On-Demand or On-Premises: Online Mitigation of DDoS Attacks via Cloud-Edge Coordination

Yi Zhong[1], Lei Jiao[2], Ruiting Zhou[1], Liujing Song[3]
[1]Wuhan University, China [2]University of Oregon, USA
[3]University of Chinese Academy of Sciences, China

***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 rigorously prove the worst-case, constant competitive performance guarantees of our online algorithms. We finally conduct extensive evaluations and validate the superiority of our approach over multiple existing alternatives.**

I. Introduction

Scrubbing suspicious traffic through cloud scrubbing centers has been increasingly adopted recently to battle Distributed Denial-of-Service (DDoS) attacks [1]. In this approach, suspicious traffic is routed into the dedicated scrubbing centers of DDoS protection services (e.g., Cloudflare [2]) 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 [1], which will impact the contained legitimate traffic towards the enterprise as well. Second, a considerable amount of today's DDoS traffic has low volume, short duration, and is even "hit-and-run" [3], 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 [4]. 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.

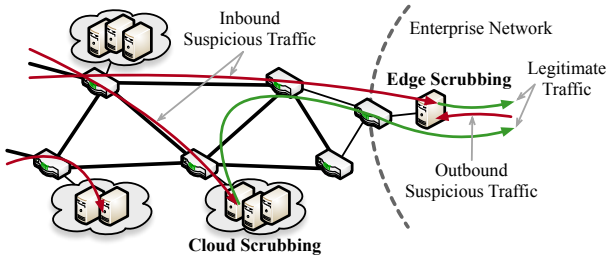

Fig. 1: DDoS mitigation via joint cloud and edge scrubbing

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 [5], 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 normal service and application behaviors, and can potentially better detect or predict abnormal traffic in advance [6]. Such predicted information should be exploited for further cost and performance optimization.

Unfortunately, this problem has never been investigated to the best of our knowledge. Existing research on DDoS mitigation with cloud/edge [1], [3], [4], [7], [8] often treat cloud and edge separately and lack systematic, performance-guaranteed algorithms for the dynamic orchestration of traffic scrubbing. Other related research include network flow scheduling [9]–[13] and network resource allocation [14]–[18]. 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 VI.

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 [19] 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 [20], 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 further extend our study to 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 connect to and 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. We prove that our prediction-aware approach can attain a competitive ratio which is at least as good as that of our prediction-free approach.

Finally, we conduct extensive evaluations to validate the practical performance of our algorithms. Utilizing the public *CICDDoS2019* data [21], Amazon GuardDuty prices [22], US industrial electricity prices [23], 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 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 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.

II. Model and Formulation

A. 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 also 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.

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 [6]. While focusing on accurate predictions in our algorithm design and theoretical analysis, we also experiment with inaccurate predictions later.

B. 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 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.

Algorithm 1: Prediction-Free Online Control Algorithm

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:

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

III. 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.

A. 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 $(y_{ijt} - y_{ijt-1})^+$ [19], 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,

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

1 $\mathcal{I}'_t \stackrel{\text{def}}{=} \mathcal{I} \backslash \{i | \hat{\mathbf{y}}_\mathbf{t} \in \{0,1\}\}$;
2 **while** $|\mathcal{I}'_t| > 1$
3 Select $i_1, i_2 \in \mathcal{I}'_t$, where $i_1 \neq i_2$;
4 $\rho_1 \stackrel{\text{def}}{=} \min\{1 - \hat{y}_{i_1jt}, \hat{y}_{i_2jt}\}$;
5 $\rho_2 \stackrel{\text{def}}{=} \min\{\hat{y}_{i_1jt}, 1 - \hat{y}_{i_2jt}\}$;
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**

$\mathbf{y_{t-1}}$ 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 [20]. 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.

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.

B. 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) [24]; 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 re-solves 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, and beat the latter in terms of the cost evaluated by the objective function of $\mathbf{P}$.

Algorithm 4: Prediction-Aware Online Control Algorithm

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 $\dot{\mathcal{S}}_{t+w-1}$;

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.

