_{ijt} - y_{ijt-1}$ and $u_{ijt} \geqslant 0$ to equivalently linearize $\mathbf{P}$.

We then transform $\mathbf{P}'$ into an equivalent problem $\mathbf{P}''$:

$$\begin{aligned}
\min \quad & P'' = \textstyle\sum_t P'_t \\
& = \textstyle\sum_t \sum_i \sum_j b_{it}\sigma_{jt} y_{ijt} + \sum_t \sum_i \sum_j c_{ij} u_{ijt} \\
& + \textstyle\sum_t \sum_j dC_t z_{jt} + M \sum_t (\sum_j z_{jt} + Q_t) \\
\text{s.t.} \quad & \text{(1a)}, \\
& \textstyle\sum_j z_{jt} - z_{jt} + \sum_i \sum_j y_{ijt} - \sum_i y_{ijt} \geqslant \sum_j \lambda_{jt} - 1 \\
& \qquad \forall j, \forall t, \qquad (6a) \\
& \tfrac{C_t}{\sigma_{jt}} z_{jt} + \textstyle\sum_i y_{ijt} \geqslant 1, \qquad \forall j, \forall t, \qquad (6b) \\
& \textstyle\sum_j z_{jt} + Q_t \geqslant 1, \qquad \forall t, \qquad (6c) \\
& u_{ijt} \geqslant y_{ijt} - y_{ijt-1}, \qquad \forall i, \forall j, \forall t, \qquad (6d) \\
& u_{ijt} \geqslant 0, x_{jt} \geqslant 0, y_{ijt} \geqslant 0, z_{jt} \geqslant 0, \\
& \qquad \forall i, \forall j, \forall t. \qquad (6e)
\end{aligned}$$

We transfer constraint (1b) into its equivalent form (6a) with the help of (1a). Constraint (1d) can be seen as $\sum_j z_{jt} + Q_t = 1$, where $Q_t \geqslant 0$. It is equivalent to $\sum_j z_{jt} + Q_t \leqslant 1$ and $\sum_j z_{jt} + Q_t \geqslant 1$. We next introduce a relatively large positive number M and change the objective function P' to $P'' = P' + \sum_t M(\sum_j z_{jt} + Q_t)$. When P'' reaches its minimum value, the value of the objective function depends on the item $\sum_t M(\sum_j z_{jt} + Q_t)$ since M is large, and at the same time, $\sum_j z_{jt} + Q_t$ will be minimized. Combined with $\sum_j z_{jt} + Q_t \geqslant 1$ and the minimum value of $\sum_j z_{jt} + Q_t$, we can achieve $\sum_j z_{jt} + Q_t = 1$. Therefore, constraint $\sum_j z_{jt} + Q_t \leqslant 1$ can be omitted. Now, as the problem $\mathbf{P}''$ reaches its offline optimum, the problem $\mathbf{P}'$ also reaches the offline optimum. Note that M depends on the inputs, e.g., the inputs can be normalized before fed into the problem so that M can be small.

Deriving the Lagrange Dual Problem **D.** We further derive the dual problem for $\mathbf{P}''$, denoted as $\mathbf{D}$, where α_{jt}, β_{jt}, γ_{jt}, μ_{jt}, ξ_t, τ_t, and φ_{ijt} are the dual variables:

$$\begin{aligned}
\max \quad & D = \textstyle\sum_t \sum_j \lambda_{jt}\alpha_{jt} + \sum_j \sum_t (\sum_j \lambda_{jt} - 1)\beta_{jt} \\
& + \textstyle\sum_t \sum_j \mu_{it} + \sum_t (1 - Q_t)\xi_t \\
\text{s.t.} \quad & \alpha_{jt} + \textstyle\sum_j \beta_{jt} - \beta_{jt} + \frac{C_t}{\sigma_{jt}}\mu_{jt} + \xi_t \leqslant M + dC_t, \\
& \qquad \forall j, \forall t, \qquad (7a) \\
& \alpha_{jt} + \textstyle\sum_j \beta_{jt} - \beta_{jt} + \mu_{jt} - \varphi_{ijt} + \varphi_{ijt+1} \leqslant b_{it}\sigma_{jt}, \\
& \qquad \forall j, \forall t, \qquad (7b) \\
& \varphi_{ijt} \leqslant c_{ij}, \qquad \forall i, \forall j, \forall t, \qquad (7c) \\
& \xi_t \leqslant M, \qquad \forall t, \qquad (7d) \\
& \text{dual variables} \geqslant 0.
\end{aligned}$$

Bounding. Now, we are ready to prove (3c)⩽(3d) and find out r_1. We actually have the following theorem:

Theorem 1. *$P''(\{\hat{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}_\mathbf{t}, \forall t\}) \leqslant r_1 D(\{\Omega(\hat{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}_\mathbf{t}), \forall t\})$, where $r_1 = 2 + (1 + |\mathcal{I}|\varepsilon)\delta$.*

Proof. See details in Appendix A. We show that the solutions solved from the series of the problems $\hat{\mathbf{P}}_\mathbf{t}$, $\forall t$ as defined in Algorithm 2, if put together over time and evaluated in the objective function of $\mathbf{P}''$, can make the objective value P'' upper-bounded by a constant r_1 times the objective value D of the dual problem $\mathbf{D}$. We derive the proof using the Karush-Kuhn-Tucker (KKT) conditions—note that this is vastly different from traditional KKT analysis, because we use the KKT conditions for $\hat{\mathbf{P}}_\mathbf{t}$, $\forall t$, rather than those for $\mathbf{P}''$.

5.3 Analysis of Randomized Rounding Algorithm

We prove (3a)⩽(3b) and find out r_2.

Theorem 2. *$E(P(\{\bar{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}^*_\mathbf{t}, \forall t\})) \leqslant r_2 P(\{\hat{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}_\mathbf{t}, \forall t\})$, where $r_2 = \Psi_y + \Psi_z$, $\Psi_y = |\mathcal{J}|\Psi'_{yz}\Psi''_{yz}$, $\Psi_z = |\mathcal{J}| \max_{i,j} c_{ij}\Psi''_{yz}$, $\Psi'_{yz} = \max_{i,j,t} b_{i,t}\sigma_{jt} + dC_t$, and $\Psi''_{yz} = \max_{i,j,t} \frac{1}{b_{it}\sigma_{jt}} + \frac{1}{dC_t}$.*

Proof. See details in Appendix B.We exploit the multiple critical properties of Algorithm 3, which have been described in Section 4.1, to complete the proof.

5.4 Analysis of Benefit of Prediction-Awareness

We prove (4c)⩽(4d) and find out r_3.

Theorem 3. *$P''(\mathcal{S}^*_1, ..., \mathcal{S}^*_{w-1}, \hat{\mathcal{S}}_w, \mathcal{S}^*_{w+1}, ..., \mathcal{S}^*_{2w-1}, \hat{\mathcal{S}}_{2w}, ...) \leqslant P''(\hat{\mathcal{S}}_t, \forall t)$, i.e., $r_3 = 1$.*

Proof. First, we exhibit that if $\{\hat{\mathcal{S}}_1, ..., \hat{\mathcal{S}}_T\}$ denote any feasible solution to $\mathbf{P}''$, and $\{\mathcal{S}^*_\tau, \mathcal{S}^*_{\tau+1}, ..., \mathcal{S}^*_{\kappa-1}\}$ denote the optimal solution to $\mathbf{P}''$ for the time slots $\{\tau, \tau+1, ..., \kappa-1\}$ given $\hat{\mathcal{S}}_\kappa$, where $1 \leqslant \tau < \kappa \leqslant T$, then we have

$$\begin{aligned}
& P''(\{\hat{\mathcal{S}}_1, ..., \hat{\mathcal{S}}_{\tau-1}, \mathcal{S}^*_\tau, \mathcal{S}^*_{\tau+1}, \mathcal{S}^*_{\kappa-1}, \hat{\mathcal{S}}_\kappa, ..., \hat{\mathcal{S}}_T\}) \qquad (8a) \\
\leqslant & P''(\{\hat{\mathcal{S}}_1, ..., \hat{\mathcal{S}}_T\}). \qquad (8b)
\end{aligned}$$

This is due to the following. Since $\{\mathcal{S}^*_\tau, \mathcal{S}^*_{\tau+1}, ..., \mathcal{S}^*_{\kappa-1}\}$ are optimal over the time slots $\{\tau, \tau+1, ..., \kappa-1\}$, we naturally have $P''(\{\mathcal{S}^*_\tau, ..., \mathcal{S}^*_{\kappa-1}, \hat{\mathcal{S}}_\kappa\}) \leqslant P''(\{\hat{\mathcal{S}}_\tau, ..., \hat{\mathcal{S}}_\kappa\})$ for $\{\tau, \tau+1, ..., \kappa-1\}$; we can then add both $P''(\{\hat{\mathcal{S}}_1, ..., \hat{\mathcal{S}}_{\tau-1}\})$ and $P''(\{\hat{\mathcal{S}}_\kappa, \hat{\mathcal{S}}_{\kappa+1}, ..., \hat{\mathcal{S}}_T\})$ to each side of this inequality, and then get (8a)⩽(8b). Second, the proof to this theorem is complete by applying (8a)⩽(8b) repetitively via setting $\tau = (n-1)w + 1$ and $\kappa = nw$, where $n = 1,2,...$, according to Algorithm 4.

We finally prove (3e)⩽(3f) and find out M. We put it here as M is the last parameter in our competitive ratio(s).

Theorem 4. *$r_1 r_2 P''_{opt} \leqslant (r_1 r_2 + M)P_{opt}$, where M is as introduced in the problem $\mathbf{P}''$.*

Proof. P'' can be seen as $P' + M\sum_t(\sum_j z_{jt} + Q_t)$. When the problem $\mathbf{P}''$ reaches its offline optimum, P' in P'' also equals its optimum, with $\sum_j z_{jt} + Q_t = 1$. Then, we have

$$\begin{aligned}
& r_1 r_2 P''_{opt} \\
\leqslant & r_1 r_2 (P'_{opt} + MT) \\
\leqslant & r_1 r_2 (P_{opt} + MT) \\
\leqslant & r_1 r_2 (P_{opt} + M P_{opt}) \\
\leqslant & (r_1 r_2 + M) P_{opt}.
\end{aligned}$$

6 Prioritizing Local Scrubbing

Due to the drawbacks of cloud scrubbing as stated in Section 1, one may desire to prioritize the usage of the local on-premises facility and only resort to remote scrubbing centers if the suspicious traffic's volume exceeds the local capacity. We study this case in this section. We penalize the waste of the local scrubbing facility at each time slot in the

optimization objective, and also introduce two additional control decisions. Specifically, in Section 6.1 we formulate the new problem with the waste penalty; in Section 6.2 we adapt our proposed algorithms to solving the new problem; and in Section 6.3 we analyze the performance bounds of the adapted algorithms.

