Online Network Flow Optimization for Multi-Grade Service Chains

Víctor Valls*, George Iosifidis†, Geeth de Mel‡, Leandros Tassiulas*
*Department of Electrical Engineering and Institute for Network Science, Yale University
†School of Computer Science and Statistics, Trinity College Dublin
‡IBM Research UK

***Abstract*—We study the problem of in-network execution of data analytic services using multi-grade VNF chains. The nodes host VNFs offering different and possibly time-varying gains for each stage of the chain, and our goal is to maximize the analytics performance while minimizing the data transfer and processing costs. The VNFs' performance is revealed only after their execution, since it is data-dependent or controlled by third-parties, while the service requests and network costs might also vary with time. We devise an operation algorithm that learns, on the fly, the optimal routing policy and the composition and length of each chain. Our algorithm combines a lightweight sampling technique and a Lagrange-based primal-dual iteration, allowing it to be scalable and attain provable optimality guarantees. We demonstrate the performance of the proposed algorithm using a video analytics service, and explore how it is affected by different system parameters. Our model and optimization framework is readily extensible to different types of networks and services.**

I. Introduction

A. Background & Motivation

Today we witness the rapid deployment of a new breed of services that require collecting and processing data in (almost) real-time. From data analytics in the Internet of Things [1], to augmented information services [2], and big data streaming [3], there is a plethora of applications where users create data flows that need to be analyzed swiftly. This, in turn, has spurred the deployment of edge computing servers and in-network middle-boxes that can execute these tasks while the data is en route. This transformation of networks is fueled by technologies such as NFV and SDN, which enable the dynamic management of network/computing resources, and promise unprecedented performance gains.

However, a key challenge in these networks is the optimal design of routing and processing policies. Typical multi-commodity flow algorithms do not consider the need (or, possibility) of in-network processing, which requires to decide if, where, and how to process the data while routed towards their destination. With the increasing softwarization of networks, such computations can be executed at different nodes (by VNFs) and in multiple stages (VNF chains), and this plenitude of options compounds further the policy design. Several works have recently focused on this problem. For instance, [4] proposed flow optimization algorithms for networks with middle-boxes, and [5] for cellular networks; [3] studied online routing/computing policies for big data streams, and [6], [7] for multi-stage processing tasks. However, a key question that has received, surprisingly, less attention is: *how these policies affect the performance and cost of the services?*

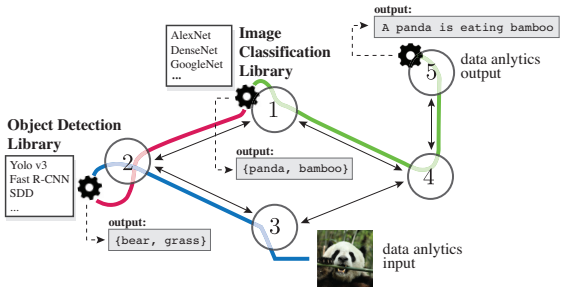

Fig. 1. **Video analytics service** implemented in 3 stages at nodes $2 \to 1 \to 5$. The colored lines indicate the different types of data (outputs of each stage): object detection with VNFs such as [8], [9]; refined object classification with [10], [11]; image description with [12], [13].

In practice, the performance of analytics and information services depends on *(i)* the actual data to be processed, *(ii)* the VNF running at each node, and *(iii)* the service chain composition. To exemplify, consider a network that collects and analyzes video streams in order to detect objects of interest, Fig. 1. Abundant experiments (including our own, see Sec. V) show that the detection depends on the quality of the features extractor, the algorithms that classify them, and the extractor/classifier combination. Moreover, a classifier that is trained to identify, say, animals might perform poorly if the data depict plantation. Hence, it is impossible to quantify the service performance before its execution, a problem that is exaggerated when the network is not aware of the accuracy (or, *grade*) of the VNFs running at each node.[1] And similar challenges arise with respect to service costs since, for instance, the computation cost depends on the VNF configuration.

This paper aspires to fill this gap by introducing an analytical framework for the in-network execution of such multi-grade service chains, that enables us to maximize their performance while overcoming key practical limitations.

