Competitive Online Convex Optimization With Switching Costs and Ramp Constraints

Ming Shi, *Student Member, IEEE*, Xiaojun Lin, *Fellow, IEEE*, and Sonia Fahmy, *Senior Member, IEEE*

***Abstract*— We investigate competitive online algorithms for online convex optimization (OCO) problems with linear in-stage costs, switching costs and ramp constraints. While OCO problems have been extensively studied in the literature, there are limited results on the corresponding online solutions that can attain small competitive ratios. We first develop a powerful computational framework that can compute an optimized competitive ratio based on the class of affine policies. Our computational framework can handle a fairly general class of costs and constraints. Compared with other competitive results in the literature, a key feature of our proposed approach is that it can handle scenarios where infeasibility may arise due to hard feasibility constraints. Second, we design a robustification procedure to produce an online algorithm that can attain good performance for both average-case and worst-case inputs. We conduct a case study on Network Functions Virtualization (NFV) orchestration and scaling to demonstrate the effectiveness of our proposed methods.**

***Index Terms*— Competitive online algorithms, network functions virtualization (NFV), online convex optimization, ramp constraints, robustification, switching costs.**

I. Introduction

WE STUDY online convex optimization (OCO) with switching costs and ramp constraints, which has become an important tool for modeling many classes of practical decision problems with uncertainty, including cloud or edge computing [2]–[6], computer networks [1], [7]–[9], cyber-physical systems [10], [11], wireless systems [12], [13] and machine learning [14]–[17]. In the type of OCO problem that we are interested in, at each time t, the environment (or adversary) reveals the input $\vec{A}(t)$. The decision maker then must choose the decision $\vec{X}(t)$ from a convex set and incurs a linear cost $C_t(\vec{X}(t), \vec{A}(t))$. Additionally, there is a switching cost that penalizes the change $|\vec{X}(t) - \vec{X}(t-1)|$ for each time t and/or a ramp constraint on the magnitude of the change $\vec{X}(t) - \vec{X}(t-1)$. The goal is to minimize the overall cost, which is non-linear (and convex) due to the switching cost. Further, since future inputs, i.e., $\vec{A}(t+1)$, $\vec{A}(t+2), \ldots, \vec{A}(\mathcal{T})$, are not revealed at time t, this problem becomes an online decision problem. Clearly, this formulation is general and can model many important online decision problems. For example, in the Network Functions Virtualization (NFV) orchestration and scaling problem [18]–[20], a data-center operator must decide where to instantiate Virtualized Network Functions (VNFs) as virtual machines (VMs) or containers (such as Docker [21]) running on physical servers in order to process incoming traffic. Here, $\vec{A}(t)$ represents the traffic load, which can be uncertain before time t; $\vec{X}(t)$ represents the mapping from VNFs to VMs or containers; the linear cost $C_t(\vec{X}(t), \vec{A}(t))$ represents VM/container cost and/or distance cost (e.g., latency) [22]. Moreover, the switching cost captures the overhead for migrating demand/state among different VNF instances and the cost of instantiating and tearing down VNF instances. Finally, since the routing of the traffic is stateful, there could be a ramp constraint on the rate with which traffic can be rerouted. As another example, in the real-time dispatch problem in power systems [10], [11], [23], the system operator needs to decide how to adjust the power level of the generators to balance the electricity demand. Here, $\vec{A}(t)$ represents the uncertain demand and renewable supply revealed on different buses at time t; $\vec{X}(t)$ represents the dispatch decisions of the generators; the linear cost $C_t(\vec{X}(t), \vec{A}(t))$ represents the generation cost of the dispatch decisions. Finally, generators have ramp constraints so that their power output level can at most change by a given value each time.

In this paper, we aim to develop online algorithms with low competitive ratios for this type of OCO problem. Here, the competitive ratio is the maximum ratio of the cost of an online algorithm to that of the optimal offline solution (the latter assuming that all inputs are known in advance), taken over all possible input sequences. For settings with switching costs but no ramp constraints, there has been a considerable body of work on competitive online convex optimization (OCO). For some restrictive settings, e.g., when there is only 1 decision variable subject to switching costs, it is possible to construct online algorithms with low and constant competitive ratios [7], [10], [24]. In contrast, the competitive ratios for general OCO problems (e.g., by using the regularization method [25]) are usually much larger when there are no constraints on the future inputs. Thus, a series of work has studied how to utilize partial future information to improve the competitive performance of online algorithms for general OCO problems. Specifically, references [7] and [8] assume that there is a perfect look-ahead window where the future inputs are precisely known, which

Manuscript received May 14, 2020; revised November 16, 2020; accepted January 6, 2021; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor L. Huang. Date of publication February 3, 2021; date of current version April 16, 2021. This work was supported in part by the National Science Foundation under grants CNS-1457137, ECCS-1509536, OAC-1738981, and CNS-1717493; and in part by the Army Research Office under grant W911NF-14-1-0368. A preliminary version of this work has appeared in 37th IEEE International Conference on Computer Communications (IEEE INFOCOM), 2018 [1]. *(Corresponding author: Ming Shi.)*

Ming Shi and Xiaojun Lin are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: sming@purdue.edu; linx@purdue.edu).

Sonia Fahmy is with the Department of Computer Science, Purdue University, West Lafayette, IN 47907 USA (e-mail: fahmy@purdue.edu).

Digital Object Identifier 10.1109/TNET.2021.3053910

can be unrealistic when short-term predictions also incur errors. The work in [5], [6] uses imperfect predictions of future inputs. However, they only show that the competitive differences of their proposed algorithms are upper-bounded. Thus, the competitive ratios may still be fairly large. Further, [5]–[8] do not consider ramp constraints. Finally, reference [26] studies OCO with ramp constraints, but it does not consider demand-supply balance constraints. As we will discuss shortly, when both ramp constraints and demand-supply constraints are imposed (which we refer to as "hard feasibility constraints"), the algorithms in [5]–[8], [26] could incur infeasibility issues. To the best of our knowledge, there is no systematic framework to optimize the competitive ratios of online algorithms under imprecise future information for the type of OCO problems that this paper studies.

To address this open question, our first contribution (in Sec. III) is to develop a general and tractable framework that allow us to find online algorithms that can utilize partial future information to optimize competitive ratios for OCO problems with both switching costs and ramp constraints. Capitalizing on the ideas from robust optimization [27], we consider the case where the future uncertain inputs, $\vec{A}(1), \vec{A}(2), \ldots, \vec{A}(\mathcal{T})$, are from an uncertainty set $\mathcal{U}$. In practice, such an uncertainty set $\mathcal{U}$ can be obtained from imprecise forecasts and historical data [23], [27]–[29]. Yet, searching among all possible online decisions appears to be intractable. Instead, in order to obtain simpler policies with reasonably good performance, we focus on affine policies, where the decision $\vec{X}(t)$ at time t is an affine function of the input $\vec{A}(t)$, i.e., $\vec{X}(t) = \vec{\eta}(t) + \boldsymbol{H}(t)\vec{A}(t)$. Thus, designing an online algorithm boils down to designing the parameters $\vec{\eta}(t)$ and $\boldsymbol{H}(t)$ (which depends only on the uncertainty set $\mathcal{U}$ but not the actual inputs). Through this restriction to affine policies, we can formulate the problem of optimizing the competitive ratio as a minimax optimization problem. We call the resulting online algorithm the Robust Affine Policy (RAP). Since this optimization problem is still non-convex, we propose approximations that effectively convexify the problem and make it tractable. In this way, our proposed computational framework can be used to design online algorithms with optimized competitive ratios for OCO problems with fairly complex structures and constraints. We note that the idea of affine policies has been used in adjustable robust optimization [29] to minimize the worst-case cost. In contrast, our approach applies affine policies to minimize the competitive ratio. This approach has not been studied before and gives rise to new technical difficulties as we discuss in Sec. III.

A key feature of our proposed approach is that it can gracefully handle situations where infeasibility may arise due to hard feasibility constraints in the OCO problems. By "hard feasibility constraints," we refer to the situation where there exist both ramp constraints and demand-supply balance constraints. In this situation, if the past decisions were not properly chosen, there may not exist feasible decision for the future. This infeasibility problem may persist even in the case with look-ahead [5]–[8], [26]. For example, in a look-ahead window from time t to $t+K-1$, if the demands are low, the algorithms in [5]–[8], [26] may keep a low decision for $\vec{X}(t)$. Then, when the demand at time $t+K$ is too high, there may not exist any $\vec{X}(t+K)$ that can meet the demand, because the ramp constraint limits the increase of $\vec{X}(t+K)$. Such an infeasibility problem can occur in many online decision problems, such as Network Functions Virtualization and power systems. Unfortunately, the studies in [5]–[8], [26] do not consider such hard feasibility constraints because they do not simultaneously enforce ramp constraints and demand-supply balance constraints. As a result, their competitive guarantees would not hold when there were hard feasibility constraints. To the best of our knowledge, our proposed approach is the first to give online algorithms with optimized competitive ratios with or without such constraints.

Our second key contribution (in Sec. IV) is to resolve a dilemma between the worst-case and average-case performance. Note that while our proposed Robust Affine Policy (RAP) in Sec. III is optimized for the worst-case competitive ratio, it may be too conservative and thus incur high costs for average-case inputs. Other heuristic algorithms, such as Receding Horizon Control (RHC) [6], [30], [31] (discussed in Sec. VI), may perform well for the average case, but produce inferior competitive ratios for worst-case scenarios. Thus, an open question is whether one can get the best of both worlds. We address this dilemma by providing a "robustification" procedure. Given any online algorithm π_0 that is perceived to have good average-case performance, we intelligently combine π_0 with RAP to produce a new online algorithm with the same worst-case competitive ratio as RAP while still attaining comparable average-case performance to π_0. We note that this "robustification" idea was first introduced in our earlier work [28]. However, our OCO problem formulation is much more general, requiring a new robustification procedure to be developed. We use Network Functions Virtualization (NFV) [18], [20] as a case study and simulate the robustified version of RHC. Our simulation results in Sec. VI show that the robustified-RHC algorithm performs close to RHC when the uncertainty is low. When the uncertainty is high, the robustified-RHC algorithm performs significantly better than RHC, especially for worst-case inputs.

As discussed above, our work is related to robust optimization [27], [29], but differs in that we focus on a different objective of competitive ratios rather than worst-case costs. Our NFV case study is also related to the literature of NFV orchestration and scaling. However, most existing studies either assume a static model [22], [32]–[34], or provide heuristic online algorithms without any performance guarantees [35], [36]. Although [19] and [37] study online NFV orchestration and scaling, they do not consider the distance cost (e.g., latency), which is an important cost component, especially when optimizing over multiple data centers [20], [22]. Further, the setting in [37] is special in that the decision variables for different types of Virtual Machines (VMs) can be decoupled in such a way that they are independent of each other, and thus each of the subproblems can be simplified to a 1-dimensional problem. In contrast, the decision variables in the more general OCO setting in this paper are always

coupled through the constraints. More recently, reference [20] uses the regularization method to develop online algorithms for NFV orchestration and scaling over multiple data centers. Compared with [19], [20], [37], one of the key differences of our work is that we utilize partial future knowledge, i.e., in the form of an uncertainty set $\mathcal{U}$, to obtain potentially smaller competitive ratios. In contrast, it is unclear how to generalize the approaches in [19], [20], [37] to utilize such partial future knowledge. Moreover, in deriving their competitive ratios, the studies in [19], [20], [37] do not consider constraints on the number of servers available or the ramp constraints on the rerouting decisions, and thus cannot handle the hard feasibility constraints described above. NFV orchestration and scaling is also related to the facility location (FL) [38] and generalized assignment problem (GAP) [39], for which competitive online algorithms have been developed. However, in NFV, the demand fluctuates (both increases and decreases) over time in both the online and offline settings. In contrast, online FL and GAP problems usually assume that new demand is sequentially added over time towards a final offline setting where all demand is present. Further, the cost constraints of OCO problems are usually more general, e.g., involving switching costs and ramp constraints. Thus, it is unclear how to apply the competitive results from this literature to OCO and online NFV orchestration and scaling problems.

II. Problem Formulation

We now present our model for online convex optimization (OCO) problems with linear in-stage costs, switching costs and ramp constraints.