IV. Performance Analysis

A. 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)$\sim$(3e) as the bridge. In (3a), we have the expectation because of the randomization in our rounding algorithm. In (3a)$\leqslant$(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)$\leqslant$(3c) naturally holds. In (3c)$\leqslant$(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)$\leqslant$(3e) due to duality, where P''_{opt} is the offline optimum of the problem $\mathbf{P}''$. Finally, in (3e)$\leqslant$(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

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

We highlight that (4a)$\leqslant$(4b) is analogous to (3a)$\leqslant$(3b), with the same r_2; (4b)$\leqslant$(4c) is analogous to (3b)$\leqslant$(3c); and (4d)$\leqslant$(4e) is analogous to (3c)$\leqslant$(3f), with the same r_1 and M. Our only focus here will be (4c)$\leqslant$(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.

B. Analysis of Regularized Fractional Algorithm

We prove (3c)$\leqslant$(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 $\mathbf{P}'$:

$$\begin{aligned}
\min \quad & P' = \textstyle\sum_t P_t \\
& = \textstyle\sum_t \sum_i \sum_j b_{it}\sigma_{jt} y_{ijt} + \sum_t \sum_i \sum_j c_{ij} u_{ijt} \\
& + \textstyle\sum_t \sum_j dC_t z_{jt} \\
\text{s.t.} \quad & \text{(1a)}\sim\text{(1d)}, \\
& u_{ijt} \geqslant y_{ijt} - y_{ijt-1}, \qquad \forall i, \forall j, \forall t, \quad (5a) \\
& u_{ijt} \geqslant 0, y_{ijt} \geqslant 0, z_{jt} \geqslant 0, \qquad \forall i, \forall j, \forall t. \quad (5b)
\end{aligned}$$

$\mathbf{P}'$ 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 $\mathbf{P}''$:

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

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 $\mathbf{D}$**.** 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:

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

Bounding. Now, we are ready to prove (3c)$\leqslant$(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 $\mathbf{P}''$.

C. Analysis of Randomized Rounding Algorithm

We prove (3a)$\leqslant$(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 III-A, to complete the proof.

D. Analysis of Benefit of Prediction-Awareness

We prove (4c)$\leqslant$(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

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

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)$\leqslant$(8b). Second, the proof to this theorem is complete by applying (8a)$\leqslant$(8b) repetitively via setting $\tau=(n-1)w+1$ and $\kappa=nw$, where $n=1,2,...$, according to Algorithm 4.

We finally prove (3e)$\leqslant$(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

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

V. Experimental Evaluation

A. Evaluation Settings

DDoS Traffic: We utilize the public *CICDDoS2019* dataset [21]. The attack, lasting for about 6 hours on March 3, 2019, uses 4 PCs as external attackers to send incoming DDoS flows toward one victim server and uses this victim server to send outgoing DDoS flows. The volumes of the flows are $0\sim4$ MB. For each flow, we have the starting time, the duration, the total volume, and the flow direction indicator. The dataset has about 73 million flows, with about 93% as incoming flows and about 7% as outgoing flows. We choose the first 1000-second data as inputs to our evaluations, containing about 24 million flows. We consider 10 seconds as a single time slot.

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 $0\sim5.8$ MB and the outgoing DDoS flows are $0\sim0.6$ 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. 2.

Cloud Scrubbing Centers: We refer to Amazon GuardDuty [22], whose regional pricing in the US is divided into 6 regions. We thus envisage 6 scrubbing centers, one in each region. We consider 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. Then, the available local scrubbing capacity C_t for t is calculated 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) [23].

Redirection Rules Overhead: The BGP route convergence time can fluctuate from seconds to minutes in reality. Using such 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 test 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 treated as the prediction error. We vary the prediction error in [0, 10%].

Algorithms for Comparison: We implement multiple alternative approaches for comparison: *Random*, *Greedy*, and *Fixed Horizon Control (FHC)*. The random approach treats every remote scrubbing center and the local facility equally, and randomly routes incoming flows to one of those; if the local capacity is not sufficient, it randomly chooses a remote