6.1 System Model

Capacity Waste: We use a to denote the penalty for wasting one unit capacity of the local scrubbing facility. The enterprise often pays an upfront expense (e.g., equipment and materials) for adopting the on-premises scrubbing facility for a projected lifespan (e.g., several years), and incurs penalty of waste if the facility is not fully used. We reasonably assume that this lifespan is longer than the length of the time horizon under consideration for DDoS mitigation.

Control Decisions: The enterprise makes two more types of control decisions: (i) $x_{jt} \in \{1, 0\}$, which denotes whether or not to conduct local scrubbing for the incoming flow j at the time slot t; and (ii) $w_t \geqslant 0$, i.e., $w_t = 1 - \sum_j z_{jt}$, which represents the proportion of the wasted local scrubbing capacity at t. Due to the dependence of w_t on z_{jt}, w_t can also be considered as an auxiliary variable.

Total Cost: We still minimize the total cost from the enterprise victim's perspective. Now, we decompose the total cost into four distinct components as follows: two of which have been previously explained in the preceding texts; $\sum_t \sum_j d\sigma_{jt}x_{jt}$ is the operational cost of scrubbing flows locally; and $\sum_t aw_t$ is the penalty of waste of the local scrubbing resources.

Problem Formulation: We formulate the new version of the total cost minimization problem $\mathbf{P_u}$ below:

$$
\begin{aligned}
\min \quad & P_u = \textstyle\sum_t \sum_j d\sigma_{jt}x_{jt} + \sum_t \sum_i \sum_j b_{it}\sigma_{jt}y_{ijt} \\
& + \textstyle\sum_t \sum_i \sum_j c_{ij}(y_{ijt} - y_{ijt-1})^+ + \sum_t aw_t \\
\text{s.t.} \quad & x_{jt} + \textstyle\sum_i y_{ijt} \geqslant \lambda_{jt}, && \forall j, \forall t, && \text{(10a)} \\
& x_{jt} + \textstyle\sum_i y_{ijt} \leqslant 1, && \forall j, \forall t, && \text{(10b)} \\
& x_{jt} \geqslant z_{jt}, && \forall j, \forall t, && \text{(10c)} \\
& C_t z_{jt} + \sigma_{jt} \textstyle\sum_i y_{ijt} \geqslant \sigma_{jt}, && \forall j, \forall t, && \text{(10d)} \\
& w_t + \textstyle\sum_j z_{jt} = 1, && \forall t, && \text{(10e)} \\
& x_{jt} \in \{0, 1\}, y_{ijt} \in \{0, 1\}, && \forall i, \forall j, \forall t, && \text{(10f)} \\
& z_{jt} \geqslant 0, w_t \geqslant 0, && \forall j, \forall t. && \text{(10g)}
\end{aligned}
$$

Constraints (10a) and (10b) correspond to (1a) and (1b). Constraint (10c) indicates that the capacity of the local scrubbing can be allocated to flow j only when flow j uses the local scrubbing facility. Constraint (10d) is the same as (1c). Constraint (10e) ensures that the sum of the allocated and the wasted proportions is one. Constraints (10f) and (10g) capture the domains of the decision variables.

6.2 Algorithm Design

To solve $\mathbf{P_u}$, we adapt Algorithms 1∼4 to form a set of new algorithms, i.e., Algorithms 5∼8, correspondingly.

Notations: We introduce several new notations for the sake of presenting our new algorithms. $\mathbf{P_u}$ is our problem formulated above. $\mathbf{P'_u}$ is the relaxed problem from $\mathbf{P_u}$, by relaxing all integral variables into the real domains. $\mathbf{P_{ut}}$ is the one-shot problem of $\mathbf{P'_u}$ at the time slot t. $\hat{\mathbf{P}}_{\mathbf{ut}}$ is the transformed problem from $\mathbf{P_{ut}}$ via replacing the switching cost term by our designated logarithmic term. We use P_u, P'_u, P'_{ut}, and $\hat{P}_{ut}$ to denote the objective functions of $\mathbf{P_u}$, $\mathbf{P'_u}$, $\mathbf{P_{ut}}$, and $\hat{\mathbf{P}}_{\mathbf{ut}}$, respectively. We write $\{\mathbf{x_t}, \mathbf{y_t}, \mathbf{z_t}, \mathbf{w_t}\}$ as the concise representation for $\{x_{jt}, y_{ijt}, z_{jt}, w_t, \forall i, j\}$ at t. Other notations will be just explained as they are used.

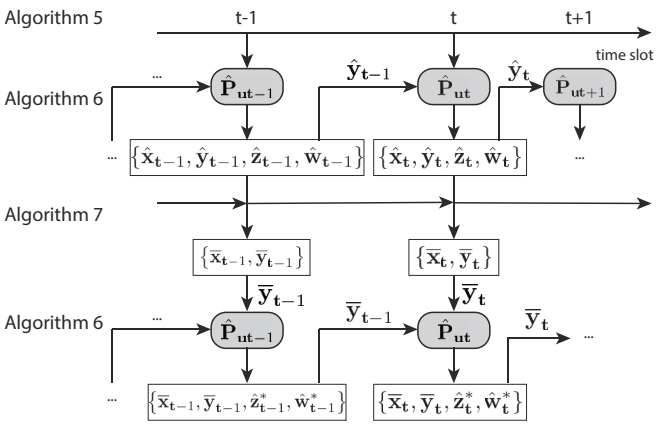

Fig. 2: Sequential execution of prediction-free online algorithm

Algorithm 5: Algorithm 5 is the major algorithm for the scenario in which there is no forecasting but the waste penalty is imposed. It repeatedly invokes Algorithms 6 and 7 at each time slot.

As shown in Fig. 2, at t, Algorithm 5 inputs the fractional decision $\hat{\mathbf{y}}_{\mathbf{t-1}}$ from the time slot $t-1$ and calls Algorithm 6 to acquire the fractional decision $\{\hat{\mathbf{x}}_\mathbf{t}, \hat{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}_\mathbf{t}, \hat{\mathbf{w}}_\mathbf{t}\}$. Then, it invokes Algorithm 7 to round $\{\hat{\mathbf{x}}_\mathbf{t}, \hat{\mathbf{y}}_\mathbf{t}\}$ into integers $\{\bar{\mathbf{x}}_\mathbf{t}, \bar{\mathbf{y}}_\mathbf{t}\}$. Further, it fixes $\{\bar{\mathbf{x}}_\mathbf{t}, \bar{\mathbf{y}}_\mathbf{t}\}$, puts them into the problem, and reuses Algorithm 6 to update the fractional decision $\{\hat{\mathbf{z}}^*_\mathbf{t}, \hat{\mathbf{w}}^*_\mathbf{t}\}$. The solution for the time slot t is therefore $\{\bar{\mathbf{x}}_\mathbf{t}, \bar{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}^*_\mathbf{t}, \hat{\mathbf{w}}^*_\mathbf{t}\}$.

Algorithm 6: Algorithm 6 solves the "regularized" one-shot problem at t to get the fractional solutions for t. We first show the relaxed problem $\mathbf{P'_u}$ of our original problem $\mathbf{P_u}$:

$$
\begin{aligned}
\min \quad & P'_u = \textstyle\sum_t P_u t \\
& = \textstyle\sum_t \sum_j d\sigma_{jt}x_{jt} + \sum_t \sum_i \sum_j b_{it}\sigma_{jt}y_{ijt} \\
& + \textstyle\sum_t \sum_i \sum_j c_{ij}u_{ijt} + \sum_t aw_t \\
\text{s.t.} \quad & \text{(1a)} \sim \text{(1e)}, \\
& u_{ijt} \geqslant y_{ijt} - y_{ijt-1}, && \forall i, \forall j, \forall t, && \text{(11a)} \\
& u_{ijt} \geqslant 0, x_{jt} \geqslant 0, y_{ijt} \geqslant 0, z_{jt} \geqslant 0, w_t \geqslant 0, \\
& && \forall i, \forall j, \forall t. && \text{(11b)}
\end{aligned}
$$

Relaxing all variables to the real domains, in the above, we have introduced P_{ut} as the one-shot objective and used auxiliary variable $\boldsymbol{u}_t$ with the corresponding constraints in order to linearize this problem. That is, like Algorithm 2, we use the logarithmic term $\frac{1}{\delta}((y_{ijt} + \varepsilon) \ln \frac{y_{ijt}+\varepsilon}{y_{ijt-1}+\varepsilon} - y_{ijt})$ to substitute the original switching cost term $(y_{ijt} - y_{ijt-1})^+$ in $\mathbf{P_{ut}}$ and thus compose the regularized problem $\hat{\mathbf{P}}_{\mathbf{ut}}$.

Algorithms 7 and 8: Algorithm 7 rounds fractional $\hat{\mathbf{y}}_\mathbf{t}$ and $\hat{\mathbf{x}}_\mathbf{t}$ to integral $\bar{\mathbf{y}}_\mathbf{t}$ and $\bar{\mathbf{x}}_\mathbf{t}$, respectively. Algorithm 8 is for the setting of prediction-aware with waste penalty.