A. OCO With Switching Costs and Ramp Constraints

In the OCO problem that we consider, there are $\mathcal{T}$ rounds of decisions, $t = 1, 2, \ldots, \mathcal{T}$. There is a cost-function $C_t(\vec{X}(t), \vec{A}(t))$ for each time t, which is a function of the input $\vec{A}(t) = [a_m(t), m = 1, \ldots, M]^{\mathrm{T}} \in \mathbb{R}^{M\times 1}$ (e.g., traffic load) revealed by the environment at time t, and the action $\vec{X}(t) = [x_n(t), n = 1, \ldots, N]^{\mathrm{T}} \in \mathbb{R}^{N\times 1}$ taken by the decision maker (e.g., system administrator) at time t ($[\cdot]^{\mathrm{T}}$ denotes the transpose of a vector or matrix). Throughout this paper, we assume that $C_t(\cdot,\cdot)$ is a linear function of $(\vec{X}(t), \vec{A}(t))$. Further, there is a switching cost $\vec{\beta}^{\mathrm{T}}|\vec{X}(t) - \vec{X}(t-1)|$ that penalizes the change of decisions at time t, where $\vec{\beta}$ is a fixed vector in $\mathbb{R}^{N\times 1}$. In addition, the action $\vec{X}(t)$ must be chosen to satisfy certain constraints. We assume that one set of constraints $\mathbb{X}_t(\vec{A}(t))$, which may depend on the input $\vec{A}(t)$, can be written as a linear inequality in $(\vec{X}(t), \vec{A}(t))$, i.e.,

$$\boldsymbol{B_1}\vec{X}(t) + \boldsymbol{B_2}\vec{A}(t) \le 0, \text{ for all time } t, \tag{1}$$

where $\boldsymbol{B_1}$ is an $R \times N$ matrix and $\boldsymbol{B_2}$ is an $R \times M$ matrix. Further, there may be ramp constraints,

$$|\vec{X}(t) - \vec{X}(t-1)| \le \vec{\Delta}^X, \text{ for all time } t. \tag{2}$$

As we will illustrate with a case study in Sec. II-D, this construction can model several types of costs and constraints. Let $\vec{A}(t_1 : t_2)$ denote the input sequence $\vec{A}(t)$ from $t = t_1$ to t_2. Define $\vec{X}(t_1 : t_2)$ similarly.

At each time t, the environment reveals $\vec{A}(t)$ first. Then the decision maker picks the action $\vec{X}(t)$ and incurs the in-stage cost $C_t(\vec{X}(t), \vec{A}(t))$ and the switching cost $\vec{\beta}^{\mathrm{T}}|\vec{X}(t) - \vec{X}(t-1)|$. Note that although $C_t(\cdot,\cdot)$ is linear, the switching cost still makes the whole problem convex. Further, this problem is an online problem because the decision maker does not know the future values of $\vec{A}(t+1)$, $\vec{A}(t+2), \ldots, \vec{A}(\mathcal{T})$ when she makes the decision $\vec{X}(t)$.

As we discussed in the introduction, the combination of the linear constraint (1) and the ramp constraint (2) may lead to infeasibility. If $\vec{X}(t-1)$ is not properly chosen, the ramp constraint limits how far $\vec{X}(t)$ can deviate from $\vec{X}(t-1)$. Then, there may not exist a feasible point that simultaneously satisfies (1) and (2). For example, this infeasibility can occur when the demand increases suddenly and the traffic cannot be rerouted as quickly to serve the demand. Thus, a key contribution of our work is to be able to deal with cases with or without such "hard infeasibility constraints."

B. Uncertainty Set

Recall that the input $\vec{A}(t)$ is unknown to the online algorithm until time t. Intuitively, if $\vec{A}(t)$ can vary in arbitrary ways, one may have to take the most conservative decisions to avoid future infeasibility. Thus, in order to make the online decision problem practically more interesting, we introduce an uncertainty set to model the set of uncertain inputs that we care about. Specifically, we assume that the trajectory $\vec{A}(1), \vec{A}(2), \ldots, \vec{A}(\mathcal{T})$ chosen by the environment must be from an uncertainty set $\mathcal{U}$. We expect that this uncertainty set $\mathcal{U}$ can be constructed from prediction and historical data [23], [27]–[29]. Next, we describe three ways (that can be used in combination) to formulate the uncertainty set $\mathcal{U}$.

(i) Day-ahead prediction: Let $\vec{A}^{\mathrm{DAP}}(1{:}\mathcal{T})$ denote a predicted trajectory of $\vec{A}(1{:}\mathcal{T})$. We may assume that the real trajectory $\vec{A}(1{:}\mathcal{T})$ must be within a neighborhood around $\vec{A}^{\mathrm{DAP}}(1{:}\mathcal{T})$,

$$\vec{A}^{\mathrm{lower}}(t) \le \vec{A}(t) \le \vec{A}^{\mathrm{upper}}(t), \text{ for all time } t, \tag{3}$$

where, for all time t, each entry of the upper bound $\vec{A}^{\mathrm{upper}}(t) = [a_m^{\mathrm{upper}}(t), m = 1, \ldots, M]^{\mathrm{T}} \in \mathbb{R}^{M\times 1}$ and lower bound $\vec{A}^{\mathrm{lower}}(t) = [a_m^{\mathrm{lower}}(t), m = 1, \ldots, M]^{\mathrm{T}} \in \mathbb{R}^{M\times 1}$ is given by

$$a_m^{\mathrm{upper}}(t) = (1 + \epsilon_m(t))\, a_m^{\mathrm{DAP}}(t), \text{ for all } m,$$
$$a_m^{\mathrm{lower}}(t) = \max\{0, (1 - \epsilon_m(t))\, a_m^{\mathrm{DAP}}(t)\}, \text{ for all } m,$$

and $\epsilon_m(t)$ is the uncertainty level for time t.

(ii) Demand changing speed: Often, demand (e.g., traffic or renewable energy) may not change arbitrarily fast. We can model such knowledge by imposing

$$|\vec{A}(t) - \vec{A}(t-1)| \le \vec{\Delta}^A, \text{ for all time } t. \tag{4}$$

(iii) The different elements of $\vec{A}(t)$ may not hit the upper or lower bounds in (3) simultaneously. Thus, we can impose the following constraint (known as the "budget" constraint in the robust optimization literature [27, p. 47]),

$$\sum_{m=1}^{M} \frac{|a_m(t) - a_m^{\mathrm{DAP}}(t)|}{\epsilon_m(t) \cdot a_m^{\mathrm{DAP}}(t)} \le \Gamma, \text{ for all time } t. \tag{5}$$

Clearly, if $\Gamma = 0$, the uncertainty set only contains the day-ahead prediction $\vec{A}^{\text{DAP}}(1{:}\mathcal{T})$. Thus, the model becomes deterministic. As Γ increases, more uncertainty will be considered.

The uncertainty set $\mathcal{U}$ that we use in this paper is specified by a combination of the above constraints. We note that the constraint (4) introduces temporal coupling of the inputs, which can be used to refine the near-term future uncertainty. Specifically, at any time t, $\vec{A}(1{:}t)$ has already been revealed to the online algorithm. Thus, the future uncertainty remaining in the interval from time $t+1$ to time $\mathcal{T}$ can be written as

$$\mathcal{U}_{|\vec{A}(1:t)} = \Big\{ \vec{A}(t+1:\mathcal{T}) \mid \exists \vec{A}'(1:\mathcal{T}) \in \mathcal{U}, \text{ such that, } \vec{A}'(1:t) = \vec{A}(1:t),\ \vec{A}'(t+1:\mathcal{T}) = \vec{A}(t+1:\mathcal{T}) \Big\}, \quad (6)$$

where the subscript "$|\vec{A}(1:t)$" emphasizes that the corresponding uncertainty set is conditioned on $\vec{A}(1:t)$.

C. The Performance Metric

As we discussed earlier, the total cost incurred by the decision maker is given by

$$C(\vec{X}(0), \vec{X}(1{:}\mathcal{T}), \vec{A}(1{:}\mathcal{T})) = \sum_{t=1}^{\mathcal{T}} \left\{ C_t(\vec{X}(t), \vec{A}(t)) + \vec{\beta}^{\text{T}} \left| \vec{X}(t) - \vec{X}(t-1) \right| \right\}. \quad (7)$$

For an online algorithm π, at each time t the decision $\vec{X}(t)$ can only be based on the already-known inputs $\vec{A}(1{:}t)$ and knowledge about the future uncertainty $\mathcal{U}_{|\vec{A}(1:t)}$ given by (6). Let $C^{\pi}(\vec{A}(1{:}\mathcal{T}))$ be the total cost of algorithm π. We compare it with an offline solution that is assumed to know the entire input $\vec{A}(1{:}\mathcal{T})$ ahead of time. We denote the cost of the optimal offline solution as $C^{\text{OPT}}(\vec{A}(1{:}\mathcal{T}))$, which is the optimal value of the following optimization problem,

$$\min_{\{\vec{X}(0:\mathcal{T}):(1),(2)\}} C(\vec{X}(0), \vec{X}(1{:}\mathcal{T}), \vec{A}(1{:}\mathcal{T})). \quad (8)$$

Then, the competitive ratio of algorithm π, given by

$$\text{CR}^{\pi} \triangleq \max_{\{\vec{A}(1:\mathcal{T}) \in \mathcal{U}\}} \frac{C^{\pi}(\vec{A}(1{:}\mathcal{T}))}{C^{\text{OPT}}(\vec{A}(1{:}\mathcal{T}))}, \quad (9)$$

is the worst-case ratio between the online cost and the optimal offline cost, over all possible inputs from the uncertainty set.

We are thus interested in online solutions to OCO with small competitive ratios. Although the notions of uncertainty sets, as well as the affine policies that will be introduced later, are from the robust optimization literature [27], our objective in (9) is quite different. In the robust optimization literature, the objective is usually to minimize the worst-case (absolute) cost, i.e., $\max_{\{\vec{A}(1:\mathcal{T}) \in \mathcal{U}\}} C^{\pi}(\vec{A}(1{:}\mathcal{T}))$. Our objective of the competitive ratio, which is commonly used in the CS literature, instead focuses on a relative ratio comparing with the optimal offline solution. This difference leads to new technical difficulties in the optimization problem. In some way, competitive ratios can be viewed as less conservative than robust optimization because we do *not* only care about the worst-case cost.

TABLE I
NOTATIONS FOR NFV NETWORK

$v \in \{1, 2, ..., \mathcal{V}\}$	server v
$f \in \{1, 2, ..., \mathcal{F}\}$	VNF f
$l \in \{1, 2, ..., \mathcal{L}\}$	traffic flow l
s_l	source of the traffic flow l
d_l	destination of the traffic flow l
SC^l	ordered service chain of the traffic flow l
$\lvert\text{SC}^l\rvert$	length of the service chain SC^l
$a_l(t)$	the incoming rate of the traffic flow l

D. A Case Study

As a concrete example, we now use the Network Functions Virtualization (NFV) orchestration and scaling problem [18]–[20] to illustrate how our model can be used to study practical costs and constraints. We will also use it in our numerical evaluation in Sec. VI. Our model may appear somewhat complicated as we model service chaining and traffic resizing. Nonetheless, it demonstrates the power of the proposed OCO framework for handling complex and practical problems.

The NFV system is modeled as an undirected graph $G(V, E)$. Each vertex $v \in V \triangleq \{1, \ldots, \mathcal{V}\}$, where Virtual Machines (VMs) can be placed to deploy the Virtualized Network Functions (VNFs). We assume that there are $\mathcal{F}$ types of VNFs. Each type $f = 1, \ldots, \mathcal{F}$ of VNFs could be firewalls, proxies, network address translators (NATs) and intrusion detection systems (IDSs).

Several types of VNFs may need to be "chained" together into a service chain to process traffic flows. We assume that there are $\mathcal{L}$ traffic flows. Each traffic flow $l = 1, 2, \ldots, \mathcal{L}$ enters at its source location $s_l \in V$ and leaves at the destination $d_l \in V$. Denote $\text{S} \triangleq \{s_1, \ldots, s_{\mathcal{L}}\}$ and $\text{D} \triangleq \{d_1, \ldots, d_{\mathcal{L}}\}$. We define a distance cost $d_{v',v}$ between two server v' and v, which may represent, e.g., the number of hops between them or the bandwidth cost. Each traffic flow l requires a service chain $\text{SC}^l : f_1^l \to f_2^l \to \ldots \to f_{|\text{SC}^l|}^l$, where $|\text{SC}^l|$ is the length of the service chain SC^l, and $f_i^l \in \{1, 2, \ldots, \mathcal{F}\}$ represents the VNF required. Further, we let $\textit{next}(f|\text{SC}^l)$ denote the next VNF on SC^l, and let $\textit{prev}(f|\text{SC}^l)$ denote the previous VNF on SC^l. For convenience, we extend the service chain SC^l to include the source node $f_0^l = s_l$ and the destination node $f_{|\text{SC}^l|+1}^l = d_l$. Correspondingly, we let $\textit{next}(f_{|\text{SC}^l|}^l|\text{SC}^l) = d_l$, $\textit{prev}(d_l|\text{SC}^l) = f_{|\text{SC}^l|}^l$, $\textit{next}(s_l|\text{SC}^l) = f_1^l$, and $\textit{prev}(f_1^l|\text{SC}^l) = s_l$. All the notations are listed in Table I.