B. Methodology & Contributions

We consider the time-slotted operation of a network with arbitrary routing/processing costs and analytics performance. The requests are generated by random processes with unknown statistical properties, and might even be revealed after network decisions are made in each slot. The links and nodes have different (fixed or time-varying) transmission and processing

[1]The networks may use computing nodes that are controlled by, e.g., over-the-top providers, or nodes that change dynamically their training datasets.

capacities. Each node might host one or more VNFs, which are combined to create a multi-grade VNF chain. We allow the possibility for early-exit from the chain, i.e., routing the data to sink node(s) that execute the remaining computations. The routing and processing decisions are coupled because the VNFs modify the volume and type of their input.[2] The performance of the VNF(s) at each node is unknown, and possibly time-varying, and we observe it only for the network decisions x_t made at each slot t. Unlike previous approaches, e.g., [14], [15], that assume fixed statistical distributions for these parameters, we develop an online mechanism which gradually learns the analytics gains, network costs, and user requests, and adapts its operation accordingly.

Our approach is inspired by the online convex optimization (OCO) framework introduced in [16] which optimizes time-varying convex functions over fixed constraints. We extend these ideas here in order to cope with the limited observations about the analytics performance and the network costs. To this end, we combine a versatile sampling technique [17] and a primal-dual online learning algorithm that handles, in a unified fashion, unknown constraints and objectives. Our main theorem proves that the proposed approach succeeds in learning the optimal performance of the system, where the benchmark is set by a hypothetical static policy that has access to a complete-information oracle.

We demonstrate the performance of our solution using the state-of-the-art Yolo v3 application [8] for video analytics in different operation modes (to capture the multiple VNF grades); representative data sets [12]; and synthetic experiments with large networks. We verify that the proposed online algorithm is scalable, lightweight, and succeeds in learning the optimal network operation policy relatively fast. In fact, it achieves asymptotically a zero optimality gap (or, *sublinear regret*) and zero constraint violation when compared to a hypothetical policy that has access to an oracle, i.e., the complete information about future arrivals and VNF configurations.

This work makes the following technical contributions:

(i) Multi-Grade Service Performance: We propose and analyze the problem of maximizing the performance of in-network data analytics using multi-grade VNF chains. This is an increasingly important problem, and we study it in its general form, under several practical limitations.

(ii) Online Learning Algorithm: A new online network optimization framework is introduced, extending ideas from OCO to account for network constraints and limited feedback. Our algorithm achieves $O(\sqrt{T})$ regret *and* $O(1/\sqrt{T})$ average constraint violation. This is a general and unifying algorithmic framework of independent interest.

(iii) Experimental Evaluation: In a set of experiments and synthetic simulations, we study the performance of our approach, namely how fast the algorithm learns the optimal performance and does not violate the constraints. We use a service with multi-stage execution, and demonstrate the robustness of our approach in different/changing VNF grades.

[2]For example, feature extraction might reduce a video frame from several MBytes to few KBytes; and changes the data from raw video to meta-data.

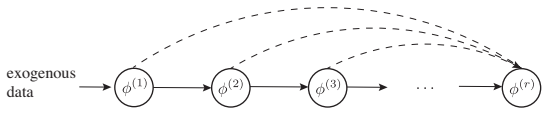

Fig. 2. **Service chain example**. Each function (stage) processes the data, changing their type, i.e., creating a new commodity, that is fed to the next stage. The network can also decide to stop processing the data and directly route them to destination (dashed line; changing to commodity r).

Organization. Sec. II presents the system model and Sec. III the resource allocation algorithms. We discuss extensions in Sec. IV, present the performance evaluation in Sec. V, and related work in Sec. VI. All the proofs are in the Appendix.

II. System Model

We model a communication network with a directed graph $G(V, E)$ consisting of $V := \{v_1, \ldots, v_m\}$ nodes and $E := \{e_1, \ldots, e_n\}$ links, $m = |V|$, $n = |E|$. The system model consists of four parts: service chain, decision variables, network constraints, and objective functions. To streamline exposition, we present a basic model with one user. The case of multiple users, and several other extensions, are presented in Sec. IV.