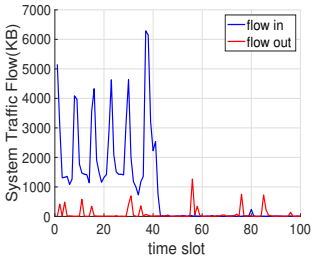

Fig. 2: DDoS flows

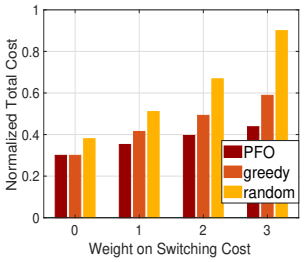

Fig. 3: Impact of switching cost

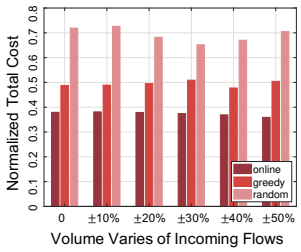

Fig. 4: Impact of incoming flows

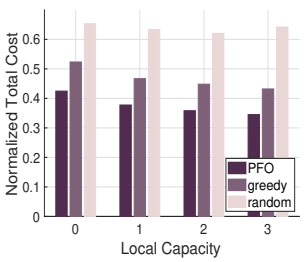

Fig. 5: Impact of local capacity

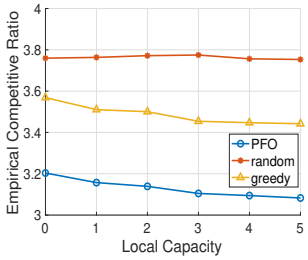

Fig. 6: Empirical competitiveness

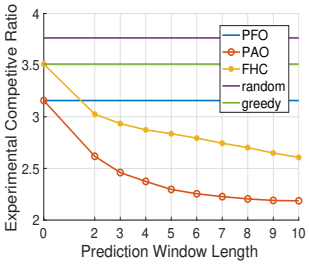

Fig. 7: Impact of window length

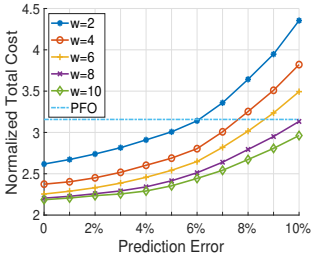

Fig. 8: Impact of prediction error

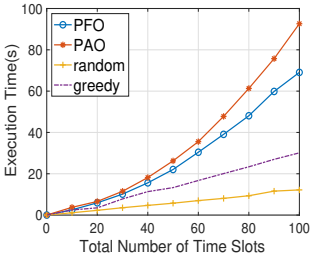

Fig. 9: Execution time

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.

B. Evaluation Results

Fig. 3 demonstrates the total cost over time as the weight associated to the cost of switching between scrubbing centers grows (note that we did not show such weight in our formulations). As the weight becomes larger, the random and the greedy approaches essentially neglect the switching cost and thus can have sharply increasing total cost; our proposed online algorithm always performs best, saving up to 28% total cost due to the explicit accommodation of the switching cost.

Fig. 4 shows the total cost as the volumes of incoming flows vary. For example, $\pm 30\%$ in the figure means that the volume of every incoming flow fluctuates in $[0.7, 1.3]$ times its original volume, and the proportion of fluctuation is generate by uniform distribution. The figure shows that our approach performs well for incoming flows of varying volumes.

Fig. 5 studies how the total cost is impacted by the available local scrubbing capacity. We find the maximum flow volume v at every time slot, and set the local scrubbing facility capacity as 0 (i.e., the cloud-only approach), v, $2v$, and $3v$, respectively. As shown, when the available local capacity becomes larger, our algorithm can choose local scrubbing facility and avoid paying for additional switching cost. The random approach has fluctuating total cost due to its random nature; and the greedy approach has decreasing cost, still larger than our approach.

Fig. 6 depicts the empirical competitive ratios as a function of the local scrubbing capacity, when no prediction is considered. The PFO algorithm always performs well, and achieves a small empirical competitive ratio of less than 3.2.

Fig. 7 visualizes the empirical competitive ratios as the length of the prediction window varies. Except FHC and PAO, all other methods do not exploit predictions and thus their competitiveness stays unchanged. When the prediction window length is 1, PFO and PAO are equivalent. As the prediction window becomes larger, the competitive ratio of PAO becomes smaller, better than FHC by $16\% \sim 24\%$ due to the handling of the switching cost across prediction windows.

Fig. 8 investigates the impact of prediction errors on the performance of PAO, when the predictions about the remote scrubbing price and the traffic flow volume are inaccurate in the prediction window. Using PFO as the baseline which is not impacted by predictions at all, we see that 6% could be the prediction error threshold beyond which PAO using erroneous predictions of 2 time slots makes less sense than PFO using no predictions at all. Also, a longer prediction window can have a higher prediction error threshold, and can make PAO more robust against the prediction errors.