(i) Input and service chain routing: The input to our online optimization problem at time t is the incoming rate $a_l(t)$ of each traffic flow l. We need to set up VNFs according to the service chain SC^l to process the traffic. Suppose VNF f_i^l is deployed at server v and VNF $\textit{next}(f_i^l|\text{SC}^l)$ is deployed at server v', we use $x_{l,f_i^l,v,\textit{next}(f_i^l|\text{SC}^l),v'}(t)$ to denote the amount of traffic of flow l that is routed from VNF f_i^l in server v to VNF $\textit{next}(f_i^l|\text{SC}^l)$ in server v'. Denote the total amount of incoming traffic of flow l processed by VNF f_i^l at node v as $\overline{x}_{l,f_i^l,v}(t)$. Note that some VNFs (e.g., VPN, firewalls and intrusion detection systems) may change the rate of the data flow due to processing [40], [41]. We model this traffic resizing by $w_{l,i}$, which is the ratio between the outgoing rate and the incoming rate of flow l at VNF f_i^l. We then have the

TABLE II
DECISION VARIABLES, COST FUNCTIONS AND COEFFICIENTS

$y_{f,v}(t)$	size of VNF f deployed in server v at time t
$x_{l,f_i^l,v,next(f_i^l\|\mathrm{SC}^l),v'}(t)$	egress rate of flow l from VNF f_i^l in server v to VNF *next*$(f_i^l\|\mathrm{SC}^l)$ in server v'
$\overline{x}_{l,f_i^l,v}(t)$	ingress rate of flow l to VNF f_i^l in server v
$c_{f,v}(t)$	cost of hosting 1 unit of VNF f in server v
$d_{v',v}$	distance cost of routing a unit of traffic load from a vertex v' to another vertex v
$\beta_{1,f,v}$	switching-cost coefficient for the overhead in instantiating and tearing-down VNF instances that processing VNF f in server v
$\beta_{2,f',v',f,v}$	switching-cost coefficient for state migration in rerouting the flow from going to VNF f' in server v' to going to VNF f in server v
$b_{f,v}$	processing capacity of one copy of VNF instance that processing VNF f in server v
$z_{f,v}$	amount of resources that each unit-size function f needs in server v
Z_v	total amount of resources available in server v
$\Delta^X_{f',v',f,v}$	ramp-constraint parameter for rerouting the traffics
$\Delta^Y_{f,v}$	ramp-constraint parameter for migrating VNF instances

following flow-balance relationship,

$$\overline{x}_{l,f_i^l,v}(t) = \sum_{v'\neq v} x_{l,prev(f_i^l|\mathrm{SC}^l),v',f_i^l,v}(t) \quad (10a)$$

$$\sum_{v'\neq v} x_{l,f_i^l,v,next(f_i^l|\mathrm{SC}^l),v'}(t) = w_{l,i}\overline{x}_{l,f_i^l,v}(t). \quad (10b)$$

Finally, at the source s_l of each flow l, the demand-supply balance constraint below must be satisfied,

$$\overline{x}_{l,f_0^l,s_l}(t) = a_l(t). \quad (11)$$

All decision variables, cost coefficients and parameters are listed in Table II.

(ii) VNF instances, resource and ramp constraints: Let $y_{f,v}(t)$ denote the size of VNF f at server v. Assume that the processing capacity of one unit-size of VNF f at server v is $b_{f,v}$. Then, $y_{f,v}(t)$ must be able to meet the processing requirement at server v for VNF f, i.e.,

$$\sum_{l=1}^{\mathcal{L}} \sum_{i:f_i^l=f} \overline{x}_{l,f_i^l,v}(t) \leq y_{f,v}(t)b_{f,v}, \text{for all } f, v \text{ and } t. \quad (12)$$

Further, resources (e.g., CPU, memory, cache) in each server v must be able to support all VNFs placed there [42]. Assume that each unit-size function f needs $z_{f,v}$ amount of resources in server v. Then, we have that

$$\sum_{f=1}^{\mathcal{F}} y_{f,v}(t)z_{f,v} \leq Z_v, \text{ for all servers } v \text{ and time } t, \quad (13)$$

where Z_v is the total amount of resources available in server v. Finally, we may impose ramp constraints (2) on the changes in routing decisions. Specifically, when we migrate one VNF from a server to another, or reroute traffic from one VNF instance to another, there is additional overhead in migrating the "state" of the VNF [19], [26]. If a limited amount of network bandwidth is reserved for state migration, such migration (which corresponds to changes in $x_{l,f,v,f',v'}(t)$ and $y_{f,v}(t)$) cannot occur too fast. Such constraints can be written as

$$\left|x_{l,f_i^l,v,next(f_i^l|\mathrm{SC}^l),v'}(t) - x_{l,f_i^l,v,next(f_i^l|\mathrm{SC}^l),v'}(t-1)\right| \leq \Delta^X_{f_i^l,v,next(f_i^l|\mathrm{SC}^l),v'}, \text{ for all } l,i,v,v',t, \quad (14a)$$

$$\left|y_{f,v}(t) - y_{f,v}(t-1)\right| \leq \Delta^Y_{f,v}, \text{ for all } f,v,t. \quad (14b)$$

(iii) Costs: There are two types of costs. First, there are costs for resource consumption of running VNFs, and distance costs for routing the traffic flow among VNFs in different servers, i.e.,

$$\Phi(\vec{X}(1:\mathcal{T}), \vec{Y}(1:\mathcal{T})) = \sum_{t=1}^{\mathcal{T}}\sum_{v=1}^{\mathcal{V}}\sum_{f=1}^{\mathcal{F}} c_{f,v}(t)y_{f,v}(t) + \sum_{t=1}^{\mathcal{T}}\sum_{l=1}^{\mathcal{L}}\sum_{\substack{v'=1\\v'\neq v}}^{\mathcal{V}}\sum_{v=1}^{\mathcal{V}}\sum_{i=0}^{|\mathrm{SC}^l|} d_{v',v}x_{l,f_i^l,v,next(f_i^l|\mathrm{SC}^l),v'}(t). \quad (15)$$

where $c_{f,v}(t)$ is the unit cost of hosting one unit of VNF f in server v at time t, $d_{v',v}$ is the unit distance (e.g., latency) cost of routing a unit of traffic load from vertex v' to another vertex v. This is the in-stage cost $C_t(\cdot)$ as in the OCO model we discussed in Sec. II-A. Second, there is overhead for changes in instantiation and state migration, i.e.,

$$\Psi(\vec{X}(0), \vec{X}(1{:}\mathcal{T}), \vec{Y}(0), \vec{Y}(1{:}\mathcal{T})) = \sum_{t=1}^{\mathcal{T}}\sum_{v=1}^{\mathcal{V}}\sum_{f=1}^{\mathcal{F}} \beta_{1,f,v}\left|y_{f,v}(t) - y_{f,v}(t-1)\right| + \sum_{t=1}^{\mathcal{T}}\sum_{l=1}^{\mathcal{L}}\sum_{\substack{v'=1\\v'\neq v}}^{\mathcal{V}}\sum_{v=1}^{\mathcal{V}}\sum_{i=0}^{|\mathrm{SC}^l|} \beta_{2,f_i^l,v,next(f_i^l|\mathrm{SC}^l),v'} \cdot\left|x_{l,f_i^l,v,next(f_i^l|\mathrm{SC}^l),v'}(t) - x_{l,f_i^l,v,next(f_i^l|\mathrm{SC}^l),v'}(t-1)\right|, \quad (16)$$

where $\beta_{1,f,v}$ is the switching-cost coefficient for the overhead in instantiating and tearing-down VNF instances, and $\beta_{2,f',v',f,v}$ is the switching-cost coefficients for state migration in rerouting. This corresponds to the switching cost $\vec{\beta}^{\mathsf{T}}|\vec{X}(t) - \vec{X}(t-1)|$ as in the OCO model we discussed in Sec. II-A.

Hence, the network function placement and traffic routing in NFV can be formulated as following,

$$\min_{\substack{\{\vec{X}(0:\mathcal{T}),\\ \vec{Y}(0:\mathcal{T})\}}} \left\{\Phi(\vec{X}(0{:}\mathcal{T}), \vec{Y}(0{:}\mathcal{T})) + \Psi(\vec{X}(0{:}\mathcal{T}), \vec{Y}(0{:}\mathcal{T}))\right\}$$
$$\text{sub. to: } (10), (11), (12), (13), (14),$$
$$\vec{X}(t) \geq 0, \vec{Y}(t) \geq 0, \vec{Y}(t) \in \mathbb{Z}, \text{ for all time } t. \quad (17)$$

In summary, we can see that the NFV orchestration and scaling problem can be effectively formulated by the OCO

model we studied. The in-stage cost $C_t(\vec{X}(t), \vec{A}(t))$ can model the costs in (15). The switching cost $\vec{\beta}^{\mathrm{T}}|\vec{X}(t) - \vec{X}(t-1)|$ can model the costs in (16). The demand-supply balance constraint (1) can model the constraints (10)-(13). The ramp constraint (2) can model the constraints (14a)-(14b). Furthermore, the possible incoming rates $a_l(t)$ of the traffic flows can be modeled by an uncertainty set $\mathcal{U}$ that is formulated by constraints (3)-(5). In the next section, we will show how to obtain online algorithms that can utilize the uncertainty set $\mathcal{U}$ to optimize the competitive ratios for this type of OCO problems.

III. A Computationally Tractable Framework

In this section, we introduce a computationally tractable framework to attain small competitive ratios for general OCO problems with linear in-stage costs, switching costs and ramp constraints. We note that this type of multi-stage online decision problem in general can be intractable. Indeed, each decision $\vec{X}(t)$ can be an arbitrary function of the past inputs $\vec{A}(1:t)$, and searching over such a large functional space incurs high complexity when the problem size is large [27, p. 364], [29]. Moreover, none of the existing approaches can handle switching costs, demand-supply balance constraints and ramp constraints simultaneously. Thus, to make progress, in the following we restrict our attention to the set of affine policies [27, p. 364], [29].

Specifically, in our proposed Robust Affine Policy (RAP), we restrict $\vec{X}(t)$ to be an affine function of $\vec{A}(t)$, i.e.,

$$\vec{X}(t) = \vec{\eta}(t) + \boldsymbol{H}(t)\vec{A}(t), \text{ for all time } t, \tag{18}$$

where $\vec{\eta}(t) \in \mathbb{R}^{N\times 1}$ and $\boldsymbol{H}(t) \in \mathbb{R}^{N\times M}$ are determined before hand. Note that once $\vec{\eta}(t)$ and $\boldsymbol{H}(t)$ are determined, the online decision (18) becomes extremely simple. Instead, the complexity moves to the pre-calculation of $\vec{\eta}(t)$ and $\boldsymbol{H}(t)$ based on the knowledge of the uncertainty set $\mathcal{U}$.

Although affine policies can be restrictive, we believe that this approach is useful for three reasons. First, based on affine policies, the online decision at each time t can be adjusted according to the new input $\vec{A}(t)$ that is just revealed. Thus, they can be more efficient than classical robust optimization where all decisions must be made ahead of time [27, p. 353], [29]. Second, affine policies can be viewed as linear approximations of more general policies. Intuitively, we can think of the "best" decision as a general function of the input. When the uncertain set is small, the uncertain input is within a neighborhood of the predicted input. In that case, a linear approximation of this general function could provide reasonable results. Thus, we expect that the affine policies would be a reasonable choice when the uncertainty set is not large. Third, the computational complexity is significantly reduced. Indeed, if we made the decision rule a little bit richer, e.g., a quadratic policy, the optimization problem would have been intractable (see the discussion at the end of *step-3* in this section).

Given $\vec{\eta}(t)$ and $\boldsymbol{H}(t)$, the cost of the online decisions can be readily calculated, which we denote by

$$C^{\mathrm{RAP}}(\vec{A}(1{:}\mathcal{T})|\vec{X}(0), \vec{\eta}, \boldsymbol{H}) = C(\vec{X}(0), \vec{X}(1{:}\mathcal{T}), \vec{A}(1{:}\mathcal{T})),$$

where $\vec{X}(1{:}\mathcal{T})$ is given by (18). However, the online decisions must still satisfy both (1) and (2). In other words, we need that

$$(1),\ (2) \text{ hold for all } \vec{A}(1{:}\mathcal{T}) \in \mathcal{U},\ \text{given } (18). \tag{19}$$

Note that once we assume the affine policy of the form (18), then (19) is a requirement for any affine policy to satisfy the hard feasibility constraints (1) and (2). In other words, (19) requires that $\vec{\eta}(t)$ and $\boldsymbol{H}(t)$ in the affine policy must be chosen such that infeasibility will never occur. If (19) cannot be satisfied by any $\vec{\eta}(t)$ and $\boldsymbol{H}(t)$, then there does not exist affine policy in the form of (18) that can satisfy the hard feasibility constraints. We can thus formulate the optimization problem for minimizing the competitive ratio as

$$\min_{\{\vec{X}(0), \vec{\eta}, \boldsymbol{H}:(19)\}} \max_{\vec{A}(1:\mathcal{T})\in\mathcal{U}} \frac{C^{\mathrm{RAP}}(\vec{A}(1{:}\mathcal{T})|\vec{X}(0), \vec{\eta}, \boldsymbol{H})}{C^{\mathrm{OPT}}(\vec{A}(1{:}\mathcal{T}))}. \tag{20}$$

Note that we will optimize the competitive ratio only within those $\vec{\eta}(t)$ and $\boldsymbol{H}(t)$ that satisfy (19).