6.3 Theoretical Analysis

In the case of having no access to the inputs of the future, we will prove a chain of inequalities:

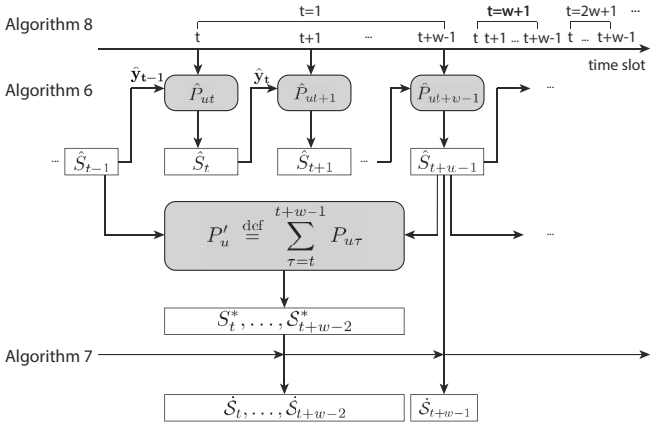

Fig. 3: Sequential execution of prediction-aware online algorithm

Algorithm 5: Prediction-Free Online Control Algorithm with Waste Penalty (PFOW)

for $t = 1, 2, 3, ...$
  With $\hat{\mathbf{y}}_{\mathbf{t-1}}$ as input, invoke **Algorithm 6** to solve $\hat{\mathbf{P}}_{\mathbf{ut}}$ to get the solution $\{\hat{\mathbf{x}}_{\mathbf{t}}, \hat{\mathbf{y}}_{\mathbf{t}}, \hat{\mathbf{z}}_{\mathbf{t}}, \hat{\mathbf{w}}_{\mathbf{t}}\}$;
  Invoke **Algorithm 7** to round $\{\hat{\mathbf{x}}_{\mathbf{t}}, \hat{\mathbf{y}}_{\mathbf{t}}\}$ to $\{\bar{\mathbf{x}}_{\mathbf{t}}, \bar{\mathbf{y}}_{\mathbf{t}}\}$;
  With $\bar{\mathbf{y}}_{\mathbf{t-1}}$ and $\{\bar{\mathbf{x}}_{\mathbf{t}}, \bar{\mathbf{y}}_{\mathbf{t}}\}$ as inputs, invoke **Algorithm 6** to solve $\hat{\mathbf{P}}_{\mathbf{ut}}$ to get the solution $\{\bar{\mathbf{x}}_{\mathbf{t}}, \bar{\mathbf{y}}_{\mathbf{t}}, \hat{\mathbf{z}}^*_{\mathbf{t}}, \hat{\mathbf{w}}^*_{\mathbf{t}}\}$;

Algorithm 6: Updated Regularized Fractional Algorithm, $\forall t$

Solve $\hat{\mathbf{P}}_{\mathbf{ut}}$ using any standard convex program solver:

$$\begin{aligned} \min \quad & \hat{P}_{ut} = \textstyle\sum_j d\sigma_{jt} x_{jt} + \sum_i \sum_j b_{it}\sigma_{jt} y_{ijt} + a w_t \\ & + \textstyle\sum_i \sum_j \frac{c_{ij}}{\delta}((y_{ijt} + \varepsilon) \ln \frac{y_{ijt}+\varepsilon}{y_{ijt-1}+\varepsilon} - y_{ijt}) \\ \text{s.t.} \quad & \text{(1a)} \sim \text{(1e)}, \text{ without } \text{“}\forall t\text{”}. \end{aligned}$$

Algorithm 7: Updated Randomized Rounding Algorithm, $\forall t$

Invoke **Algorithm 3** to round $\hat{\mathbf{y}}_{\mathbf{t}}$;
if $\sum_i \bar{y}_{ijt} = 1$
  Set $\bar{x}_{jt} = 0$;
else Set $\bar{x}_{jt} = \lambda_{jt}$;

Algorithm 8: Prediction-Aware Online Control Algorithm with Waste Penalty (PAOW)

for $t = 1, w+1, 2w+1, 3w+1, ...$
  for $\tau = t, t+1, ..., t+w-1$
    With $\hat{\mathbf{y}}_{\tau-1}$as input, use **Algorithm 6** to get $\hat{\mathcal{S}}_\tau$;
  With $\hat{\mathcal{S}}_{t-1}$ and $\hat{\mathcal{S}}_{t+w-1}$ as inputs, use any standard convex solver to minimize $P'_u \stackrel{\text{def}}{=} \sum_{\tau=t}^{t+w-1} P_{u\tau}$ and get the solutions $\{\mathcal{S}^*_t, ..., \mathcal{S}^*_{t+w-2}, \hat{\mathcal{S}}_{t+w-1}\}$;
  for $\tau = t, t+1, ..., t+w-2$
    Invoke **Algorithm 7** to round $\mathcal{S}^*_\tau$ into $\dot{\mathcal{S}}_\tau$;
  Invoke **Algorithm 7** to round $\hat{\mathcal{S}}_{t+w-1}$ into $\dot{\mathcal{S}}_{t+w-1}$;

$$\begin{aligned} & E(P_u(\{\bar{\mathbf{x}}_{\mathbf{t}}, \bar{\mathbf{y}}_{\mathbf{t}}, \hat{\mathbf{z}}^*_{\mathbf{t}}, \hat{\mathbf{w}}^*_{\mathbf{t}}, \forall t\})) && \text{(13a)} \\ \leqslant & r'_2 P_u(\{\hat{\mathbf{x}}_{\mathbf{t}}, \hat{\mathbf{y}}_{\mathbf{t}}, \hat{\mathbf{z}}_{\mathbf{t}}, \hat{\mathbf{w}}_{\mathbf{t}}, \forall t\}) && \text{(13b)} \\ \leqslant & r'_2 P''_u(\{\hat{\mathbf{x}}_{\mathbf{t}}, \hat{\mathbf{y}}_{\mathbf{t}}, \hat{\mathbf{z}}_{\mathbf{t}}, \hat{\mathbf{w}}_{\mathbf{t}}, \forall t\}) && \text{(13c)} \\ \leqslant & r'_1 r'_2 D_u && \text{(13d)} \\ \leqslant & r'_1 r'_2 P''_{uopt} && \text{(13e)} \\ \leqslant & (r'_1 r'_2 + M') P_{uopt}. && \text{(13f)} \end{aligned}$$

We will prove (13a)⩽(13b) based on the design of Algorithm 7. We have (13b)⩽(13c) due to the way we construct the auxiliary problem $\mathbf{P''_u}$ as shown next to facilitate our analysis, where its objective function P'' has an additional non-negative term of $\sum_t M(w_t + \sum_j z_{jt})$. We will prove (13c)⩽(13d) based on the design of Algorithm 6 and we derive the problem $\mathbf{D_u}$ with its objective function D_u, which is the Lagrange dual problem of $\mathbf{P''_u}$. We have (13d)⩽(13e) due to weak duality. We will finally prove (13e)⩽(13f) using the value of the term $\sum_t M(w_t + \sum_j z_{jt})$ at the optimum. To sum up, for our prediction-free approach, the competitive ratio is $r_1 r_2 + M$.

In the case of having access to the inputs in each prediction window as time goes, we will remove the proof here because it is similar to Section 5.4.

We prove (13c)⩽(13d) and find out r'_1.

Reformulating Problem $\mathbf{P'_u}$ into $\mathbf{P''_u}$. We transform $\mathbf{P'_u}$ into an equivalent form $\mathbf{P''_u}$ as follows:

$$\begin{aligned} \min \quad & P''_u = \textstyle\sum_t P'_{ut} \\ & = \textstyle\sum_t \sum_j d\sigma_{jt} x_{jt} + \sum_t \sum_i \sum_j b_{it}\sigma_{jt} y_{ijt} \\ & + \textstyle\sum_t \sum_i \sum_j c_{ij} u_{ijt} + \sum_t a w_t + \sum_t M(w_t + \sum_j z_{jt}) \\ \text{s.t.} \quad & \text{(1a), (1c)} \\ & \textstyle\sum_j x_{jt} - x_{jt} + \sum_i \sum_j y_{ijt} - \sum_i y_{ijt} \geqslant \sum_j \lambda_{jt} - 1, \\ & \qquad \forall j, \forall t, && \text{(14a)} \\ & \textstyle\frac{C_t}{\sigma_{jt}} z_{jt} + \sum_i y_{ijt} \geqslant 1, \qquad \forall j, \forall t, && \text{(14b)} \\ & w_t + \textstyle\sum_j z_{jt} \geqslant 1, \qquad \forall t, && \text{(14c)} \\ & u_{ijt} \geqslant y_{ijt} - y_{ijt-1}, \qquad \forall i, \forall j, \forall t, && \text{(14d)} \\ & u_{ijt} \geqslant 0, x_{jt} \geqslant 0, y_{ijt} \geqslant 0, z_{jt} \geqslant 0, w_t \geqslant 0, \\ & \qquad \forall i, \forall j, \forall t. && \text{(14e)} \end{aligned}$$

M is a large constant, based on which we have changed Constraint (14c) from its previous equality form into its current inequality form. Also, the new Constraint (14a) is from (1a) and (1b). Now, note that when $\mathbf{P''_u}$ reaches its offline optimum, $\mathbf{P'_u}$ also reaches its offline optimum.

Deriving the Lagrange Dual Problem. We write the dual problem $\mathbf{D_u}$ for the primal problem $\mathbf{P''_u}$, where α_{jt}, β_{jt}, γ_{jt}, μ_{jt}, ξ_t, τ_t, and φ_{ijt} are the dual variables.

$$\begin{aligned} \max \quad & D_u = \textstyle\sum_t \sum_j \lambda_{jt}\alpha_{jt} + \sum_j \sum_t (\sum_j \lambda_{jt} - 1)\beta_{jt} \\ & + \textstyle\sum_t \sum_j \mu_{it} + \sum_t \xi_t + \sum_t (\sum_t \tau_t - \tau_t) \\ \text{s.t.} \quad & \alpha_{jt} + \textstyle\sum_j \beta_{jt} - \beta_{jt} + \gamma_{jt} \leqslant a\sigma_{jt}, \quad \forall j, \forall t, && \text{(15a)} \\ & \alpha_{jt} + \textstyle\sum_j \beta_{jt} - \beta_{jt} + \mu_{jt} - \varphi_{ijt} + \varphi_{ijt+1} \leqslant b_{it}\sigma_{jt}, \\ & \qquad \forall j, \forall t, && \text{(15b)} \\ & -\gamma_{jt} + \textstyle\frac{C_t}{\sigma_{jt}}\mu_{jt} + \xi_t \leqslant M, \qquad \forall j, \forall t, && \text{(15c)} \\ & \varphi_{ijt} \leqslant c_{ij}, \qquad \forall i, \forall j, \forall t, && \text{(15d)} \\ & \xi_t \leqslant a + M, \qquad \forall t, && \text{(15e)} \\ & \text{dual variables} \geqslant 0 \end{aligned}$$

Bounding. Using the KKT conditions, we can prove

(13c)⩽(13d), i.e., Theorem 5:

Theorem 5. *$P''_u(\{\hat{\mathbf{x}}_\mathbf{t}, \hat{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}_\mathbf{t}, \hat{\mathbf{w}}_\mathbf{t}, \forall t\}) \leqslant r'_1 D_u(\{\Omega(\hat{\mathbf{x}}_\mathbf{t}, \hat{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}_\mathbf{t}, \hat{\mathbf{w}}_\mathbf{t}), \forall t\})$, where $r'_1 = 1 + (1 + |\mathcal{I}|\varepsilon)\delta$.*

Proof. See Appendix C. We need to exhibit that our online solution at each time slot, i.e., the optimal solution to $\hat{\mathbf{P}}_\mathbf{ut}$ defined in Algorithm 6, if collected over time and evaluated in the objective of $\mathbf{P''_u}$, makes (13c)⩽(13d) hold.