Fig. 9 compares the execution time of the algorithms as the input size increases. We run all algorithms for 10 times and calculate the average; for PAO, we set the prediction window to 5 time slots. Our algorithms run moderately slower than others that have simpler algorithmic logic. The total execution time of our approach is less than 100 seconds, versus the total length of 1000 seconds of all the time slots; as a single time slot can be longer (e.g., hours) in reality, our execution time in the order of magnitude of seconds is acceptable.

VI. Related Work

DDoS Mitigation at Cloud and Edge: Zilberman et al. [1] studied cloud scrubbing center placements, considering traffic footprint, link load, and network latency. Myneni et al. [4] presented a distributed DDoS defense solution deployed at customer and provider edges using neural networks. Bhardwaj

et al. [3] leveraged edge functions for the accelerated detection of DDoS attacks. Zhou et al. [8] designed a DDoS mitigation scheme which flexibly allocates traffic to distributed locations near attack sources. You et al. [7] studied DDoS flow scheduling via auction mechanisms involving scrubbing centers.

These efforts study cloud or edge DDoS defense separately, instead of coordinating them jointly, which lacks performance-guaranteed orchestrations of DDoS mitigation resources. Sitting in the victim's position, our work also differs from them by considering both incoming and outgoing DDoS traffic.

Network Flow Scheduling: Liu et al. [9] utilized priority-based flow scheduling in decentralized federated graph learning systems. Li et al. [10] developed unified dynamic flow and function scheduling for the real-time security function enforcement problem. Lan et al. [11] studied wireless networks with continuous and dynamic flows and proposed online and adaptive scheduling algorithms. Gu et al. [12] maximized network utility given unpredictable network traffic and fairness resource allocation requirements using Lyapunov optimization. Gushchin et al. [13] investigated deadline-constrained flow scheduling via both offline and online algorithms.

These studies do not focus on malicious or DDoS traffic, and can thus hardly be applied to our DDoS problem with unique features such as switching-cost-based traffic rerouting and scrubbing management across locations which fundamentally change the problem and impact the algorithm design.

Resource Allocation with Switching Cost: Ouyang et al. [14] investigated service placements and migrations in edge environments via multi-armed bandits. Gao et al. [15] balanced access, switching, and communication delay using laziness-inspired approaches. Wang et al. [16] designed regularization-based online approaches for edge resource allocation for user mobility. Krishnasamy et al. [17] optimized wireless network energy via managing base station status switching by Markov-chain-based approaches. Chen et al. [18], using game theory, studied drone network interference and channel switching mitigation. Also, there is other research from a pure algorithmic perspective which studied online optimization with switching cost for predicted inputs [24].

This group of literatures adopt various different solution techniques but do not often address *all* of the challenges such as the intractability of integer variables, the online setting, the prediction-awareness, and the competitive analysis for predictive algorithms as in our paper, which are all non-trivial.

VII. Concluding Remarks

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

Acknowledgement

This work is supported in part by the National Natural Science Foundation of China under Grants 62072344 and U20A20177, and in part by the Ripple Faculty Fellowship. The corresponding authors are Lei Jiao (jiao@cs.uoregon.edu) and Ruiting Zhou (ruitingzhou@whu.edu.cn).

Appendix

A. Proof of Theorem 1

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

$$M + dC_t - \alpha_{jt} - \textstyle\sum_j \beta_{jt} + \beta_{jt} - \frac{C_t}{\sigma_{jt}}\mu_{jt} - \xi_t = 0,\ \forall j, \tag{10a}$$

$$b_{it}\sigma_{jt} - \alpha_{jt} - \textstyle\sum_j \beta_{jt} + \beta_{jt} - \mu_{jt} + \frac{c_{ij}}{\delta}\ln\frac{y_{ijt}+\varepsilon}{y_{ijt-1}+\varepsilon} = 0,\ \forall i, \forall j, \tag{10b}$$

$$M - \xi_t = 0,\ \forall j, \tag{10c}$$

$$\alpha_{jt}(\lambda_{jt} - z_{jt} - \textstyle\sum_i y_{ijt}) = 0,\ \forall j, \tag{10d}$$

$$\beta_{jt}(\textstyle\sum_j \lambda_{jt} - 1 - \sum_j z_{jt} + z_{jt} - \sum_i \sum_j y_{ijt} + \sum_i y_{ijt}) = 0,\ \forall j, \tag{10e}$$

$$\mu_{jt}(1 - \frac{C_t}{\sigma_{jt}} z_{jt} - \textstyle\sum_i y_{ijt}) = 0,\ \forall j, \tag{10f}$$

$$\xi_t(1 - Q_t - \textstyle\sum_j z_{jt}) = 0,\ \forall j. \tag{10g}$$

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 (10b).