Although affine policies have been used in [27, p. 364] and [29], using a similar approach as in the optimization problem (20) introduces new technical difficulties. Note that for each $\vec{A}(1{:}\mathcal{T}) \in \mathcal{U}$, the ratio $\frac{C^{\mathrm{RAP}}(\vec{A}(1:T)|\vec{X}(0), \vec{\eta}, \boldsymbol{H})}{C^{\mathrm{OPT}}(\vec{A}(1:\mathcal{T}))}$ is convex in $\vec{\eta}$, $\boldsymbol{H}$ and $\vec{X}(0)$. Thus, the inner maximization produces a convex objective in $\vec{\eta}$, $\boldsymbol{H}$, $\vec{X}(0)$ for outer minimization. However, it is unclear how to solve the inner maximization problem itself because it involves a ratio of convex functions. Next, we will show step-by-step how to optimize an upper bound of (20) via a tractable convex optimization problem.

Step-1: Even without considering the ratio, the numerator $C^{\mathrm{RAP}}(\vec{A}(1{:}\mathcal{T})|\vec{X}(0), \vec{\eta}, \boldsymbol{H})$ in (20) is convex in $\vec{A}(1{:}\mathcal{T})$. Maximizing a convex function is in general intractable. We resolve this issue by introducing a linear upper bound on $C^{\mathrm{RAP}}(\vec{A}(1{:}\mathcal{T})|\vec{X}(0), \vec{\eta}, \boldsymbol{H})$ (see also [31, p. 228]). Specifically, note that the only non-linearity in $C^{\mathrm{RAP}}(\vec{A}(1{:}\mathcal{T})|\vec{X}(0), \vec{\eta}, \boldsymbol{H})$ is from the switching cost $\vec{\beta}^{\mathrm{T}}|\vec{X}(t) - \vec{X}(t-1)| = \vec{\beta}^{\mathrm{T}}|\vec{\eta}(t) + \boldsymbol{H}(t)\vec{A}(t) - \vec{\eta}(t-1) - \boldsymbol{H}(t-1)\vec{A}(t-1)|$. We now introduce a new variable $\mu(t) \in \mathbb{R}$ that upper-bounds this switching cost for all $\vec{A}(1{:}\mathcal{T}) \in \mathcal{U}$, i.e.,

$$\left.\begin{aligned} \mu(t) &\geq \vec{\beta}^{\mathrm{T}}|\vec{\eta}(t) + \boldsymbol{H}(t)\vec{A}(t) - \vec{\eta}(t-1) \\ &\quad - \boldsymbol{H}(t-1)\vec{A}(t-1)|, \text{ for all } t > 1 \\ \mu(1) &\geq \vec{\beta}^{\mathrm{T}}|\vec{\eta}(t) + \boldsymbol{H}(t)\vec{A}(t) - \vec{X}(0)| \end{aligned}\right\} \overset{\text{for all}}{\vec{A}(1:\mathcal{T})\in\mathcal{U}}. \tag{21}$$

Let $\tilde{C}^{\mathrm{RAP}}(\vec{A}(1{:}\mathcal{T})|\vec{\eta}, \boldsymbol{H}, \mu) = \sum_{t=1}^{\mathcal{T}}\{C_t(\vec{X}(t), \vec{A}(t)) + \mu(t)\}$, where $\vec{X}(t)$ is given by (18). Then, $C^{\mathrm{RAP}}(\vec{A}(1{:}\mathcal{T})|\vec{X}(0), \vec{\eta}, \boldsymbol{H}) \leq \tilde{C}^{\mathrm{RAP}}(\vec{A}(1{:}\mathcal{T})|\vec{\eta}, \boldsymbol{H}, \mu)$, for all $\vec{A}(1{:}\mathcal{T}) \in \mathcal{U}$. Hence, we can obtain an upper bound of (20) by solving the following optimization problem instead

$$\min_{\{\vec{X}(0), \vec{\eta}, \boldsymbol{H}, \mu:(19),(21)\}} \max_{\vec{A}(1:\mathcal{T})\in\mathcal{U}} \frac{\tilde{C}^{\mathrm{RAP}}(\vec{A}(1{:}\mathcal{T})|\vec{\eta}, \boldsymbol{H}, \mu)}{C^{\mathrm{OPT}}(\vec{A}(1{:}\mathcal{T}))}. \tag{22}$$

Note that the numerator is now a linear function in $\vec{A}(1{:}\mathcal{T})$.

Step-2: The ratio in the inner maximization problem in (22) is usually not a concave function of $\vec{A}(1{:}\mathcal{T})$. Thus, it is still not obvious how to maximize the ratio. Using the following lemma from our earlier work [43], we now show that this

inner maximization problem can be converted to an equivalent convex problem. (Please see [43] for the proof of the lemma.)

Lemma 1: For fixed $\mathbf{B} \in \mathbb{R}^{M\times N}$, $\vec{b} \in \mathbb{R}^{M\times 1}$, $\vec{c} \in \mathbb{R}^{1\times N}$ *and* $\alpha \in \mathbb{R}$, *suppose that the following conditions are simultaneously satisfied:*

(a) $f(\vec{X})$ *is a convex function of* $\vec{X} \in \mathbb{R}^{N\times 1}$;

(b) $f(\vec{X}) > 0$ *in the constrained region of* $\mathbf{B}\vec{X} \leq \vec{b}$;

(c) There exists $\vec{X}$ *satisfying* $\mathbf{B}\vec{X} \leq \vec{b}$ *and* $\vec{c}\vec{X} + \alpha > 0$.

Then, $\sup_{\{\vec{X},y\}} \{\frac{\vec{c}\vec{X}+\alpha}{y} : y = f(\vec{X}), \mathbf{B}\vec{X} \leq \vec{b}\} = \sup_{\{\vec{X}',u\}} \{\vec{c}\vec{X}' + \alpha u : 1 \geq u f(\frac{\vec{X}'}{u}), \mathbf{B}\vec{X}' \leq \vec{b}u, u > 0\}$.

Note that the second supremum is a convex problem because $uf(\frac{\vec{X}}{u})$ is a convex function whenever $f(\vec{X})$ is convex [44, p. 89]. The result of this lemma is somewhat similar to the convex transformation of linear-fractional program [44, p. 89]. However, here the denominator is non-linear, and thus Lemma 1 is more general.

We now verify that the conditions of Lemma 1 hold for (22). For condition (a), we note that $C^{\text{OPT}}(\vec{A}(1{:}\mathcal{T}))$ is the minimum of a convex function $C(\vec{X}(0), \vec{X}(1{:}\mathcal{T}), \vec{A}(1{:}\mathcal{T}))$ over $\vec{X}(0{:}\mathcal{T})$ in a convex set. Thus, $C^{\text{OPT}}(\vec{A}(1{:}\mathcal{T}))$ is a convex function of $\vec{A}(1{:}\mathcal{T})$ [44, p. 87]. For conditions (b) and (c), $C^{\text{OPT}}(\vec{A}(1{:}\mathcal{T}))$ and $\tilde{C}^{\text{RAP}}(\vec{A}(1{:}\mathcal{T})|\vec{\eta}, \boldsymbol{H}, \mu)$ are both positive, so these conditions trivially hold. Hence, based on Lemma 1, we can convert the inner maximization of (22) to an equivalent convex optimization problem. We note that although this transformation has been used in [28], *step-1* from (20) to (22) is also crucial because otherwise the numerator of (20) is not linear and thus Lemma 1 cannot be applied.

Step-3: Note that the inner maximization of (22) can be converted to a convex program, we can then focus on the outer minimization. As we discussed earlier, the objective of the outer minimization is convex in $\vec{\eta}$, $\boldsymbol{H}$ and $\vec{X}(0)$. It remains to check its constraints. These constraints are of the form that some inequalities must hold for all $\vec{A}(1{:}\mathcal{T}) \in \mathcal{U}$. It turns out that these constraints are also convex in $\vec{\eta}$, $\boldsymbol{H}$ and $\vec{X}(0)$, and can be converted to linear constraints (See [29] for related techniques). We take one part of the constraint (19) as an example. Note that by (19), the linear inequality (1) must hold for all $\vec{A}(1{:}\mathcal{T}) \in \mathcal{U}$. For any $\vec{\eta}$, $\boldsymbol{H}$, inequality (1) for each t becomes

$$\boldsymbol{B_1}[\vec{\eta}(t) + \boldsymbol{H}(t)\vec{A}(t)] + \boldsymbol{B_2}\vec{A}(t) \leq 0, \text{ for all } \vec{A}(1{:}\mathcal{T}) \in \mathcal{U} \tag{23a}$$

$$\Leftrightarrow \max_{\vec{A}(1:\mathcal{T})\in\mathcal{U}} \{\boldsymbol{B_1}[\vec{\eta}(t) + \boldsymbol{H}(t)\vec{A}(t)] + \boldsymbol{B_2}\vec{A}(t)\} \leq 0 \tag{23b}$$

$$\Leftrightarrow \max_{\vec{A}(1:\mathcal{T})\in\mathcal{U}} \{[\boldsymbol{B_1}\boldsymbol{H}(t) + \boldsymbol{B_2}]\vec{A}(t)\} \leq -\boldsymbol{B_1}\vec{\eta}(t). \tag{23c}$$

Note that $\vec{A}(1{:}\mathcal{T}) \in \mathcal{U}$ can be written as a set of linear constraints,[1] thus the left-hand-side of (23) is of the form $\max \boldsymbol{cA}$, subject to $\boldsymbol{BA} \leq \boldsymbol{b}$, where $\boldsymbol{A}$ corresponds to $\vec{A}(1{:}\mathcal{T})$, and $\vec{\eta}$ and $\boldsymbol{H}$ enter into the matrix $\boldsymbol{c}$. By standard duality [p. 226] [44],

$$\max_{\boldsymbol{BA}\leq\boldsymbol{b}} \boldsymbol{cA} = \min_{\boldsymbol{B}^{\mathrm{T}}\boldsymbol{\lambda}\geq\boldsymbol{c}^{\mathrm{T}}} \boldsymbol{b}^{\mathrm{T}}\boldsymbol{\lambda}. \tag{24}$$

[1]Note that $\mathcal{U}$ may involve absolute values, but can still be converted to a linear form. (Please see our technical report [45] for details.)

Thus, (23) is equivalent to: there exists $\boldsymbol{\lambda}$ such that,

$$\begin{cases} \boldsymbol{b}^{\mathrm{T}}\boldsymbol{\lambda} \leq -\boldsymbol{B_1}\vec{\eta}(t), \\ \boldsymbol{B}^{\mathrm{T}}\boldsymbol{\lambda} \geq \boldsymbol{c}^{\mathrm{T}}, \end{cases} \tag{25}$$

which is a convex constraint in $\boldsymbol{\lambda}$, $\vec{\eta}(t)$ and $\boldsymbol{c}$ (i.e., $\vec{\eta}(1:\mathcal{T})$, $\boldsymbol{H}(1:\mathcal{T})$). (Please see our technical report [45] for further details.) As readers can see, here it is critical that the control decision is an affine function of the input as in (18). If we made the decision rule a little bit richer, e.g., a quadratic policy, constraints in the form of (23) will no longer be convex, which makes the whole optimization problem not convex anymore.

In summary, through the above three steps, we have obtained a convex problem (22), which can be effectively solved to obtain the optimal $\vec{\eta}^*$ and $\boldsymbol{H}^*$. (Please see Appendix A for a summary of the computational complexity.) Let $\overline{\text{CR}}$ be the optimal value of (22). Then, the competitive ratio of RAP (18) based on the optimal $\vec{\eta}^*$ and $\boldsymbol{H}^*$ is no larger than $\overline{\text{CR}}$.

We acknowledge that the above-proposed approach does not produce the optimal competitive ratio for (9) due to two reasons. First, there is a gap due to our restriction to affine policies. Second, there is another gap due to the approximation in *step-1*. Intuitively, when the size of the uncertainty set is smaller, affine policies may become a better approximation of the "best" policy, and the relaxation in *step-1* will also become tighter. Thus, we expect that the resulting performance gap will be smaller. This will be demonstrated in our simulation results in Sec. VI (see Fig. 2).

Remark 1: Although the solution approach in this section assumes continuous decision variables, it can be generalized to deal with certain integer constraints.