We prove (13a)⩽(13b) and find out r'_2. We show Theorem 6.

Theorem 6. *$E(P_u(\{\bar{\mathbf{x}}_\mathbf{t}, \bar{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}^*_\mathbf{t}, \hat{\mathbf{w}}^*_\mathbf{t}, \forall t\})) \leqslant r'_2 P_u(\{\hat{\mathbf{x}}_\mathbf{t}, \hat{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}_\mathbf{t}, \hat{\mathbf{w}}_\mathbf{t}, \forall t\})$, where $r'_2 = \Psi_x + \Psi_y$, $\Psi_x = (|\mathcal{J}|\Psi'_{xy} + a)\Psi''_{xy}$, $\Psi_y = |\mathcal{J}| \max_{i,j} c_{ij} \Psi''_{xy}$, $\Psi'_{xy} = \max_{j,t} d\sigma_{jt} + max_{i,j,t} b_{i,t}\sigma_{jt}$, and $\Psi''_{xy} = \max_{j,t} \frac{1}{d\sigma_{jt}} + \max_{i,j,t} \frac{1}{b_{it}\sigma_{jt}}$.*

Proof. We remove the proof as it is similar to the proof of Section 5.3.

7 Experimental Evaluation

7.1 Evaluation Settings

DDoS Traffic: We utilize the publicly availabl *CICDDoS2019* dataset [22] for our evaluation. The attack, which lasted for approximately 6 hours on March 3, 2019, utilized 4 PCs as external attackers to send incoming DDoS flows towards a single victim server and also used the victim server to send outgoing DDoS flows. The volumes of the flows are $0 \sim 4$ MB. For each flow, we have the starting time, duration, total volume, and flow direction indicator. The dataset includes approximately 73 million flows, with approximately 93% as incoming flows and around 7% as outgoing flows. We choose the first 1000-second data as input for our evaluations, which includes approximately 24 million flows. We consider 10 seconds as a single time slot.

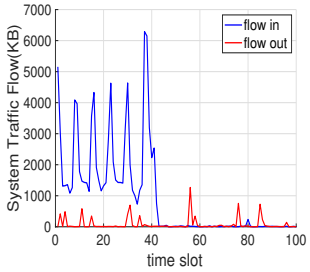

Fig. 4: DDoS flows

We then obtain our inputs σ_{jt} and λ_{jt} as follows. For every second s, we consider all the DDoS flows that start at s as a single combined DDoS flow that starts at the time slot where s belongs and ends at the time slot where the component flow of the longest duration ends. The volume of such a combined flow (i.e., σ_{jt}) at the time slot t is calculated as the sum of the volume of each component flow at t, where the volume of a component flow at any time slot in its duration is calculated as its recorded total volume divided by the number of its time slots. As a result, the volumes of the incoming DDoS flows are $0 \sim 5.8$ MB and the outgoing DDoS flows are $0 \sim 0.6$ MB. The existence of each flow (i.e., λ_{jt}) is set accordingly. Note that the outgoing flow volume at t is reflected in C_t. We depict the traffic volumes in the first 100 time slots in Fig. 4.

Cloud Scrubbing Centers: We use the pricing information from Amazon GuardDuty [23], which divides its regional pricing in the US into 6 regions. We therefore consider 6 scrubbing centers, one in each region. We assume a relatively stable b_{it} over time for each scrubbing center. At every time slot t, we use the scrubbing prices of \$4, \$4, \$4, \$4.2, \$4.4, and \$4.8 as the base, generate a uniformly-distributed random number in [0.8,1.2], and set b_{it} as the base times the corresponding random number.

Edge Scrubbing Facility: Without loss of generality, we set the local scrubbing capacity based on the maximum volume of the combined outgoing DDoS flow over the whole time horizon. Specifically, as the outgoing flows can only be scrubbed locally, the local scrubbing capacity must adequately cover the outgoing flows. We calculate the maximum virtual outgoing flow volume $v = 1267$ KB, and set the local scrubbing facility capacity as 1.5 times v. The available local scrubbing capacity C_t for t is set as the local scrubbing capacity minus the overall outgoing volume at t.

Operational Cost: We assume the local scrubbing facility is powered by the same electricity market as the cloud scrubbing centers. We use the average price of the US industrial electricity in 2019 (6.81 cents/kWh) as the operational cost of scrubbing one unit traffic locally (i.e., d) [24].

Redirection Rules Overhead: The BGP route convergence time can fluctuate from seconds to minutes in reality. Using this BGP convergence time as the base, we vary its weight to decide the unit switching cost c_{ij}. Analogously, we create a series of random numbers from the range of [5,100], and set the unit switching cost as the multiplication of the BGP convergence time and the random numbers.

Prediction Error: To evaluate the robustness of our prediction-aware algorithm, we inject zero-mean Gaussian white noise into b_{it} and σ_{jt}, while setting the standard deviation of such noise as a percentage of the standard deviation at t of b_{it} and σ_{jt}, respectively. This percentage is then considered as the prediction error. We vary the prediction error in [0, 10%].

Algorithms for Comparison: We implement multiple alternative approaches for comparison: *Random*, *Greedy*, *Fixed Horizon Control (FHC), and Approximation [41]*. The random approach treats every remote scrubbing center and the local facility equally, and randomly routes incoming flows to one of them; if the local capacity is not sufficient, it randomly chooses a remote one. The greedy approach calculates the optimal fractional solution at every time slot t while ignoring the switching cost, and then uses our own Algorithm 3 to get the final solution. In the prediction-aware setting, we additionally consider the FHC approach for comparison.

We write PFO to represent our the Prediction-Free Online Algorithm (i.e., Algorithm 1), and PAO to represent our Prediction-Aware Online Algorithm (i.e., Algorithm 4). We set the parameter $\varepsilon = 0.01$ in Algorithm 2. For our extended study of waste penalty, we have PFOW (i.e., Algorithm 5) and PAOW (i.e., Algorithm 8), correspondingly.

7.2 Evaluation Results

Fig. 5 illustrates the temporal evolution of the total cost as the weight assigned to the cost of transitioning between

scrubbing centers increases (it is worth noting that this weight was not included in our previous formulations). As the weight becomes more substantial, both the random and the greedy approach tend to disregard the switching cost, resulting in a significant increase in total cost; in contrast, our proposed online algorithm consistently yields superior performance, achieving a maximum savings of 28% in total cost through the explicit consideration of the switching cost.

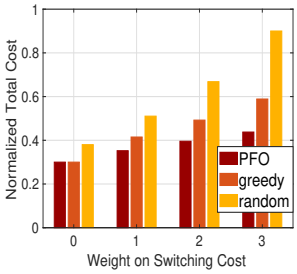

Fig. 5: Impact of switching cost

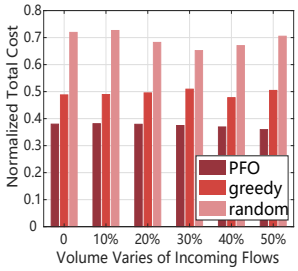

Fig. 6: Impact of incoming flows

Fig. 6 shows the total cost as the volumes of incoming flows vary. For instance, a value of approximately $\pm 30\%$ in the figure denotes that the volume of each incoming flow oscillates within a range of 0.7 to 1.3 times its initial volume, and the degree of fluctuation is generated through a uniform distribution. The figure illustrates that our method demonstrates robustness when confronted with varying volumes of incoming flow.

Fig. 7 examines the effect of the available local scrubbing capacity on the total cost. We determine the maximum flow volume v at each time slot, and set the local scrubbing facility capacity to 0 (i.e., utilizing a cloud-only approach),v, $2v$, and $3v$ respectively. As evident from the figure, as the available local capacity increases, our algorithm is able to select a local scrubbing facility and thus mitigate the need for additional switching costs. The random approach exhibits a variable total cost due to its inherent randomness, while the greedy approach exhibits a decreasing cost, however, it still remains higher than that of our approach.

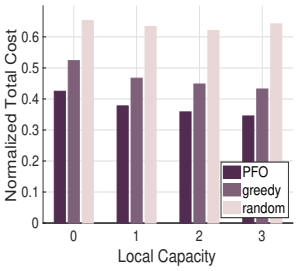

Fig. 7: Impact of local capacity

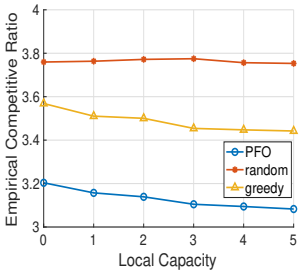

Fig. 8: Empirical competitiveness

Fig. 8 illustrates the empirical competitive ratios as a function of the local scrubbing capacity when no prediction is taken into account. The PFO algorithm consistently demonstrates superior performance, achieving an empirical competitive ratio of less than 3.2.

Fig. 9 visualizes the empirical competitive ratios as the length of the prediction window varies. With the exception of FHC and PAO, all other methods do not incorporate predictions and thus their competitiveness remains constant. When the prediction window length is 1, the performance of PFO and PAO is comparable. As the prediction window becomes more extensive, the competitive ratio of PAO decreases, providing a better performance than FHC by $16\% \sim 24\%$, owing to its ability to effectively manage the switching cost across prediction windows.