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

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

$$(\textstyle\sum_n p_n)\ln\frac{\sum_n p_n}{\sum_n q_n} \leqslant \sum_n p_n \ln\frac{p_n}{q_n}, \tag{12a}$$

$$p - q \leqslant p\ln\frac{p}{q}, \tag{12b}$$

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

$$\begin{aligned}&\textstyle\sum_t \hat{y}_{ijt}\ln\frac{\hat{y}_{ijt}+\varepsilon}{\hat{y}_{ijt-1}+\varepsilon}\\ =&\textstyle\sum_t(\hat{y}_{ijt}+\varepsilon)\ln\frac{\hat{y}_{ijt}+\varepsilon}{\hat{y}_{ijt-1}+\varepsilon} - \sum_t \varepsilon\ln\frac{\hat{y}_{ijt}+\varepsilon}{\hat{y}_{ijt-1}+\varepsilon}\\ \geqslant&\textstyle(\sum_t(\hat{y}_{ijt}+\varepsilon))\ln\frac{\sum_t(\hat{y}_{ijt}+\varepsilon)}{\sum_t(\hat{y}_{ijt-1}+\varepsilon)} + (\hat{y}_{ik0}+\varepsilon)\ln\frac{\hat{y}_{ij0}+\varepsilon}{\hat{y}_{ijT}+\varepsilon}\\ \geqslant&\textstyle\sum_t(\hat{y}_{ijt}+\varepsilon) - \sum_t(\hat{y}_{ijt-1}+\varepsilon) + \hat{y}_{ij0} - \hat{y}_{ijT} = 0.\end{aligned}$$

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

$$
\begin{aligned}
&\textstyle\sum_t \sum_i \sum_j b_{it}\sigma_{jt}\hat{y}_{ijt} + \sum_t \sum_j (M + dC_t) z_{jt} + \sum_t MQ_t \\
\leqslant &\textstyle\sum_t \sum_j (\alpha_{jt} + \sum_j \beta_{jt} - \beta_{jt} + \frac{C_t}{\sigma_{jt}}\mu_{jt} + \xi_t)\hat{z}_{jt} + \sum_t \xi_t Q_t \\
+ &\textstyle\sum_t \sum_i \sum_j (\alpha_{jt} + \sum_j \beta_{jt} - \beta_{jt} + \mu_{jt} - \frac{c_{ij}}{\delta} \ln \frac{\hat{y}_{ijt}+\varepsilon}{\hat{y}_{ijt-1}+\varepsilon}) \hat{y}_{ijt} \qquad (14a) \\
\leqslant &\textstyle\sum_t \sum_j (\alpha_{jt} + \sum_j \beta_{jt} - \beta_{jt} + \frac{C_t}{\sigma_{jt}}\mu_{jt} + \xi_t)\hat{z}_{jt} + \sum_t \xi_t Q_t \\
+ &\textstyle\sum_t \sum_i \sum_j (\alpha_{jt} + \sum_j \beta_{jt} - \beta_{jt} + \mu_{jt}) \hat{y}_{ijt} \qquad (14b) \\
= &\textstyle\sum_t \sum_j \alpha_{jt}\lambda_{jt} + \sum_t \sum_j \beta_{jt} (\sum_j \lambda_{jt} - 1) + \sum_t \sum_j \mu_{jt} \\
+ &\textstyle\sum_t \xi_t \qquad (14c) \\
= &D + \textstyle\sum_t Q_t \xi_t \leqslant 2D.
\end{aligned}
$$