For example, in the case of NFV orchestration and scaling (as shown in Sec. II-D), $y_{f,v}(t)$ *must be integer values. Our basic idea is to round* $y_{f,v}(t)$ *to its ceiling* $\lceil y_{f,v}(t) \rceil$. *However, we need to control the change in the objective value and constraints after this rounding. For the capacity constraint, e.g., (13), we can replace it by* $\sum_{f=1}^{\mathcal{F}} (y_{f,v}(t) + 1) z_{f,v} \leq Z_v$. *In this way, for any continuous value of* $y_{f,v}$ *that satisfies this resource constraint, the rounded value will also satisfy the original constraint (13). On the other hand, for the switching cost* $\beta_{1,f,v}|y_{f,v}(t) - y_{f,v}(t-1)|$, *we can upper bound the switching cost of the rounded decision by* $\beta_{1,f,v}(|y_{f,v}(t) - y_{f,v}(t-1)| + 1)$. *These changes can be made in the numerator of (22). However, the denominator of (22) can still take continuous values of* $y_{f,v}(t)$ *because such relaxation produces a lower bound on the true optimal offline cost.*

In summary, such changes still allow us to compute an optimized competitive ratio even when $y_{f,v}(t)$ *is constrained to be an integer. We believe that when the value of* $y_{f,v}(t)$ *is large, the loss due to the rounding will be small. This is particularly relevant with the more recent container technology [46], when the number of containers in each server can be large.*

IV. Algorithm Robustification

In Sec. III, we developed a tractable computational framework to calculate an optimized competitive ratio $\overline{\text{CR}}$ among the class of affine policies. Let the corresponding Robust Affine Policy (RAP) be denoted by π^*_{RAP}, which will attain

a competitive ratio no larger than $\overline{\mathrm{CR}}$. However, as is often the case with competitive online algorithms in the literature, the policy π^*_{RAP} may be too conservative in nature. For example, consider the scenario where the uncertainty set $\mathcal{U}$ is given by a predicted input trajectory plus/minus possible errors. In order for π^*_{RAP} to attain the competitive ratio $\overline{\mathrm{CR}}$, it must "defend" against the worst case where the input is far away from the prediction. Specifically, it may have to over-provision resources. As a result, if the input is actually very close to the prediction (which usually corresponds to a larger probability mass on average), π^*_{RAP} may incur a higher cost than necessary. In contrast, a popular method in the literature to deal with sequential decisions under uncertainty is RHC (Receding Horizon Control) [6]. At each time t, RHC assumes that the future demand is exactly the same as the most-recent near-term prediction, which is based on revealed demand, day-ahead prediction and possible constraints, e.g., (3), (4), (5). Then, RHC minimizes the cost over the entire future horizon and commits to the first decision $\vec{X}(t)$. Intuitively, if the input is close to the prediction (which we refer to as the average case), RHC may actually perform very well. The problem, of course, is that RHC cannot guarantee as low a competitive ratio as $\overline{\mathrm{CR}}$. In summary, we see a dilemma between worst-case and average-case performance. In this section, we will address this dilemma by significantly generalizing the "robustification" procedure of our earlier work [28] to obtain good performance for both worst-case and average-case inputs.

In our proposed robustification procedure, we begin with an online algorithm π_0 that is believed to achieve good average-case performance (e.g., π_0 could be a variant of RHC from [6]). We are also given the competitive ratio $\overline{\mathrm{CR}}$, which is optimized among the class of affine policies as in Sec. III. We aim to produce a new online policy π that attains comparable average-case performance as π_0, but at the same time the worst-case competitive ratio $\overline{\mathrm{CR}}$. Our basic idea for this new policy π is to follow the decisions of π_0 as much as possible, unless doing so will violate the competitive ratio $\overline{\mathrm{CR}}$. Our first step is thus to develop a way to check whether the decision of π_0 will violate the competitive ratio $\overline{\mathrm{CR}}$.

Toward this end, let us focus on a time t. Note that the decisions of this algorithm π before time t have already been made. The algorithm π_0 now produces a decision $\vec{X}^{\pi_0}(t)$ for time t. In order to verify whether this new decision will still attain $\overline{\mathrm{CR}}$, we need to check whether the following holds:

$$C^{\pi}(\vec{A}(1{:}\mathcal{T})) \leq \overline{\mathrm{CR}} \cdot C^{\mathrm{OPT}}(\vec{A}(1{:}\mathcal{T})), \quad \text{for all } \vec{A}(t+1:\mathcal{T}) \in \mathcal{U}_{|\vec{A}(1:t)}, \tag{26}$$

where $\mathcal{U}_{|\vec{A}(1:t)}$ is given in (6). Let $C^{\pi}(\vec{A}(1:t-1))$ denote the past cost of the online algorithm π from time 1 to $t-1$, excluding the switching cost from time $t-1$ to t. (Again, this cost is known at time t, regardless of whether or not π has followed π_0 before time t.) Based on the decision $\vec{X}^{\pi_0}(t)$ of algorithm π_0, let $C^{\pi_0}(\vec{A}(t)) = C_t(\vec{X}^{\pi_0}(t), \vec{A}(t)) + \vec{\beta}^{\mathrm{T}}|\vec{X}^{\pi_0}(t) - \vec{X}^{\pi}(t-1)|$. Further, let $C^{\pi}(\vec{A}(t+1:\mathcal{T}))$ denote the future cost from time $t+1$ to time $\mathcal{T}$ (including the switching cost from $\vec{X}^{\pi_0}(t)$ to $\vec{X}^{\pi}(t+1)$). Then, because we want to follow the decision $\vec{X}^{\pi_0}(t)$, (26) is equivalent to checking that

$$C^{\pi}(\vec{A}(1:t-1)) + C^{\pi_0}(\vec{A}(t)) + C^{\pi}(\vec{A}(t+1:\mathcal{T})) - \overline{\mathrm{CR}} \cdot C^{\mathrm{OPT}}(\vec{A}(1{:}\mathcal{T})) \leq 0, \text{for all } \vec{A}(t+1:\mathcal{T}) \in \mathcal{U}_{|\vec{A}(1:t)}. \tag{27}$$

However, the difficulty of checking (27) is that not only the future input has not been revealed yet, we do not even know what decision the algorithm π will take on these future inputs! To circumvent this difficulty, we estimate $C^{\pi}(\vec{A}(t+1:\mathcal{T}))$ based on affine policies. In this case, the affine policy can be written as

$$\vec{X}(t') = \vec{\eta}(t') + \boldsymbol{H}(t')\vec{A}(t'),\ t' = t+1, t+2, \ldots, \mathcal{T}. \tag{28}$$

Note that in general this pair of $(\vec{\eta}, \boldsymbol{H})$ may be different from those calculated in the previous section. Let $C^{\mathrm{RAP}}(\vec{A}(t+1:\mathcal{T})|\vec{\eta}, \boldsymbol{H})$ denote the future cost $C^{\pi}(\vec{A}(t+1:\mathcal{T}))$ if π follows the affine policy (28). We can then formulate the following optimization problem:

$$\zeta'_{1,t} \triangleq \min_{\substack{\{\vec{\eta}, \boldsymbol{H}: (19) \text{ restricted} \\ \text{to } \vec{A}(t+1:\mathcal{T}) \in \mathcal{U}_{|\vec{A}(1:t)}\}}} \ \max_{\substack{\vec{A}(t+1:\mathcal{T}) \\ \in \mathcal{U}_{|\vec{A}(1:t)}}} \{C^{\pi}(\vec{A}(1:t-1)) + C^{\pi_0}(\vec{A}(t)) + C^{\mathrm{RAP}}(\vec{A}(t+1:\mathcal{T})|\vec{\eta}, \boldsymbol{H}) - \overline{\mathrm{CR}} \cdot C^{\mathrm{OPT}}(\vec{A}(1{:}\mathcal{T}))\}. \tag{29}$$

Similar to *step-1* of Sec. III, (29) may be intractable because the maximization part is a non-convex problem. Nonetheless, we can use the technique in *step-1* of Sec. III [see (21)] to introduce a new set of variables μ that upper-bounds the switching costs in (29). In this way, we can obtain an upper bound of $\zeta'_{1,t}$ via a convex program, given by

$$\zeta_{1,t} \triangleq \min_{\substack{\{\vec{\eta}, \boldsymbol{H}, \mu: (19), (21) \\ (31) \text{ restricted to} \\ \vec{A}(t+1:\mathcal{T}) \in \mathcal{U}_{|\vec{A}(1:t)}\}}} \ \max_{\substack{\vec{A}(t+1:\mathcal{T}) \\ \in \mathcal{U}_{|\vec{A}(1:t)}}} \{C^{\pi}(\vec{A}(1:t-1)) + C^{\pi_0}(\vec{A}(t)) + \tilde{C}^{\mathrm{RAP}}(\vec{A}(t+1:\mathcal{T})|\vec{\eta}, \boldsymbol{H}, \mu) - \overline{\mathrm{CR}} \cdot C^{\mathrm{OPT}}(\vec{A}(1{:}\mathcal{T}))\}, \tag{30}$$

where $\tilde{C}^{\mathrm{RAP}}(\vec{A}(t+1{:}\mathcal{T})|\vec{\eta}, \boldsymbol{H}, \mu) = \sum_{t'=t+1}^{\mathcal{T}} \{C_{t'}(\vec{X}(t'), \vec{A}(t')) + \mu(t')\}$, $\vec{X}(t')$ is given by (28), and the additional constraint (31) is given below by

$$\mu(t+1) \geq \vec{\beta}^{\mathrm{T}}|\vec{\eta}(t+1) + \boldsymbol{H}(t+1)\vec{A}(t+1) - \vec{X}^{\pi_0}(t)|, \quad \text{for all } \vec{A}(t+1) \in \mathcal{U}_{|\vec{A}(1:t)}. \tag{31}$$

Thus, whenever $\zeta_{1,t} \leq 0$, we must have $\zeta'_{1,t} \leq \zeta_{1,t} \leq 0$. In other words, $\zeta_{1,t} \leq 0$ guarantees that, by using π_0 at time t, there must exist an affine policy in the future such that the resulting competitive ratio is less than or equal to $\overline{\mathrm{CR}}$. Therefore, we can be assured that following the decision of the algorithm π_0 at time t will retain the competitive ratio $\overline{\mathrm{CR}}$.

We still need to determine what to do if $\zeta_{1,t} > 0$. In that case, we no longer follow the decision of algorithm π_0 at time t. Instead, we find $\vec{X}(t)$ as well as a different affine policy

based on the following optimization problem,

$$\zeta_{2,t} \triangleq \min_{\substack{\{\vec{X}(t),\vec{\eta},\boldsymbol{H},\mu:(19),\\ (21)\text{ restricted to}\\ \vec{A}(t+1:\mathcal{T})\in\mathcal{U}_{|\vec{A}(1:t)}\}}} \max_{\substack{\vec{A}(t+1:\mathcal{T})\\ \in\mathcal{U}_{|\vec{A}(1:t)}}} \{C^{\pi}(\vec{A}(1:t-1)) + C_t(\vec{X}(t),\vec{A}(t)) + \vec{\beta}^{\mathrm{T}}|\vec{X}(t)-\vec{X}^{\pi}(t-1)| + \tilde{C}^{\mathrm{RAP}}(\vec{A}(t+1{:}\mathcal{T})|\vec{\eta},\boldsymbol{H},\mu) - \overline{\mathrm{CR}}\cdot C^{\mathrm{OPT}}(\vec{A}(1{:}\mathcal{T}))\}. \quad (32)$$

Note that as long as $\zeta_{2,t} \leq 0$, by using the decision $\vec{X}(t)$ from the optimization problem (32) at time t, we can be assured that following the resulting affine policy from (32) will attain the competitive ratio $\overline{\mathrm{CR}}$. Thus, at time t, if $\zeta_{1,t} > 0$, our new algorithm π will follow $\vec{X}(t)$ from the optimization problem (32). The detailed robustification procedure is shown in Algorithm 1.

Algorithm 1 Robustification Procedure

Input: $\overline{\mathrm{CR}}$, $\mathcal{U}$, and Algorithm π_0
Output: π: Robustified version of π_0
FOR $t = 1 : \mathcal{T}$
Update $\mathcal{U}_{|\vec{A}(1:t)}$, $C^{\pi}(\vec{A}(1:t-1))$, $\vec{X}^{\pi_0}(t)$ and $C^{\pi_0}(\vec{A}(t))$.
Solve (30) to get $\zeta_{1,t}$
if $\zeta_{1,t} \leq 0$ **then**
$\vec{X}^{\pi}(t) \leftarrow \vec{X}^{\pi_0}(t)$
else
Solve (32) to get a new optimal $\vec{X}(t)$
$\vec{X}^{\pi}(t) \leftarrow \vec{X}(t)$
end if
END

Intuitively, if we can show that for all time t,

$$\text{either } \zeta_{1,t} \leq 0 \text{ or } \zeta_{2,t} \leq 0 \quad (33)$$

must hold, then Algorithm 1 will always attain the same competitive ratio $\overline{\mathrm{CR}}$. (33) can be shown by induction. Specifically, if $\zeta_{1,t+1} \leq 0$ at the next time $t+1$, (33) holds trivially for time $t+1$. On the other hand, if $\zeta_{1,t+1} > 0$, we can show $\zeta_{2,t+1} \leq 0$ from the hypothesis (33) at time t. In summary, (33) also holds at time $t+1$. We then obtain the following main result.