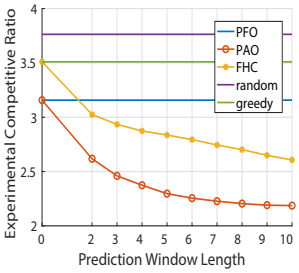

Fig. 9: Impact of window length

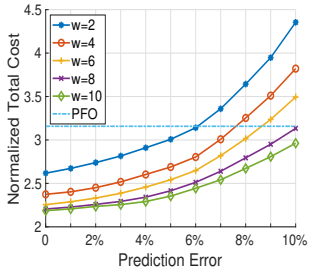

Fig. 10: Impact of Gaussian prediction error

Fig. 10 and 11 respectively investigate the impact of Gaussian prediction errors and uniform prediction errors on the performance of PAO, when the predictions about the remote scrubbing price and traffic flow volume in the prediction window are inaccurate. Using PFO as the baseline, which is not affected by predictions, we observe that the Gaussian prediction error threshold is 6% when the length of the prediction window is two. The threshold of uniform prediction error is 5% when the length of the prediction window is two. Furthermore, a longer prediction window can tolerate a higher prediction error threshold, thereby rendering PAO more resilient to prediction errors.

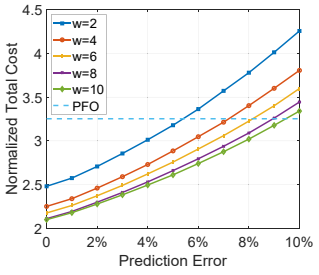

Fig. 11: Impact of uniform prediction error

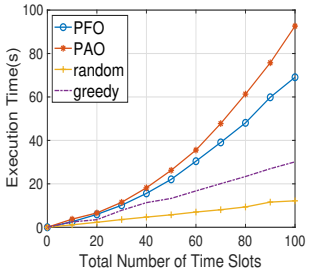

Fig. 12: Execution time

Fig. 12 compares the execution time of the algorithms as the input size increases. We execute all algorithms 10 times and calculate the average; for PAO, we set the prediction window to 5 time slots. Our algorithms run at a slightly slower pace than those with simpler algorithmic logic. The total execution time of our approach is less than 100 seconds, compared to the total duration of 1000 seconds for all the time slots; as a single time slot can be longer (e.g., hours) in reality, our execution time of the order of magnitude of seconds is deemed acceptable.

7.3 More Evaluation Results

Now we evaluate the performance of PFOW and PAOW. The settings here are the same as the settings in Section 7.1. To better illustrate the performance of our algorithm, we implement a new baseline. The algorithm Liu [41] was designed based on the primal-dual theory to address the request assignment problem. Both the remote scrubbing

center and the local facility are regarded as "cloudlets" in Liu. Liu selected a cloudlet for each suspicious flow to scrub and determine the allocated resources. One of the cloudlets was used to represent the local scrubbing facility with limited scrubbing capacity.

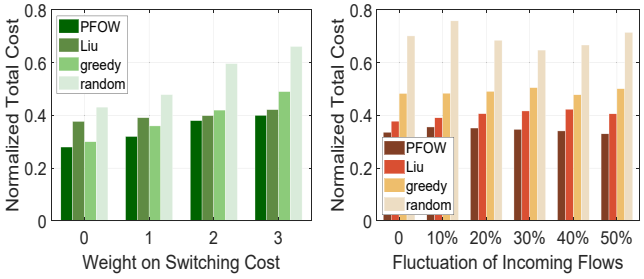

Fig. 13: Impact of switching cost

Fig. 14: Impact of incoming flows

Impact of Switching Cost: Fig. 13 compares the cumulative total cost as the weight of the switching cost increases. When the switching cost is not present, random selection can yield good results. However, when the weight of switching cost grows, the total cost of the random algorithm increases significantly and our approach performs optimally. Our algorithm consistently beats the greedy algorithm and the Liu algorithm.

Impact of Incoming Flows: We also examine the fluctuation of incoming flows, which can be represented by the variations in traffic volume of flow. We configure σ_{jt} to fluctuate between plus or minus 10% and 50%, and investigate the impact of this variation on the results. As shown in Fig. 14, our online algorithm demonstrates consistent performance for various incoming flows.

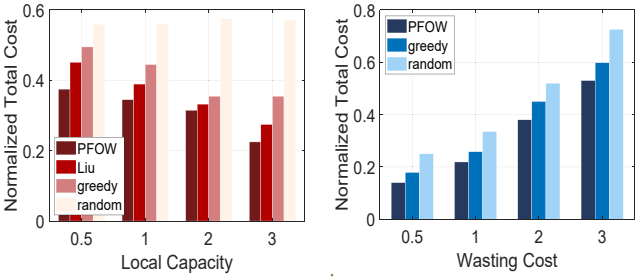

Fig. 15: Impact of local capacity

Fig. 16: Impact of unit waste cost

Impact of Local Capacity: Fig. 15 illustrates the total costs of the algorithms when the local scrubbing capacity fluctuates. We determine the maximum traffic flow volume v at each time slot and set the local capacity to $0.5, 2, 3$ times v. As the available local capacity increases, it selects local scrubbing, thereby avoiding additional switching costs.

Impact of Waste Penalty: In real situations, the waste penalty depends on how victims treat the importance of local scrubbing. We set an original waste penalty and then consider 0.5, 1, 2, and 3 times of it, respectively. As the weight of the waste increases, our online algorithm incurs up to 18% less total cost than other algorithms in Fig. 16.

Impact of Prediction Window: Fig. 17 examines the competitive ratio of different algorithms with varying prediction window lengths in the case where the input is predictable. With the exception of the PAOW and FHC algorithms, the

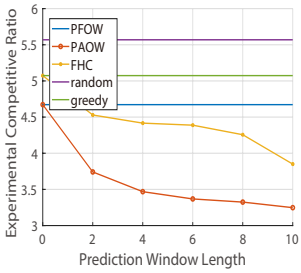

Fig. 17: Impact of window length

others are not influenced by predicted information and thus remain constant. When the window length of prediction is zero, PAOW will be equivalent to PFOW. As the prediction window length increases, the competitive ratio of PAOW decreases, and it approaches the offline optimum. PAOW consistently outperforms the standard FHC algorithm.

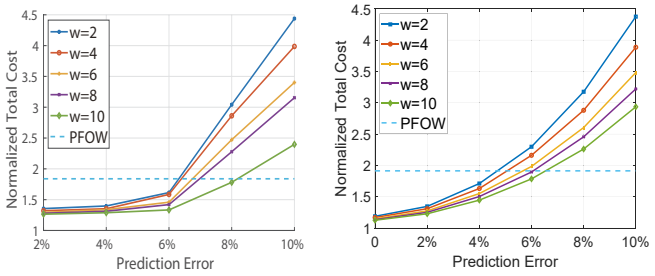

Fig. 18: Impact of Gaussian prediction error

Fig. 19: Impact of uniform prediction error

Impact of Inaccurate Predictions: Fig. 18 and Fig. 19 investigate the effect of Gaussian prediction errors and uniform prediction errors on PAOW, respectively. We vary the prediction error rate for different prediction window lengths. The results confirm that PAOW can operate with inaccurately predicted inputs, particularly when the Gaussian prediction error rates are less than $6\% \sim 8\%$ or the uniform prediction error rates are less than $4\% \sim 6\%$.

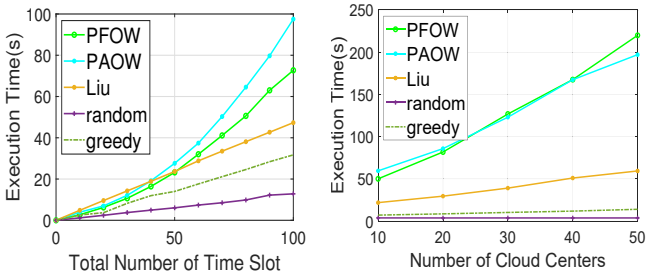

Fig. 20: Execution time

Execution Time: Fig. 20 shows the algorithms' running time as the data size and the number of cloud centers increase. We set the prediction window w as 5 and run all the algorithms 10 times, and then compute the average value. Our algorithms run at a slower pace than others; however, when taking into account the total cost in Fig. 13, such execution time is deemed relatively acceptable. It is important to note that the execution time is measured on our commodity laptop, which is equipped with 12 i7-8700 CPU cores, 64GB RAM and 1000GB HDDs; executing the algorithms in real-world data centers can be much faster.

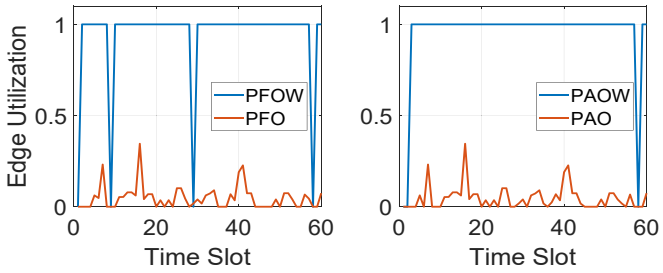

Fig. 21: Edge Utilization

Edge Utilization: Fig. 21 shows the resource utilization of the local edge scrubbing facility. At each time slot, the scheduling decisions obtained by the algorithm considering the waste penalty (PFOW and PAOW) preferentially use the local facility for scrubbing.

8 Discussions on Security Analysis

While the scope of our work is the scrubbing performance in terms of the total cost, including the operational cost, the switching cost, and the penalty of resource waste, we discuss the security analysis as below.

What type of DDoS attacks and what type of scrubbing technologies are targeted in this paper? We assume that the DDoS traffic under consideration can be scrubbed by the corresponding scrubbing technology under consideration, as long as such traffic can be redirected to scrubbing centers and scrubbing such traffic incurs "cost", which is often the case. That is, we do not focus on a specific type of DDoS traffic (e.g., transport layer, application layer), or a specific type of scrubbing technologies (e.g., deep packet inspection, machine-learning-based); instead, we build general models in this paper to capture the different types of costs, and present a mathematical study of online cost optimization.

Who, as the controller, runs the proposed algorithms, and can this controller be compromised? The scrubbing system mainly consists of the remote cloud scrubbing centers and the local edge scrubbing center. The operator or the provider of such a scrubbing system can be responsible for gathering the inputs and running the online optimization algorithms to redirect the traffic, scrub the traffic, etc. In this paper, we focus on letting the scrubbing system protect the enterprise victim, and do not consider the situation where the scrubbing system itself is compromised; we also focus on DDoS, and do not consider other types of attacks that may or may not be launched simultaneously.