(14a) is due to (10a)∼(10c). (14b) is due to $\hat{y}_{ijt} \ln \frac{\hat{y}_{ijt}+\varepsilon}{\hat{y}_{ijt-1}+\varepsilon} \geqslant 0$, as shown right above. (14c) follows from (10d)∼(10g). 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,

$$
\begin{aligned}
&\textstyle\sum_t \sum_i \sum_j c_{ij}(\hat{y}_{ijt} - \hat{y}_{ijt-1})^+ \\
\leqslant &\textstyle\sum_t \sum_i \sum_j c_{ij}(\hat{y}_{ijt} - \hat{y}_{ijt-1}) \qquad (15a) \\
\leqslant &\textstyle\sum_t \sum_i \sum_j c_{ij}(\hat{y}_{ijt} + \varepsilon) \ln \frac{(\hat{y}_{ijt}+\varepsilon)}{(\hat{y}_{ijt-1}+\varepsilon)} \qquad (15b) \\
= &\delta \textstyle\sum_t \sum_i \sum_j (\hat{y}_{ijt} + \varepsilon)(\alpha_{jt} + \sum_j \beta_{jt} - \beta_{jt} + \mu_{jt} - b_{it}\sigma_{jt}) \qquad (15c) \\
\leqslant &(1 + |\mathcal{I}|\varepsilon)\delta \textstyle\sum_t \sum_j (\alpha_{jt} + \sum_j \beta_{jt} - \beta_{jt} + \mu_{jt}) \qquad (15d) \\
\leqslant &(1 + |\mathcal{I}|\varepsilon)\delta \big(\textstyle\sum_t \sum_j \lambda_{jt}\alpha_{jt} + \sum_t \sum_j (\sum_j \lambda_{jt} - 1)\beta_{jt} \\
&+ \textstyle\sum_t \sum_j \mu_{jt} \big) \qquad (15e) \\
\leqslant &(1 + |\mathcal{I}|\varepsilon)\delta D.
\end{aligned}
$$

(15a) is owing to the definition of $(\hat{y}_{ijt} - \hat{y}_{ijt-1})^+$. (15b) is due to (12b). (15c) follows from (10b). (15d) holds because of (1b). (15e) is due to our assumption of $\sum_j \lambda_{jt} \geqslant 1$.

B. Proof of Theorem 2

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

$$
\begin{aligned}
&E(\textstyle\sum_t \sum_i \sum_j b_{it}\sigma_{jt}\bar{y}_{ijt} + \sum_t \sum_j dC_t z_{jt}) \\
\leqslant &\Psi'_{yz} \textstyle\sum_t \sum_j E\left(z_{jt} + \sum_i \bar{y}_{ijt}\right) \qquad (16a) \\
\leqslant &\Psi'_{yz} \textstyle\sum_t \sum_j 1 \qquad (16b) \\
\leqslant &|\mathcal{J}|\Psi'_{yz} \textstyle\sum_t \sum_j \lambda_{jt} \qquad (16c) \\
\leqslant &|\mathcal{J}|\Psi'_{yz} \textstyle\sum_t \sum_j (z_{jt} + \sum_i \hat{y}_{ijt}) \qquad (16d) \\
\leqslant &\Psi_y (\textstyle\sum_t \sum_i \sum_j b_{it}\sigma_{jt} y_{ijt} + \sum_t \sum_j dC_t z_{jt}). \qquad (16e)
\end{aligned}
$$

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 (16b). (16c) holds since we assume $\sum_j \lambda_{jt} \geqslant 1$. (16d) is due to (1a).

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

$$
\begin{aligned}
&E\big(\textstyle\sum_t \sum_i \sum_j c_{ij}(\bar{y}_{ijt} - \bar{y}_{ijt-1})^+\big) \\
\leqslant &E\big(\textstyle\sum_t \sum_i \sum_j c_{ij}\bar{y}_{ijt}\big) \qquad (17a) \\
\leqslant &\max_{i,j} c_{ij} E(\textstyle\sum_t \sum_i \sum_j \bar{y}_{ijt}) \qquad (17b) \\
\leqslant &\Psi_z (\textstyle\sum_t \sum_i \sum_j b_{it}\sigma_{jt} y_{ijt} + \sum_t \sum_j dC_t z_{jt}). \qquad (17c)
\end{aligned}
$$

(17a) follows from $(\cdot)^+ \overset{\text{def}}{=} \max\{\cdot, 0\}$. (17b)∼(17c) is analogous to (16a)∼(16e).

REFERENCES

[1] 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.

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

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

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

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

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

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

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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 *IEEE INFOCOM*, 2019.

[16] 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.

[17] 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.

[18] 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.

[19] N. Buchbinder, S. Chen, and J. Naor, "Competitive analysis via regularization," in *ACM-SIAM SODA*, 2014.

[20] 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.

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

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

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

[24] 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.