Theorem 2: Algorithm π from the above robustification procedure is $\overline{\mathrm{CR}}$−competitive. That is, for all $\vec{A}(1:\mathcal{T}) \in \mathcal{U}$,

$$C^{\pi}(\vec{A}(1:\mathcal{T})) \leq \overline{\mathrm{CR}} \cdot C^{OPT}(\vec{A}(1:\mathcal{T})). \quad (34)$$

Please see Appendix B for the complete proof of Theorem 2. Theorem 2 shows that the algorithm π from the robustification procedure attains the optimized competitive ratio $\overline{\mathrm{CR}}$. Our hope is that it will also attain good average-case performance for the following reason. Note that for a reasonable prediction, the "average case" corresponds to the actual input being close to the prediction, while the "worst case" corresponds to the actual input being far from the prediction. We expect that, when the input is close to prediction, the algorithm π_0 will likely do well, and it would also be easier for the online algorithm to beat the "worst case" competitive ratio. Thus, we expect that the robustified algorithm π will likely follow π_0. Only when the input is far from the prediction (i.e., the worst case), the robustified algorithm will invoke different decisions. In this way, we hope that the resulting algorithm will perform well for both average-case and worst-case inputs. This behavior will be further demonstrated using simulation results in Sec. VI (see Fig. 3).

V. RAP and Robustification With Memory

In (18), the decision $\vec{X}(t)$ of the Robust Affine Policy (RAP) is only an affine function of the current input $\vec{A}(t)$. We can extend this idea to the case when $\vec{X}(t)$ can be based on past inputs within a certain memory range. Specifically, for the Robust Affine Policy with memory (RAP-mem), we restrict $\vec{X}(t)$ to be an affine function of $\vec{A}(s)$ from a past time $s = t - t_{\mathrm{mem}}$ to the current time $s = t$, i.e.,

$$\vec{X}(t) = \vec{\eta}(t) + \sum_{s=t-t_{\mathrm{mem}}}^{t} \boldsymbol{H}_t(s)\vec{A}(s), \text{ for all time } t, \quad (35)$$

where t_{mem} is the memory size, $\vec{\eta}(t) \in \mathbb{R}^{N\times 1}$ and $\boldsymbol{H}_t(s) \in \mathbb{R}^{N\times M}$ are determined before hand. Then, all the techniques from *step-1* to *step-3* can still be applied. Therefore, we can transfer the optimization problem (20) using the new affine policy (35) to a convex optimization problem. Intuitively, as the memory size t_{mem} increases, the performance of RAP-mem should be better. (Please see Fig. 4b in Sec. VI for the numerical results.) However, the computational complexity also increases linearly in t_{mem}. (Please see Appendix A for the computational complexity.)

Finally, it is also easy to extend the robustification procedure to the case with memory. We only need to apply the new affine policy (35) to (31) and when calculating the cost of RAP in (29), (30) and (32).

VI. Evaluation

In this section, we first conduct simulations to evaluate the competitiveness of our proposed Robust Affine Policy (RAP). Then, under both average and worst cases, we compare the performance of RAP, RHC (which is an existing algorithm with good average-case performance), and its robustified version (by applying our proposed robustification procedure). At last, we illustrate the impact of the switching costs on the optimized competitive ratio.

A. Simulation Setup

We use the NFV orchestration and scaling problem in Sec. II-D as a case study to evaluate the performance of our proposed competitive online algorithm RAP and the robustification procedure. Since there is little public data on NFV topologies and traces, we borrow the topology of the physical network of Cogent [47] to create the NFV network. There are 50 servers ($\mathcal{V} = 50$) distributed in multiple data centers. The distance costs $d_{v',v}$ among the servers are proportional to the geographical distances (given in [47]) and scaled by a random value in the interval $[0.5, 1.5]$, so that we can evaluate how the performance of the algorithms changes with different problem parameters. Moreover, there are 30 different ($\mathcal{L} = 30$) traffic flows. The service chain that each flow requires is an

TABLE III
SIMULATION PARAMETERS

Time horizon	$\mathcal{T}$	50
Resources needed for a unit-size VNF f on server s	$z_{f,v}$	$U[0,1]$
Resources available on server v	Z_v	5000
Processing capacity of a unit-size VNF f on server v	$b_{f,v}$	$U[1,2]$
Cost of hosting resources consumed by a unit-size VNF f on server v	$c_{f,v}$	$U[0,20]$
Ramp constraint	$\Delta^X_{f',v',f,v}$	$U[5,10]$
Changing speed of demand	Δ^A_l	50
Budget constraint	Γ	15

ordered subset of 5 VNFs ($\mathcal{F} = 5$). Other parameters[2] are listed in Table III. $U[a,b]$ denotes the uniform distribution in the interval $[a,b]$.

In order to evaluate the online algorithms, we first use a scaled version of the traffic load trajectory from the HP trace [48] to generate the day-ahead prediction $\vec{A}^{\text{DAP}}(1:\mathcal{T})$. (See Fig. 1.) We then define the uncertainty set $\mathcal{U}$ around the day-ahead prediction $\vec{A}^{\text{DAP}}(1:\mathcal{T})$ based on the model in (3)-(5), where the parameters Δ^A_l and Γ are given in Table III, and the parameter $\epsilon_l(t)$ will be varied in our simulation later. In order to evaluate the algorithms with online input, we generate the real demand $\vec{A}^{\text{real}'}(1{:}\mathcal{T})$ around $\vec{A}^{\text{DAP}}(1{:}\mathcal{T})$ by adding i.i.d white Gaussian noise with variance $[\sigma_l(t)]^2 = [\epsilon_l(t) \cdot a_l^{\text{DAP}}(t) \cdot \rho]^2$, where we call ρ the "variability" of the demand sequence. Note that when ρ is large, the demand sequence fluctuates more significantly in time and may even exceed the range defined by the uncertainty set $\mathcal{U}$. When that happens, we further change $\vec{A}^{\text{real}'}(1{:}\mathcal{T})$ to the closest value $\vec{A}^{\text{real}}(1:\mathcal{T})$ that satisfies the constraints of $\mathcal{U}$. Specifically, we find $\vec{A}^{\text{real}}(1:\mathcal{T})$ with the minimal ℓ_2-distance from $\vec{A}^{\text{real}'}(1:\mathcal{T})$ that satisfies the constraints of the uncertainty set. We then use $\vec{A}^{\text{real}}(t)$ as the online input to test our algorithms.

We compare our proposed competitive online algorithm with both the optimal offline solution and Receding Horizon Control (RHC). Receding Horizon Control, also known as Model Predictive Control (MPC), is a popular framework in the literature to make sequential control decisions based on imperfect prediction [5], [8], [26], [30], [31]. Specifically, at each time t, RHC optimizes the total costs in a look-ahead window of size K based on the predicted inputs, and then only commits to the first decision $\vec{X}(t)$. As shown in [6], RHC is often found to exhibit good average-case performance. Thus, we can use it as the online algorithm π_0, which we will robustify through the robustification procedure. Furthermore, note that the optimal offline solution is assumed to know $\vec{A}^{\text{real}}(1{:}\mathcal{T})$ in advance. In contrast, at each time t, our proposed policies and RHC only know the real demand up to time t, as well as the predicted demand in $\mathcal{U}_{|\vec{A}(1:t)}$. As we mentioned in Sec. I, RHC may lead to infeasibility due to the hard feasibility constraints. Here, whenever RHC finds no feasible solution at time t, we allow it to violate the ramp constraint (2) by paying another high penalty of 10^2 for each unit of violation

[2]Note that with container technology (such as Docker [21]), the granularity of VNF resource allocation is much finer than with VMs. Hence, we use a large value of 5000 for the server capacity.

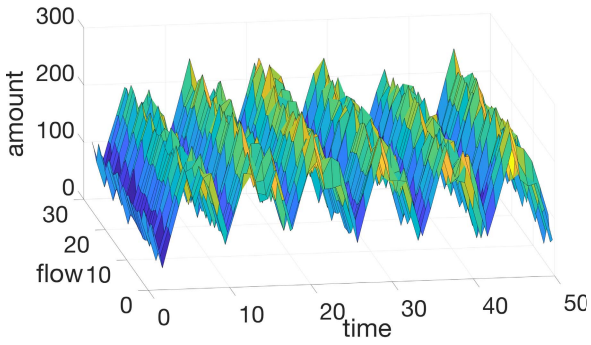


Fig. 1. Day-ahead prediction of the amount of the traffic flow.

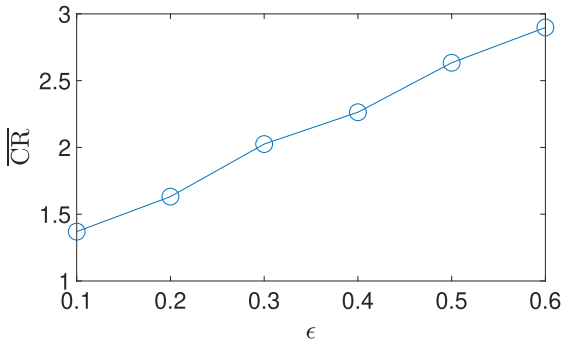


Fig. 2. The competitive ratio $\overline{\text{CR}}$ vs. the uncertain level ϵ.

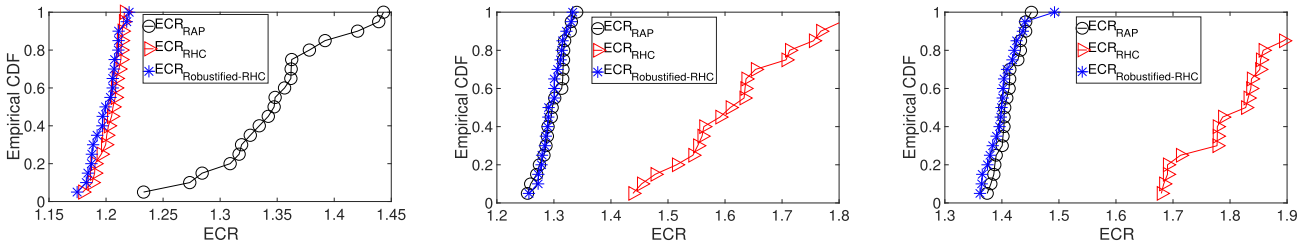
of the ramp constraint. Note that our proposed algorithms never pay this penalty because they always respect the ramp constraints.

B. Simulation Results

In Fig. 2, we report the competitive ratio $\overline{\text{CR}}$ of our proposed robust affine policy π^*_{RAP}. The switching-cost coefficients β are uniformly chosen from $[0,10]$. As we vary the uncertainty level ϵ of $\mathcal{U}$, the competitive ratio increases almost linearly. Note that even when $\epsilon = 0.6$, i.e., the real demand may vary 60% from the predicted value, the competitive ratio is around 2.898, which is relatively small.

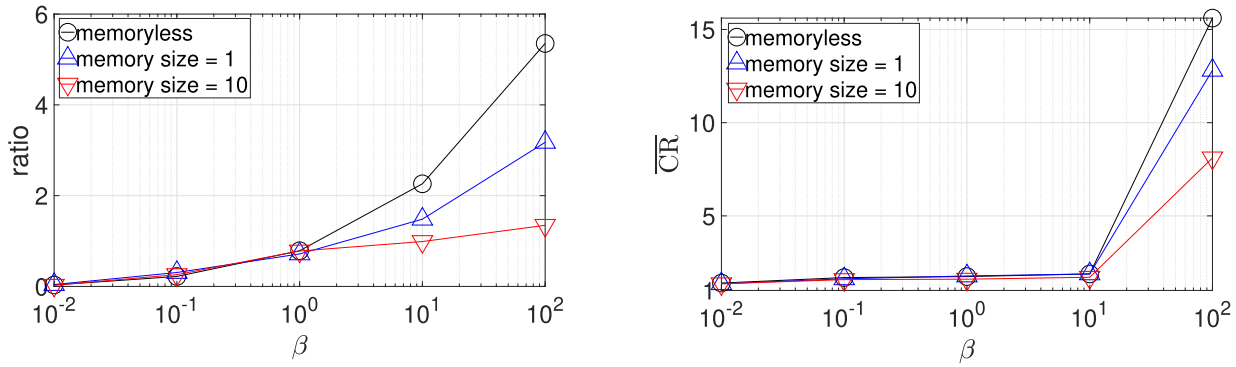
The value of $\overline{\text{CR}}$ reported above is the theoretical upper bound of the competitive ratio over all inputs. We also collect the empirical competitive ratio (ECR) under the random demand trajectory that we generated, which is the ratio of the total cost of an online algorithm π to that of the optimal offline solution for each generated trajectory.