Can the attacker attack the scrubbing system itself by DDoS? A typical assumption for cloud computing is that the cloud has almost infinite resources, thus capable of scrubbing the DDoS traffic no matter how large such traffic is. So, we do not consider the case where the attackers directly attack the cloud scrubbing centers using DDoS. Regarding the local edge scrubbing facility with limited capacity, it seems that the attackers can overwhelm it by DDoS and prevent it from scrubbing the traffic that targets the victim. Yet, note that such local edge scrubbing facility is co-located with the victim, which could be invisible to the attacker; even if the attacker detects it, it is the scrubbing system operator's responsibility to firstly protect the local edge scrubbing facility and make it function normally to protect the victim.

Can the attacker circumvent the scrubbing system to attack the victim? We assume the suspicious traffic is already detected in the first place, and then the suspicious traffic is scrubbed so that the malicious part is filtered out. We do not consider the case where the suspicious traffic cannot be detected and thus one has no idea whether or when to start traffic scrubbing. The context of this paper is that traffic scrubbing is started, and our task is to carefully schedule and manage it in a cost-efficient manner.

9 Concluding Remarks

In this paper, we investigate the optimization problem of scheduling DDoS traffic through both remote cloud and local edge scrubbing facilities. We design prediction-free and prediction-aware online algorithms, based on the availability of predicted inputs, and demonstrate that our online algorithms can achieve guaranteed performance compared to offline optimums in terms of the total cost over time. We also demonstrate the effectiveness and the advantages of our approaches in practice. For future work, we plan to formally incorporate various inaccurate prediction models into our algorithm design and analysis, and also investigate other non-scrubbing-based cloud-edge DDoS mitigation approaches in general.

Acknowledgement

The authors acknowledge Dr. Yebo Feng (Nanyang Technological University, Singapore) for helpful discussions on security analysis and the anonymous reviewers for their constructive comments.

Appendix A
Proof of Theorem 1

First, we write the Karush-Kuhn-Tucker (KKT) conditions that are satisfied by the optimal solution of $\hat{\mathbf{P}}_\mathbf{t}$:

$$M + dC_t - \alpha_{jt} - \textstyle\sum_j \beta_{jt} + \beta_{jt} - \frac{C_t}{\sigma_{jt}}\mu_{jt} - \xi_t = 0,\ \forall j, \tag{16a}$$

$$b_{it}\sigma_{jt} - \alpha_{jt} - \textstyle\sum_j \beta_{jt} + \beta_{jt} - \mu_{jt} + \frac{c_{ij}}{\delta}\ln\frac{y_{ijt}+\varepsilon}{y_{ijt-1}+\varepsilon} = 0,\ \forall i, \forall j, \tag{16b}$$

$$M - \xi_t = 0,\ \forall j, \tag{16c}$$

$$\alpha_{jt}(\lambda_{jt} - z_{jt} - \textstyle\sum_i y_{ijt}) = 0,\ \forall j, \tag{16d}$$

$$\beta_{jt}(\textstyle\sum_j \lambda_{jt} - 1 - \sum_j z_{jt} + z_{jt} - \sum_i \sum_j y_{ijt} + \sum_i y_{ijt}) = 0,\ \forall j, \tag{16e}$$

$$\mu_{jt}(1 - \frac{C_t}{\sigma_{jt}} z_{jt} - \textstyle\sum_i y_{ijt}) = 0,\ \forall j, \tag{16f}$$

$$\xi_t(1 - Q_t - \textstyle\sum_j z_{jt}) = 0,\ \forall j. \tag{16g}$$

Second, to show (3c)$\leqslant$(3d), we need a dual solution to be evaluated in D, as in (3d). We can actually always construct such a dual solution via a designated mapping Ω, which converts $\hat{\mathbf{P}}_\mathbf{t}$'s optimal solution $\{\hat{\mathbf{y}}_\mathbf{t}, \hat{\mathbf{z}}_\mathbf{t}, \forall t\}$ into a feasible dual solution. We can leverage our KKT conditions above to easily show that the constructed dual solution via the following mapping satisfies all the constrains of $\mathbf{D}$:

$$\alpha_{jt} = \alpha_j, \forall j;\ \beta_{jt} = \beta_j, \forall j;\ \gamma_{jt} = \gamma_j, \forall j;\ \mu_{jt} = \mu_j, \forall j;$$
$$\xi_t = \xi;\ \varphi_{ijt} = \frac{c_{ij}}{\delta}\ln\frac{y_{ijt}+\varepsilon}{y_{ijt-1}+\varepsilon}, \forall i, \forall j.$$

Taking Constraint (7b) for example. The left-hand side of (7b) is equal to the right-hand side, which is based on (16b).

$$\begin{aligned}&\alpha_{jt}+\textstyle\sum_j\beta_{jt}-\beta_{jt}+\mu_{jt}-\varphi_{ijt}+\varphi_{ijt+1}\\=&\alpha_j+\textstyle\sum_j\beta_j-\beta_j+\mu_j-\frac{c_{ij}}{\delta}\ln\frac{y_{ijt}+\varepsilon}{y_{ijt-1}+\varepsilon}+\frac{c_{ij}}{\delta}\ln\frac{y_{ijt+1}+\varepsilon}{y_{ijt}+\varepsilon}\\=&b_{it}\sigma_{jt}.\end{aligned}$$

Third, let us now prove (3c)$\leqslant$(3d). With the two facts below, $\forall p,q>0$,

$$(\textstyle\sum_n p_n)\ln\frac{\sum_n p_n}{\sum_n q_n}\leqslant\sum_n p_n\ln\frac{p_n}{q_n},\tag{18a}$$
$$p-q\leqslant p\ln\frac{p}{q},\tag{18b}$$

we can have the following, $\forall i,\forall j$:

$$\begin{aligned}&\textstyle\sum_t\hat{y}_{ijt}\ln\frac{\hat{y}_{ijt}+\varepsilon}{\hat{y}_{ijt-1}+\varepsilon}\\=&\textstyle\sum_t(\hat{y}_{ijt}+\varepsilon)\ln\frac{\hat{y}_{ijt}+\varepsilon}{\hat{y}_{ijt-1}+\varepsilon}-\sum_t\varepsilon\ln\frac{\hat{y}_{ijt}+\varepsilon}{\hat{y}_{ijt-1}+\varepsilon}\\\geqslant&\textstyle(\sum_t(\hat{y}_{ijt}+\varepsilon))\ln\frac{\sum_t(\hat{y}_{ijt}+\varepsilon)}{\sum_t(\hat{y}_{ijt-1}+\varepsilon)}+(\hat{y}_{ij0}+\varepsilon)\ln\frac{\hat{y}_{ij0}+\varepsilon}{\hat{y}_{ijT}+\varepsilon}\\\geqslant&\textstyle\sum_t(\hat{y}_{ijt}+\varepsilon)-\sum_t(\hat{y}_{ijt-1}+\varepsilon)+\hat{y}_{ij0}-\hat{y}_{ijT}=0.\end{aligned}$$

Then we can prove that the non-switching cost in P'' satisfies $\sum_t\sum_i\sum_j b_{it}\sigma_{jt}\hat{y}_{ijt}+\sum_t\sum_j(M+dC_t)\hat{z}_{jt}+\sum_t MQ_t\leqslant 2D$:

$$\begin{aligned}&\textstyle\sum_t\sum_i\sum_j b_{it}\sigma_{jt}\hat{y}_{ijt}+\sum_t\sum_j(M+dC_t)z_{jt}+\sum_t MQ_t\\\leqslant&\textstyle\sum_t\sum_j(\alpha_{jt}+\sum_j\beta_{jt}-\beta_{jt}+\frac{C_t}{\sigma_{jt}}\mu_{jt}+\xi_t)\hat{z}_{jt}+\sum_t\xi_tQ_t\\&\textstyle+\sum_t\sum_i\sum_j(\alpha_{jt}+\sum_j\beta_{jt}-\beta_{jt}+\mu_{jt}-\frac{c_{ij}}{\delta}\ln\frac{\hat{y}_{ijt}+\varepsilon}{\hat{y}_{ijt-1}+\varepsilon})\hat{y}_{ijt}\qquad(20a)\\\leqslant&\textstyle\sum_t\sum_j(\alpha_{jt}+\sum_j\beta_{jt}-\beta_{jt}+\frac{C_t}{\sigma_{jt}}\mu_{jt}+\xi_t)\hat{z}_{jt}+\sum_t\xi_tQ_t\\&\textstyle+\sum_t\sum_i\sum_j(\alpha_{jt}+\sum_j\beta_{jt}-\beta_{jt}+\mu_{jt})\hat{y}_{ijt}\qquad(20b)\\=&\textstyle\sum_t\sum_j\alpha_{jt}\lambda_{jt}+\sum_t\sum_j\beta_{jt}(\sum_j\lambda_{jt}-1)+\sum_t\sum_j\mu_{jt}\\&\textstyle+\sum_t\xi_t\qquad(20c)\\=&D+\textstyle\sum_t Q_t\xi_t\leqslant 2D.\end{aligned}$$

(20a) is due to (16a)$\sim$(16c). (20b) is due to $\hat{y}_{ijt}\ln\frac{\hat{y}_{ijt}+\varepsilon}{\hat{y}_{ijt-1}+\varepsilon}\geqslant 0$, as shown right above. (20c) follows from (16d)$\sim$(16g). As for the switching cost, it can then be upper-bounded as $\sum_t\sum_i\sum_j c_{ij}(\hat{y}_{ijt}-\hat{y}_{ijt-1})^+\leqslant(1+|\mathcal{I}|\varepsilon)\delta D$, where we define $\eta=(1+\varepsilon)\sigma$. That is,

$$\begin{aligned}&\textstyle\sum_t\sum_i\sum_j c_{ij}(\hat{y}_{ijt}-\hat{y}_{ijt-1})^+\\\leqslant&\textstyle\sum_t\sum_i\sum_j c_{ij}(\hat{y}_{ijt}-\hat{y}_{ijt-1})\qquad(21a)\\\leqslant&\textstyle\sum_t\sum_i\sum_j c_{ij}(\hat{y}_{ijt}+\varepsilon)\ln\frac{(\hat{y}_{ijt}+\varepsilon)}{(\hat{y}_{ijt-1}+\varepsilon)}\qquad(21b)\\=&\textstyle\delta\sum_t\sum_i\sum_j(\hat{y}_{ijt}+\varepsilon)(\alpha_{jt}+\sum_j\beta_{jt}-\beta_{jt}+\mu_{jt}-b_{it}\sigma_{jt})\qquad(21c)\\\leqslant&\textstyle(1+|\mathcal{I}|\varepsilon)\delta\sum_t\sum_j(\alpha_{jt}+\sum_j\beta_{jt}-\beta_{jt}+\mu_{jt})\qquad(21d)\\\leqslant&\textstyle(1+|\mathcal{I}|\varepsilon)\delta\big(\sum_t\sum_j\lambda_{jt}\alpha_{jt}+\sum_t\sum_j(\sum_j\lambda_{jt}-1)\beta_{jt}\\&\textstyle+\sum_t\sum_j\mu_{jt}\big)\qquad(21e)\\\leqslant&(1+|\mathcal{I}|\varepsilon)\delta D.\end{aligned}$$

(21a) is owing to the definition of $(\hat{y}_{ijt}-\hat{y}_{ijt-1})^+$. (21b) is due to (18b). (21c) follows from (16b). (21d) holds because of (1b). (21e) is due to our assumption of $\sum_j\lambda_{jt}\geqslant 1$.