We plot in Fig. 3 the empirical CDF (Cumulative Distribution Function) of the value of ECR over 20 trials, where ECR_{RAP}, ECR_{RHC} and $\text{ECR}_{\text{Robustified-RHC}}$ correspond to the ECRs of RAP, RHC and the robustified version of RHC, respectively. We first observe that at all values of variability ρ, the ECR of our proposed robustified-RHC algorithm never exceeds the value of $\overline{\text{CR}}$ (which is 1.632 when $\epsilon = 0.2$). In contrast, the ECR of RHC increases as ρ increases, and it exceeds $\overline{\text{CR}}$ for a significant fraction of trials when $\rho = 1$ and $\rho = 20$. Clearly, RHC fails to control the worst-case competitive ratio when the future demand is highly variable. Specifically, due to the ramp constraint, such high variability may lead to infeasibility for RHC, which produces the high online costs. On the other hand, π^*_{RAP} incurs much higher



(a) $\rho = 1/30$ (b) $\rho = 1$ (c) $\rho = 20$

Fig. 3. The empirical CDFs of ECRs (empirical competitive ratios) by varying the variability $\rho = \frac{1}{30}, 1, 20$.



(a) The ratio $\frac{\text{switching cost}}{\text{in-stage cost}}$ vs. the switching-cost coefficient β. (b) The competitive ratio $\overline{\text{CR}}$ vs. the switching-cost coefficient β.

Fig. 4. Impact of the switching-cost coefficient β.

ECR than both robustified-RHC and RHC when $\rho = \frac{1}{30}$. This suggests that RAP is too conservative when the prediction turns out to be quite accurate. In comparison to both RHC and RAP, what is particularly appealing for our proposed robustified-RHC policy is that it not only attains much smaller ECR than RHC when ρ is large, but also attains almost the same performance as RHC when ρ is small.

In Fig. 4, we plot how $\overline{\text{CR}}$ varies with the magnitude of the switching costs. Intuitively, when the switching-cost coefficient β is large, the online decisions become more difficult, and thus the competitive ratio will increase. This is shown in Fig. 4. For example, when $\beta = 10$, Fig. 4a shows that the total amount of switching cost (16) is almost twice the in-stage cost (15). From Fig. 4b, the corresponding competitive ratio increases to about 1.885. However, when β further increases, the switching cost dominates (See Fig. 4a), and our competitive ratio further increases. We note that for certain online problems (e.g., [10]), constant competitive ratios may be obtained even when the switching cost is arbitrarily high. This suggests that there may be room to improve our online algorithms, e.g., by integrating ideas from ski-rental problems [7], [10], to obtain even better competitive ratios.

Recall from Sec. V that the idea of RAP can be extended from (18) to allow for memory. In Fig. 4, we also compare the $\overline{\text{CR}}$ of the memoryless RAP and that of RAP with different memory size t_{mem}. From Fig. 4b, we can see that as the memory size increases, $\overline{\text{CR}}$ will decrease, especially when the switching-cost coefficient β is large. In particular, when $\beta = 100$, the $\overline{\text{CR}}$ of RAP with a memory size $t_{\text{mem}} = 10$ is only a half of the $\overline{\text{CR}}$ of memoryless RAP. Moreover, in Fig. 4a, we can see that RAP with memory also pays lower switching costs.

VII. Conclusion

We study competitive online algorithms for OCO problems with linear in-stage costs, switching costs and ramp constraints. First, we present a powerful computational framework to obtain an optimized competitive ratio given an uncertainty set. Second, we provide a robustification procedure to obtain robustified online algorithms with both good average-case performance and an optimized competitive ratio. We demonstrate the power of our proposed approach through a case study for NFV. The robustified version of a popular heuristic algorithm RHC is shown to attain good performance for both average-case and worst-case inputs. For future work, we plan to study matching lower bounds for the optimal competitive ratio and compare that with ours.

Appendix A
Computational Complexity

In Sec. III, we have shown that the optimization problem (20) can be converted to a convex optimization problem (22). We can then use the Interior Point Method (IPM) [44, p. 561] to solve this convex problem. To calculate the subgradients that IPM needs to use in each loop of the problem, we apply two standard techniques from convex optimization [44, p. 249]. (Please see our technical report [45] for more details of solving (22).)

In this appendix, we estimate the complexity of our computational procedure to solve the problem (22). The solution to this convex problem (22) involves three loops, one

for solving the offline optimization problem to get $C^{\text{OPT}}(\cdot)$, one for solving the inner maximization problem over the input variables $\vec{A}(1:\mathcal{T})$, and one for solving the eventual outer minimization problem over the affine decision variables $\vec{\eta}(1:\mathcal{T})$ and $\boldsymbol{H}(1:\mathcal{T})$ as well as the auxiliary variables $\mu(1:\mathcal{T})$.

For the purpose of estimating the complexity, assume that we use K_1, K_2 and K_3 to denote the number of iterations to solve $C^{\text{OPT}}(\cdot)$, the inner maximization problem and the outer minimization problem, respectively. In each iteration of solving $C^{\text{OPT}}(\cdot)$, it takes $O(N\mathcal{T})$ elementary operations to update the variables $\vec{X}(1:\mathcal{T})$, where N is the dimension of $\vec{X}(t)$ at each time t and $\mathcal{T}$ is the total time length. In each iteration of solving the inner maximization problem, it takes $O(M\mathcal{T})$ elementary operations to update the variables $\vec{A}(1:\mathcal{T})$ and u, where M is the dimension of $\vec{A}(t)$ at each time t. In each iteration of solving the outer minimization problem, it takes $O(N\mathcal{T})$ elementary operations to update the variables $\vec{\eta}(1:\mathcal{T})$, $\boldsymbol{H}(1:\mathcal{T})$ and $\mu(1:\mathcal{T})$. Hence, the overall complexity of our computational procedure to solve the convex optimization problem (22) is $O(((N\mathcal{T}K_1+M\mathcal{T})K_2+N\mathcal{T})K_3)$.

Finally, when applying the affine policy with memory, i.e., (35), in each iteration of solving the outer minimization problem, it takes $O(N\mathcal{T}t_{\text{mem}})$ more elementary operations to update the new variables in (35). Then, the overall complexity of our computational procedure to solve the obtained convex optimization problem becomes $O(((N\mathcal{T}K_1+M\mathcal{T})K_2+N\mathcal{T}t_{\text{mem}})K_3)$. Hence, the computational complexity increases linearly in t_{mem}.

APPENDIX B
PROOF OF THEOREM 2

We start from two lemmas that capture the significance of having $\zeta_{1,t}\le 0$ and $\zeta_{2,t}\le 0$.

Lemma 3: For any t, let $\vec{\eta}^*_{1,t}$, $\boldsymbol{H}^*_{1,t}$ and $\mu^*_{1,t}$ be the optimal solution to (30) at time t. If $\zeta_{1,t}\le 0$, we must have that

$$C^{\pi}(\vec{A}(1{:}t-1))+C^{\pi_0}(\vec{A}(t))+C^{RAP}(\vec{A}(t+1{:}\mathcal{T})|\vec{\eta}^*_{1,t},\boldsymbol{H}^*_{1,t}) \le \overline{\text{CR}}\cdot C^{OPT}(\vec{A}(1:\mathcal{T})),\ \textit{for all}\ \vec{A}(t+1:\mathcal{T})\in\mathcal{U}_{|\vec{A}(1:t)}. \tag{36}$$

Further, the RAP using $\vec{\eta}^*_{1,t}$ and $\boldsymbol{H}^*_{1,t}$ meets constraints (19) and (21) for all $\vec{A}(t+1:\mathcal{T})\in\mathcal{U}_{|\vec{A}(1:t)}$.

Lemma 3 implies that, as long as $\zeta_{1,t}\le 0$, we can be assured that following the decision of the algorithm π_0 at time t will retain the competitive ratio $\overline{\text{CR}}$.

Proof: Since $\zeta_{1,t}\le 0$, at the optimal solution $\vec{\eta}^*_{1,t}$, $\boldsymbol{H}^*_{1,t}$ and $\mu^*_{1,t}$, the maximum for the inner loop of (30) must be no greater than 0. Hence, for all $\vec{A}(t+1:\mathcal{T})\in\mathcal{U}_{|\vec{A}(1:t)}$, we must have that

$$\begin{aligned}&C^{\pi}(\vec{A}(1:t-1))+C^{\pi_0}(\vec{A}(t))\\&\quad+\widetilde{C}^{\text{RAP}}(\vec{A}(t+1:\mathcal{T})|\vec{\eta}^*_{1,t},\boldsymbol{H}^*_{1,t},\mu^*_{1,t})\\&\le\overline{\text{CR}}\cdot C^{\text{OPT}}(\vec{A}(1:\mathcal{T})).\end{aligned}$$

Since $\widetilde{C}^{\text{RAP}}(\vec{A}(t+1:\mathcal{T})|\vec{\eta}^*_{1,t},\boldsymbol{H}^*_{1,t},\mu^*_{1,t})\ge C^{\text{RAP}}(\vec{A}(t+1:\mathcal{T})|\vec{\eta}^*_{1,t},\boldsymbol{H}^*_{1,t})$ for all $\vec{A}(t+1:\mathcal{T})\in\mathcal{U}_{|\vec{A}(1:t)}$, (36) then holds. The last part of the lemma follows from the formulation of the constraints of (30). □

Following the same idea as we shown in the proof of Lemma 3 above, we can obtain Lemma 4 below.

Lemma 4: For any t, let $\vec{X}^*_{2,t}(t)$, $\vec{\eta}^*_{2,t}$, $\boldsymbol{H}^*_{2,t}$ and $\mu^*_{2,t}$ be the optimal solution to (32) at time t. If $\zeta_{2,t}\le 0$, we must have

$$\begin{aligned}&C^{\pi}(\vec{A}(1{:}t-1))+C_t(\vec{X}^*_{2,t}(t),\vec{A}(t))\\&\quad+\vec{\beta}^T|\vec{X}^*_{2,t}(t)-\vec{X}^{\pi}(t-1)|\\&\quad+C^{RAP}(\vec{A}(t+1:\mathcal{T})|\vec{\eta}^*_{2,t},\boldsymbol{H}^*_{2,t})\\&\le\overline{\text{CR}}\cdot C^{OPT}(\vec{A}(1:\mathcal{T})),\ \textit{for all}\ \vec{A}(t+1:\mathcal{T})\in\mathcal{U}_{|\vec{A}(1:t)}.\end{aligned} \tag{37}$$

Further, the RAP using $\vec{X}^*_{2,t}(t)$, $\vec{\eta}^*_{2,t}$ and $\boldsymbol{H}^*_{2,t}$ meets constraints (19) and (21) for all $\vec{A}(t+1:\mathcal{T})\in\mathcal{U}_{|\vec{A}(1:t)}$.

Lemma 4 implies that, as long as $\zeta_{2,t}\le 0$, by using the decision $\vec{X}(t)$ from the optimization problem (32) at time t, we can be assured that following the resulting affine policy from (32) will attain the competitive ratio $\overline{\text{CR}}$.

The next lemma is crucial for the overall proof.

Lemma 5: For any t, $\zeta_{1,t}\ge\zeta_{2,t}\ge\zeta_{2,t+1}$.

The significance of Lemma 5 is that it allows us to use induction to prove Lemma 6 below.

Proof: (i) To prove $\zeta_{1,t}\ge\zeta_{2,t}$, note that the difference between $\zeta_{1,t}$ and $\zeta_{2,t}$ is the term for the current cost. If we use $\vec{X}(t)=\vec{X}^{\pi_0}(t)$ in (32), the objective value of (32) would be equal to $\zeta_{1,t}$. Since (32) further minimizes over $\vec{X}(t)$, the objective value can only be lower. Hence, $\zeta_{1,t}\ge\zeta_{2,t}$.

(ii) To prove $\zeta_{2,t}\ge\zeta_{2,t+1}$, note that the difference between $\zeta_{2,t}$ and $\zeta_{2,t+1}$ is the term for the cost at time $t+1$. If we use $\vec{X}(t+1)=\vec{\eta}^*_{2,t}(t+1)+\boldsymbol{H}^*_{2,t}(t+1)\vec{A}(t+1)$, where $\vec{\eta}^*_{2,t}(t+1)$ and $\boldsymbol{H}^*_{2,t}(t+1)$ are the optimal solutions to (32) at time t, the objective value of (32) at time $t+1$ would be equal to $\zeta_{2,t}$. Since at time $t+1$, (32) further minimizes over $\vec{X}(t+1)$, the objective value can only be lower. Hence, $\zeta_{2,t}\ge\zeta_{2,t+1}$. □

Lemma 6: Either $\zeta_{1,t}\le 0$ or $\zeta_{2,t}\le 0$ must hold for all time $t=1,2,\ldots,\mathcal{T}$.

Lemma 6 implies that, following the robustification procedure (i.e., Algorithm 1), there always exists a robust affine policy in the future, such that the optimized competitive ratio $\overline{\text{CR}}$ is guaranteed.

Proof: We will prove Lemma 6 by mathematical induction.

a. Base case: $t=1$. We divide into two sub-cases.

(a-i) If $\zeta_{1,1}\le 0$, the statement of the lemma holds trivially for time $t=1$.

(a-ii) If $\zeta_{1,1}>0$, Algorithm 1 will set $\vec{X}^{\pi}(1)=\vec{X}^*_{2,1}(1)$, where $\vec{X}^*_{2,1}(1)$ is the optimal solution to (32) at time $t=1$. Next, we show that there exists a robust affine policy in the future, such that $\zeta_{2,1}\le 0$. Note that if we let $\vec{\eta}^*(1:\mathcal{T})$, $\boldsymbol{H}^*(1:\mathcal{T})$ and $\mu^*(1:\mathcal{T})$ be the optimal solution to (22). Then, at time $t=1$, by letting $\vec{X}(1)=\vec{\eta}^*(1)+\boldsymbol{H}^*(1)\vec{A}(1)$, $\vec{\eta}(t')=\vec{\eta}^*(t')$, $\boldsymbol{H}(t')=\boldsymbol{H}^*(t')$ and $\mu(t')=\mu^*(t')$ for $t'=2,\ldots,\mathcal{T}$, we obtain a feasible solution to (32) at time $t=1$. The objective of (32) at this feasible solution must be no greater than 0 because the objective value of (22) is equal to $\overline{\text{CR}}$ at $\vec{\eta}^*(1:\mathcal{T})$, $\boldsymbol{H}^*(1:\mathcal{T})$ and $\mu^*(1:\mathcal{T})$. Since (32)

further minimizes its objective value, we have $\zeta_{2,1} \leq 0$ at time $t = 1$.

Thus, the statement of the lemma holds at time $t = 1$.

b. Induction step: Suppose that the statement of the lemma holds for time $t = t_0$. We wish to show that it also holds for time $t = t_0 + 1$. We divide into two sub-cases.

(b-i) If $\zeta_{1,t_0+1} \leq 0$, the statement of the lemma holds trivially for time $t = t_0 + 1$.

(b-ii) If $\zeta_{1,t_0+1} \geq 0$, then by the induction hypothesis and Lemma 5, we must have $\zeta_{2,t_0+1} \leq 0$.

Therefore, the statement of the lemma holds for time $t = t_0 + 1$.

□

Finally, we can state the proof of Theorem 2.

Proof: (Proof of Theorem 2)

Letting $t = \mathcal{T}$ in Lemma 6, we have that $\zeta_{2,\mathcal{T}} \leq 0$. Then, from Lemma 4, we have that by following Algorithm 1, $C^{\pi}(\vec{A}(1:\mathcal{T})) \leq \overline{\mathrm{CR}} \cdot C^{\mathrm{OPT}}(\vec{A}(1:\mathcal{T}))$ for all $\vec{A}(1:\mathcal{T}) \in \mathcal{U}$. The result of the theorem then follows.

□

REFERENCES

[1] M. Shi, X. Lin, S. Fahmy, and D.-H. Shin, "Competitive online convex optimization with switching costs and ramp constraints," in *Proc. IEEE INFOCOM - IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 1835–1843.

[2] L. Jiao, L. Pu, L. Wang, X. Lin, and J. Li, "Multiple granularity online control of cloudlet networks for edge computing," in *Proc. 15th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2018, pp. 1–9.

[3] Z. Liu, I. Liu, S. Low, and A. Wierman, "Pricing data center demand response," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 1, pp. 111–123, Jun. 2014.

[4] H. Wang, J. Huang, X. Lin, and H. Mohsenian-Rad, "Exploring smart grid and data center interactions for electric power load balancing," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 3, pp. 89–94, Jan. 2014.

[5] N. Chen, A. Agarwal, A. Wierman, S. Barman, and L. L. H. Andrew, "Online convex optimization using predictions," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 1, pp. 191–204, Jun. 2015.

[6] N. Chen, J. Comden, Z. Liu, A. Gandhi, and A. Wierman, "Using predictions in online optimization: Looking forward with an eye on the past," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 44, no. 1, pp. 193–206, Jun. 2016.

[7] M. Lin, A. Wierman, Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1378–1391, Oct. 2013.

[8] M. Lin, Z. Liu, A. Wierman, and L. L. H. Andrew, "Online algorithms for geographical load balancing," in *Proc. Int. Green Comput. Conf. (IGCC)*, Jun. 2012, pp. 1–10.

[9] Z. Liu, A. Wierman, Y. Chen, B. Razon, and N. Chen, "Data center demand response: Avoiding the coincident peak via workload shifting and local generation," *Perform. Eval.*, vol. 70, no. 10, pp. 770–791, Oct. 2013.

[10] L. Lu, J. Tu, C.-K. Chau, M. Chen, and X. Lin, "Online energy generation scheduling for microgrids with intermittent energy sources and co-generation," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 1, pp. 53–66, Jun. 2013.

[11] S.-J. Kim and G. B. Giannakis, "An online convex optimization approach to real-time energy pricing for demand response," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2784–2793, Nov. 2017.

[12] V. Joseph and G. de Veciana, "Jointly optimizing multi-user rate adaptation for video transport over wireless systems: Mean-fairness-variability tradeoffs," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 567–575.

[13] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 460–468.

[14] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. 20th Int. Conf. Mach. Learn. (ICML)*, 2003, pp. 928–936.

[15] L. Xiao, "Dual averaging methods for regularized stochastic learning and online optimization," *J. Mach. Learn. Res.*, vol. 11, pp. 2543–2596, Oct. 2010.

[16] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2011.

[17] E. Hazan, "Introduction to online convex optimization," *Found. Trends Optim.*, vol. 2, nos. 3–4, pp. 157–325, 2016.

[18] *ETSI Network Functions Virtualisation (NFV) Architectural Framework*. Accessed: Jan. 24, 2021. [Online]. Available: https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV%20002v1.2.1%20-%20GS%20-%20NFV%20Architectural%20Framework.pdf

[19] X. Wang, C. Wu, F. Le, and F. C. M. Lau, "Online learning-assisted VNF service chain scaling with network uncertainties," in *Proc. IEEE 10th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2017, pp. 205–213.

[20] Y. Jia, C. Wu, Z. Li, F. Le, and A. Liu, "Online scaling of NFV service chains across geo-distributed datacenters," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 699–710, Apr. 2018.

[21] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, p. 2, 2014.

[22] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 1346–1354.

[23] S. Zhao, X. Lin, D. Aliprantis, H. N. Villegas, and M. Chen, "Online multi-stage decisions for robust power-grid operations under high renewable uncertainty," in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[24] M. H. Hajiesmaili, C.-K. Chau, M. Chen, and L. Huang, "Online microgrid energy generation scheduling revisited: The benefits of randomization and interval prediction," in *Proc. 7th Int. Conf. Future Energy Syst.*, Jun. 2016, p. 1.

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

[26] M. Badiei, N. Li, and A. Wierman, "Online convex optimization with ramp constraints," in *Proc. 54th IEEE Conf. Decis. Control (CDC)*, Dec. 2015, pp. 6730–6736.

[27] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust Optimization*. Princeton, NJ, USA: Princeton Univ. Press, 2009.

[28] S. Zhao, X. Lin, and M. Chen, "Peak-minimizing online EV charging: Price-of-uncertainty and algorithm robustification," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 2335–2343.

[29] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski, "Adjustable robust solutions of uncertain linear programs," *Math. Program.*, vol. 99, no. 2, pp. 351–376, Mar. 2004.

[30] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory Design*. Portland, OR, USA: Nob Hill Pub., 2009.

[31] E. F. Camacho and C. B. Alba, *Model Predictive Control*. London, U.K.: Springer, 2013.

[32] H. Moens and F. D. Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. 10th Int. Conf. Netw. Service Manage. (CNSM) Workshop*, Nov. 2014, pp. 418–423.

[33] A. Gupta, M. F. Habib, U. Mandal, P. Chowdhury, M. Tornatore, and B. Mukherjee, "On service-chaining strategies using virtual network functions in operator networks," *Comput. Netw.*, vol. 133, pp. 1–16, 2018.

[34] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.

[35] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *Proc. 11th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2015, pp. 50–56.

[36] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic virtual network function placement," in *Proc. IEEE 4th Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2015, pp. 255–260.

[37] X. Zhang, C. Wu, Z. Li, and F. C. M. Lau, "Proactive VNF provisioning with multi-timescale cloud resources: Fusing online learning and online optimization," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.

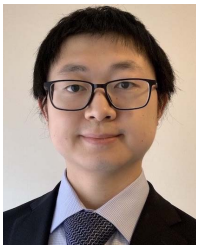
[38] D. Fotakis, "On the competitive ratio for online facility location," *Algorithmica*, vol. 50, no. 1, pp. 1–57, Jan. 2008.

[39] S. Alaei, M. Hajiaghayi, and V. Liaghat, "The online stochastic generalized assignment problem," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2013, pp. 11–25.

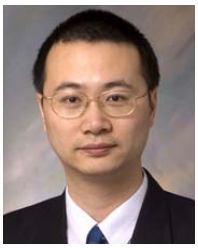
[40] D. Newman. (1999). *Benchmarking Terminology for Firewall Performance*. [Online]. Available: https://tools.ietf.org/html/rfc2647

[41] L. Andersson and T. Madsen. (2005). *Provider Provisioned Virtual Private Network (VPN) Terminology*. [Online]. Available: https://tools.ietf.org/html/rfc4062

[42] L. Cao, P. Sharma, S. Fahmy, and V. Saxena, "NFV-VITAL: A framework for characterizing the performance of virtual network functions," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2015, pp. 93–99.
[43] S. Zhao, X. Lin, and M. Chen, "Robust online algorithms for peak-minimizing EV charging under multistage uncertainty," *IEEE Trans. Autom. Control*, vol. 62, no. 11, pp. 5739–5754, Nov. 2017.
[44] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[45] M. Shi, X. Lin, S. Fahmy, and D.-H. Shin, "Competitive online convex optimization with switching costs and ramp constraints," Purdue Univ., West Lafayette, IN, USA, Tech. Rep., 2017. [Online]. Available: https://engineering.purdue.edu/%7elinx/papers.html
[46] D. Bernstein, "Containers and cloud: From LXC to Docker to kubernetes," *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81–84, 2014.
[47] *The Cogent's Network Map*. Accessed: Jan. 24, 2021. [Online]. Available: https://cogentco.com/en/network/network-map
[48] D. Gmach *et al.*, "Profiling sustainability of data centers," in *Proc. IEEE Int. Symp. Sustain. Syst. Technol.*, May 2010, pp. 1–6.



Ming Shi (Student Member, IEEE) received the B.S. degree from Tianjin University, Tianjin, China, in 2015. He is currently pursuing the Ph.D. degree in Electrical and Computer Engineering with Purdue University, West Lafayette, IN, USA. His research interests are in competitive online algorithms, with applications in various networking problems, such as network functions virtualization.



Xiaojun Lin (Fellow, IEEE) received the B.S. degree from Zhongshan University, Guangzhou, China, in 1994, and the M.S. and Ph.D. degrees from Purdue University, West Lafayette, IN, USA, in 2000 and 2005, respectively. He is currently a Professor of Electrical and Computer Engineering with Purdue University. His research interests are in the analysis, control, and optimization of large and complex networked systems, including both communication networks and power grid. He received the IEEE INFOCOM 2008 Best Paper and 2005 Best Paper of the year Award from the *Journal of Communications and Networks*. He received the NSF CAREER Award in 2007. He was the Workshop Co-Chair for IEEE GLOBECOM 2007, the Panel Co-Chair for WICON 2008, the TPC Co-Chair for ACM MobiHoc 2009, the Mini-Conference Co-Chair for IEEE INFOCOM 2012, and the General Co-Chair for ACM e-Energy 2019. He has served as an Area Editor for *Computer Networks Journal* (Elsevier), an Associate Editor for the IEEE/ACM TRANSACTIONS ON NETWORKING, and a Guest Editor for *Ad Hoc Networks journal* (Elsevier).

Sonia Fahmy (Senior Member, IEEE) is currently a Professor of Computer Science with Purdue University. Her research interests lie in the design and evaluation of network architectures and protocols. She received the National Science Foundation CAREER award in 2003, and was named a Purdue University Faculty Scholar in 2016. Her research has been supported by grants from the government and industry including NSF, DHS, Cisco Systems, Hewlett-Packard, Northrop Grumman, Schlumberger, and BBN. She served on the organizing or technical program committees of several conferences, including ACM SIGCOMM, SIGMETRICS, MOBICOM, CoNEXT, and SOSR, and IEEE INFOCOM, ICNP, and ICDCS, and on the Editorial Boards of the IEEE/ACM TRANSACTIONS ON NETWORKING, the IEEE TRANSACTIONS ON MOBILE COMPUTING, and the IEEE TRANSACTIONS ON COMPUTERS.