Appendix B
Proof of Theorem 2

We firstly show $E\big((\sum_t\sum_i\sum_j b_{it}\sigma_{jt}\bar{y}_{ijt}+\sum_t\sum_j dC_tz_{jt}),\forall t\big)\leqslant\Psi_yP\left(\{\hat{\mathbf{y}}_\mathbf{t},\hat{\mathbf{z}}_\mathbf{t},\forall t\}\right)$:

$$\begin{aligned}&E(\textstyle\sum_t\sum_i\sum_j b_{it}\sigma_{jt}\bar{y}_{ijt}+\sum_t\sum_j dC_tz_{jt})\\\leqslant&\Psi'_{yz}\textstyle\sum_t\sum_j E\left(z_{jt}+\sum_i\bar{y}_{ijt}\right)\qquad(22a)\\\leqslant&\Psi'_{yz}\textstyle\sum_t\sum_j 1\qquad(22b)\\\leqslant&|\mathcal{J}|\Psi'_{yz}\textstyle\sum_t\sum_j\lambda_{jt}\qquad(22c)\\\leqslant&|\mathcal{J}|\Psi'_{yz}\textstyle\sum_t\sum_j(z_{jt}+\sum_i\hat{y}_{ijt})\qquad(22d)\\\leqslant&\Psi_y(\textstyle\sum_t\sum_i\sum_j b_{it}\sigma_{jt}y_{ijt}+\sum_t\sum_j dC_tz_{jt}).\qquad(22e)\end{aligned}$$

After executing Algorithm 3, the sum of $z_{jt}+\sum_i\bar{y}_{ijt}$ is less than or equal to 1, hence we can reach (22b). (22c) holds since we assume $\sum_j\lambda_{jt}\geqslant 1$. (22d) is due to (1a).

Then we show $E\Big(\big(\sum_t\sum_i\sum_j c_{ij}(\bar{y}_{ijt}-\bar{y}_{ijt-1})^+\big),\forall t\Big)\leqslant\Psi_z\hat{P}\left(\{\hat{\mathbf{y}}_\mathbf{t},\hat{\mathbf{z}}_\mathbf{t},\forall t\}\right)$:

$$\begin{aligned}&E\big(\textstyle\sum_t\sum_i\sum_j c_{ij}(\bar{y}_{ijt}-\bar{y}_{ijt-1})^+\big)\\\leqslant&E\big(\textstyle\sum_t\sum_i\sum_j c_{ij}\bar{y}_{ijt}\big)\qquad(23a)\\\leqslant&\max_{i,j}c_{ij}E(\textstyle\sum_t\sum_i\sum_j\bar{y}_{ijt})\qquad(23b)\\\leqslant&\Psi_z(\textstyle\sum_t\sum_i\sum_j b_{it}\sigma_{jt}y_{ijt}+\sum_t\sum_j dC_tz_{jt}).\qquad(23c)\end{aligned}$$

(23a) follows from $(\cdot)^+\overset{\text{def}}{=}\max\{\cdot,0\}$. (23b)$\sim$(23c) is analogous to (22a)$\sim$(22e).

Appendix C
Proof of Theorem 5

First, we write the Karush-Kuhn-Tucker (KKT) conditions that are satisfied by the optimal solution of $\hat{\mathbf{P}}_\mathbf{t}$:

$$\alpha_{jt}\sigma_{jt}-\alpha_{jt}-\textstyle\sum_j\beta_{jt}+\beta_{jt}-\gamma_{jt}=0,\ \forall j,\tag{24a}$$
$$b_{it}\sigma_{jt}-\alpha_{jt}-\textstyle\sum_j\beta_{jt}+\beta_{jt}-\mu_{jt}+\frac{c_{ij}}{\delta}\ln\frac{y_{ijt}+\varepsilon}{y_{ijt-1}+\varepsilon}=0,\ \forall i,\forall j,\tag{24b}$$
$$M+\gamma_{jt}-\tfrac{C_t}{\delta_{jt}}\mu_{jt}-\xi_t=0,\ \forall j,\tag{24c}$$
$$M+a-\xi_t=0,\tag{24d}$$
$$\alpha_{jt}(\lambda_{jt}-x_{jt}-\textstyle\sum_i y_{ijt})=0,\ \forall j,\tag{24e}$$
$$\beta_{jt}(\textstyle\sum_j\lambda_{jt}-1-\sum_j x_{jt}+x_{jt}-\sum_i\sum_j y_{ijt}+\sum_i y_{ijt})=0,\ \forall j,\tag{24f}$$
$$\gamma_{jt}(z_{jt}-x_{ijt})=0,\ \forall j,\tag{24g}$$
$$\mu_{jt}(1-\tfrac{C_t}{\sigma_{jt}}z_{jt}-\textstyle\sum_i y_{ijt})=0,\ \forall j,\tag{24h}$$
$$\xi_t(1-w_t-\textstyle\sum_j z_{jt})=0,\ \forall j,\tag{24i}$$

Second, to show (13c)$\leqslant$(13d), we need a dual solution to be evaluated in D_u, as in (13d). Now, we show we can always construct such a dual solution via a mapping Ω, which converts $\hat{\mathbf{P}}_\mathbf{ut}$'s optimal solution $\{\hat{\mathbf{x}}_\mathbf{t},\hat{\mathbf{y}}_\mathbf{t},\hat{\mathbf{z}}_\mathbf{t},\hat{\mathbf{w}}_\mathbf{t},\forall t\}$ into a feasible dual solution. We can leverage our KKT conditions above to easily show that the following constructed dual solution satisfies all the constrains of $\mathbf{D_u}$:

$$\alpha_{jt}=\alpha_j,\forall j;\ \beta_{jt}=\beta_j,\forall j;\ \gamma_{jt}=\gamma_j,\forall j;\ \mu_{jt}=\mu_j,\forall j;\ \xi_t=\xi;\ \varphi_{ijt}=\tfrac{c_{ij}}{\delta}\ln\tfrac{y_{ijt}+\varepsilon}{y_{ijt-1}+\varepsilon},\forall i,\forall j.$$

Third, we can have the following, which has been proved in Appendix A, $\forall i,\forall j$: $\hat{y}_{ijt}\ln\frac{\hat{y}_{ijt}+\varepsilon}{\hat{y}_{ijt-1}+\varepsilon}\geqslant 0$.

Then, we can prove that the non-switching cost in P''_u satisfies: $\sum_t\sum_j d\sigma_{jt}\hat{x}_{jt}+\sum_t\sum_i\sum_j b_{it}\sigma_{jt}\hat{y}_{ijt}+\sum_t a\hat{w}_t+M(\hat{w}_t+\sum_j\hat{z}_{jt})\leqslant D_u$.

$$\textstyle\sum_t\sum_j d\sigma_{jt}\hat{x}_{jt}+\sum_t\sum_i\sum_j b_{it}\sigma_{jt}\hat{y}_{ijt}+\sum_t a\hat{w}_t$$

$$
\begin{aligned}
&+M(\hat{w}_t+\textstyle\sum_j \hat{z}_{jt})\\
&\leqslant\textstyle\sum_t\sum_j(\alpha_{jt}+\sum_j\beta_{jt}-\beta_{jt}+\gamma_{jt})\hat{x}_{jt}\\
&+\textstyle\sum_t\sum_i\sum_j(\alpha_{jt}+\sum_j\beta_{jt}-\beta_{jt}+\mu_{jt}-\frac{c_{ij}}{\delta}\ln\frac{\hat{y}_{ijt}+\varepsilon}{\hat{y}_{ijt-1}+\varepsilon})\hat{y}_{ijt}\\
&+\textstyle\sum_t\xi_t\hat{w}_t+\sum_t\sum_j(\frac{C_t}{\delta_{jt}}\mu_{jt}+\xi_t-\gamma_{jt})\hat{z}_{jt} &\text{(25a)}\\
&\leqslant\textstyle\sum_t\sum_j(\alpha_{jt}+\sum_j\beta_{jt}-\beta_{jt}+\gamma_{jt})\hat{x}_{jt}\\
&+\textstyle\sum_t\sum_i\sum_j(\alpha_{jt}+\sum_j\beta_{jt}-\beta_{jt}+\mu_{jt})\hat{y}_{ijt}\\
&+\textstyle\sum_t\xi_t\hat{w}_t+\sum_t\sum_j(\frac{C_t}{\delta_{jt}}\mu_{jt}+\xi_t-\gamma_{jt})\hat{z}_{jt} &\text{(25b)}\\
&=\textstyle\sum_t\sum_j\alpha_{jt}\lambda_{jt}+\sum_t\sum_j\beta_{jt}(\sum_j\lambda_{jt}-1)+\sum_t\sum_j\mu_{jt}\\
&+\textstyle\sum_t\xi_t &\text{(25c)}\\
&=D_u &\text{(25d)}
\end{aligned}
$$

(25a) is due to (24a)∼(24d). (25b) holds since $\hat{y}_{ijt}\ln\frac{\hat{y}_{ijt}+\varepsilon}{\hat{y}_{ijt-1}+\varepsilon}\geqslant 0$, as shown before. (25c) follows from (24e)∼(24i).

Then the switching cost in P_u can be upper-bounded as follows: $\sum_t\sum_i\sum_j c_{ij}(\hat{y}_{ijt}-\hat{y}_{ijt-1})^+\leqslant(1+|\mathcal{I}|\varepsilon)\delta D_u$. Here, we will omit its proof as it has been proofed in Appendix A.

REFERENCES

[1] Y. Zhong, L. Jiao, R. Zhou, and L. Song, "On-demand or on-premises: Online mitigation of ddos attacks via cloud-edge coordination," in *Proc. of IEEE SECON*, 2022.

[2] P. Zilberman, R. Puzis, and Y. Elovici, "On network footprint of traffic inspection and filtering at global scrubbing centers," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 5, pp. 521–534, 2015.

[3] "Cloudflare - The Web Performance & Security Company," https://www.cloudflare.com/.

[4] K. Bhardwaj, J. C. Miranda, and A. Gavrilovska, "Towards iot-ddos prevention using edge computing," in *Proc. of USENIX HotEdge*, 2018.

[5] S. Myneni, A. Chowdhary, D. Huang, and A. Alshamrani, "Smart-defense: A distributed deep defense against ddos attacks with edge computing," *Elsevier Computer Networks*, vol. 209, p. 108874, 2022.

[6] V. Giotsas, G. Smaragdakis, C. Dietzel, P. Richter, A. Feldmann, and A. Berger, "Inferring bgp blackholing activity in the internet," in *Proc. of ACM IMC*, 2017.

[7] D. Kwon, H. Kim, D. An, and H. Ju, "Ddos attack volume forecasting using a statistical approach," in *Proc. of IFIP/IEEE IM*, 2017.

[8] W. You, L. Jiao, J. Li, and R. Zhou, "Scheduling ddos cloud scrubbing in isp networks via randomized online auctions," in *Proc. of IEEE INFOCOM*, 2020.

[9] L. Zhou, H. Guo, and G. Deng, "A fog computing based approach to ddos mitigation in iiot systems," *Elsevier Computers & Security*, vol. 85, pp. 51–62, 2019.

[10] T. Liu, P. Li, and Y. Gu, "Glint: Decentralized federated graph learning with traffic throttling and flow scheduling," in *Proc. of IEEE/ACM IWQoS*, 2021.

[11] Q. Li, X. Deng, Z. Liu, Y. Yang, X. Zou, Q. Wang, M. Xu, and J. Wu, "Dynamic network security function enforcement via joint flow and function scheduling," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 486–499, 2022.

[12] X. Lan, Y. Chen, and L. Cai, "Throughput-optimal h-qmw scheduling for hybrid wireless networks with persistent and dynamic flows," *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 1182–1195, 2019.

[13] L. Gu, D. Zeng, S. Tao, S. Guo, H. Jin, A. Y. Zomaya, and W. Zhuang, "Fairness-aware dynamic rate control and flow scheduling for network utility maximization in network service chain," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1059–1071, 2019.

[14] A. Gushchin, S.-H. Tseng, and A. Tang, "Optimization-based network flow deadline scheduling," in *Proc. of IEEE ICNP*, 2016.

[15] T. Ouyang, X. Chen, Z. Zhou, L. Li, and X. Tan, "Adaptive user-managed service placement for mobile edge computing via contextual multi-armed bandit learning," *IEEE Transactions on Mobile Computing*, 2021.

[16] 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. of IEEE INFOCOM*, 2019.

[17] L. Wang, L. Jiao, J. Li, J. Gedeon, and M. Mühlhäuser, "Moera: Mobility-agnostic online resource allocation for edge computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1843–1856, 2018.

[18] S. Krishnasamy, P. Akhil, A. Arapostathis, R. Sundaresan, and S. Shakkottai, "Augmenting max-weight with explicit learning for wireless scheduling with switching costs," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2501–2514, 2018.

[19] J. Chen, Y. Xu, Q. Wu, Y. Zhang, X. Chen, and N. Qi, "Interference-aware online distributed channel selection for multicluster fanet: A potential game approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3792–3804, 2019.

[20] N. Buchbinder, S. Chen, and J. Naor, "Competitive analysis via regularization," in *Proc. of SODA*, 2014.

[21] A. A. Ageev and M. I. Sviridenko, "Pipage rounding: A new method of constructing algorithms with proven performance guarantee," *Journal of Combinatorial Optimization*, vol. 8, no. 3, pp. 307–328, 2004.

[22] "DDoS Evaluation Dataset," https://www.unb.ca/cic/datasets/ddos-2019.html.

[23] "Amazon GuardDuty Price Reduction," https://aws.amazon.com/cn/guardduty/pricing/.

[24] "Total electric power industry summary statistics," https://www.eia.gov/electricity/annual/html/epa_01_01.html.

[25] Q. He, C. Wang, G. Cui, B. Li, R. Zhou, Q. Zhou, Y. Xiang, H. Jin, and Y. Yang, "A game-theoretical approach for mitigating edge ddos attack," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2333–2348, 2022.

[26] D. IR and S. K, "Dad: Domain adversarial defense system against ddos attacks in cloud," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 554–568, 2022.

[27] A. Kumar and G. Somani, "Ddos attack mitigation in cloud targets using scale-inside out assisted container separation," in *Proc. of IEEE INFOCOM*, 2022.

[28] Y. Deng, H. Jiang, P. Cai, T. Wu, P. Zhou, B. Li, H. Lu, J. Wu, X. Chen, and K. Wang, "Resource provisioning for mitigating edge ddos attacks in mec-enabled sdvn," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24 264–24 280, 2022.

[29] C. Tsanikidis and J. Ghaderi, "Online scheduling and routing with end-to-end deadline constraints in multihop wireless networks," in *Proc. of ACM MobiHoc*, 2022.

[30] Y. Mao, X. Shang, and Y. Yang, "Provably efficient algorithms for traffic-sensitive sfc placement and flow routing," in *Proc. of IEEE INFOCOM*, 2022.

[31] W. Z. J. R. Kai Gong, Dong Yang, "An efficient scheduling approach for multi-level industrial chain flows in time-sensitive networking," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 221, no. C, 2023.

[32] W.-K. Chung, Y. Li, C.-H. Ke, S.-Y. Hsieh, A. Y. Zomaya, and R. Buyya, "Dynamic parallel flow algorithms with centralized scheduling for load balancing in cloud data center networks," *IEEE Transactions on Cloud Computing*, vol. 11, no. 1, pp. 1050–1064, 2023.

[33] T. Liu, L. Fang, Y. Zhu, W. Tong, and Y. Yang, "A near-optimal approach for online task offloading and resource allocation in edge-cloud orchestrated computing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 8, pp. 2687–2700, 2022.

[34] W. Fan, L. Zhao, X. Liu, Y. Su, S. Li, F. Wu, and Y. Liu, "Collaborative service placement, task scheduling, and resource allocation for task offloading with edge-cloud cooperation," *IEEE Transactions on Mobile Computing*, pp. 1–18, 2022.

[35] V. Farhadi, F. Mehmeti, T. He, T. F. L. Porta, H. Khamfroush, S. Wang, K. S. Chan, and K. Poularakis, "Service placement and request scheduling for data-intensive applications in edge clouds," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 779–792, 2021.

[36] W. Wang, M. Tornatore, Y. Zhao, H. Chen, Y. Li, A. Gupta, J. Zhang, and B. Mukherjee, "Infrastructure-efficient virtual-machine placement and workload assignment in cooperative edge-cloud computing over backhaul networks," *IEEE Transactions on Cloud Computing*, vol. 11, no. 1, pp. 653–665, 2023.

[37] R. Alsurdeh, R. N. Calheiros, K. M. Matawie, and B. Javadi, "Hybrid workflow provisioning and scheduling on cooperative edge cloud computing," in *2021 IEEE/ACM 21st International Symposium*

on Cluster, Cloud and Internet Computing (CCGrid), 2021, pp. 445–454.
[38] Y. Li, G. Qu, and N. Li, "Online optimization with predictions and switching costs: Fast algorithms and the fundamental limit," *IEEE Transactions on Automatic Control*, vol. 66, no. 10, pp. 4761–4768, 2021.
[39] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang, and X. Fu, "Delay-aware virtual network function placement and routing in edge clouds," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 445–459, 2021.
[40] Y. Ren, S. Shen, Y. Ju, X. Wang, W. Wang, and V. C. Leung, "Edge-matrix: A resources redefined edge-cloud system for prioritized services," in *Proc. of IEEE INFOCOM*, 2022.
[41] 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, 2023.

Ruiting Zhou is a Professor in the School of Computer Science Engineering at Southeast University. She received her Ph.D. degree in 2018 from the Department of Computer Science, University of Calgary, Canada. 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, IEEE/ACM TON, IEEE JSAC, IEEE TMC. She serves as the TPC chair for INFOCOM workshop-ICCN 2019-2023. She also serves as a reviewer for international conferences and journals such us IEEE ICDCS, IEEE/ACM IWQoS, IEEE SECON, IEEE JSAC, IEEE TON, IEEE TMC, IEEE TCC.

Yifan Zeng received the B.E. degree from the School of Cyber Science and Engineering, Wuhan University, China, in 2022. She is currently working toward the master's degree from the School of Cyber Science and Engineering, Wuhan University, China. Her research interests include edge computing, online learning and network optimization.

Lei Jiao received the Ph.D. degree in computer science from the University of Göttingen, Germany. He is with the Department of Computer Science, University of Oregon, USA and was previously with Nokia Bell Labs, Ireland. He is interested in the mathematics of optimization, control, learning, and economics applied to large-scale computer systems, services, and applications. He is an NSF CAREER awardee and also a recipient of several IEEE Best Paper Awards. He publishes papers in journals such as JSAC, ToN, TPDS, and TMC and in conferences such as INFOCOM, MOBIHOC, ICNP, ICDCS, SECON, and IPDPS. He has served as the program committee track co-chair of ICDCS and as a program committee member for many conferences such as INFOCOM, MOBIHOC, ICDCS, and WWW.

Yi Zhong received a B.E. degree in School of Information Management, and a M.E. degree in School of Cyber Science and Engineering at Wuhan University, China. Her research interests include optimization algorithms, online scheduling and network security.

Liujing Song is currently with the Computer Network Information Center, Chinese Academy of Sciences, China. She is also a Ph.D. student in the University of Chinese Academy of Sciences, majoring in Computer Software and Theory. Her research interests mainly include network function virtualization and network security.