A. Service Chain and Commodities

We model a service chain for data analytics as an ordered collection of $r \in \mathbf{N}$ functions (or, VNFs) $\phi^{(1)}, \phi^{(2)} \ldots, \phi^{(r)}$, where the output of the i'th function, $i \in \{1, \ldots, r-1\}$, is the input of the next function in the chain. The output of each function is modeled as a new type of commodity, and without loss of generality we assume that the last function is a sink (does not generate data). Hence, there are r types of commodities, the first of which, $i = 1$, corresponds to exogenous data arrivals (requests). We also allow functions to terminate the in-network processing by transforming a commodity from type $i \in \{1, \ldots, r-1\}$ directly to type r, which is then routed to the end sink node, Fig. 2. This *early-exit* option is useful, for instance, when the network resources are scarce and/or the computations cost-inefficient, and offers an additional degree of freedom in our policy.[3]

A node can execute more than one VNFs, and each VNF can be executed by different nodes. We use vector $w^{(i)} \in \{0,1\}^m$ to indicate the nodes where function $\phi^{(i)}$ runs. Specifically, if the k'th component of vector $w^{(i)}$ is equal to 1, it means that node v_k runs function $\phi^{(i)}$. Similarly, we use vector $\lambda^{(i)} \in \mathbf{R}_+^m$ to denote the exogenous arrivals of commodity $i \in \{1, \ldots, r\}$, where its k'th component represents the arrivals at node v_k, $k \in \{1, \ldots, m\}$. Finally, we collect every $\lambda^{(i)}$, $i \in \{1, \ldots, r\}$ in vector $\lambda = (\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(m)})$. Note that $\lambda \in \mathbf{R}^{rm}$.

B. Decision Variables

There are three types of variables per commodity, indicating how much data to: *(i)* route over each link; *(ii)* process at each node; and *(iii)* pull out from the service chain. Vector $x^{(i)} \in \mathbf{R}^{n+2m}$ collects the variables of each commodity

[3]This model generalization captures several practical cases, e.g., chains that have varying length, and analytics with performance that increases with the processing stages. See also the discussion in Sec. IV.

$i \in \{1,\dots,r\}$. Specifically, the j'th component of $x^{(i)}$ indicates the amount of data we decide to route over link e_j, $j \in \{1,\dots,n\}$; the $(k+n)$'th component the amount of data to process at node v_k, $k \in \{1,\dots,m\}$; and the $(l+n+m)$'th component the amount of data node v_l, $l \in \{1,\dots,m\}$ decides to transform to type r and send, possibly over multiple hops, to the sink. We collect all decision variables for all commodities in vector $x = (x^{(1)},\dots,x^{(r)}) \in \mathbf{R}^{r(n+2m)}$.

C. Constraints

The capacity constraints capture the maximum amount of data that can be routed over a link and processed by a node. Let $c(e_j), c(v_k) \in \mathbf{R}_+$ denote the capacities of link e_j and node v_k, $j \in \{1,\dots,n\}$, $k \in \{1,\dots,m\}$, respectively. The *link capacity constraints* are

$$\sum_{i=1}^{r} x^{(i)}(e_j) \le c(e_j) \qquad \forall j \in \{1,\dots,n\} \tag{1}$$

where $x^{(i)}(e_j)$ denotes the j'th component of vector $x^{(i)}$. Similarly, the *node capacity constraints* are given by

$$\sum_{i=1}^{r} x^{(i)}(v_k) \le c(v_k) \qquad \forall k \in \{1,\dots,m\} \tag{2}$$

where $x^{(i)}(v_k)$ denotes the $(n+k)$'th component of $x^{(i)}$. We assume there is no capacity constraint associated with retagging a commodity from type $i \in \{1,\dots,r-1\}$ to type r. Finally, we collect the capacity constraints in set

$$C := \left\{ x \in \mathbf{R}^{n+2m} \mid \text{Eq. (1) and (2)} \right\}. \tag{3}$$

We augment the typical per-node flow conservation constraints in order to account for the service chain operation. In particular, the processing of the i'th commodity generates the $(i+1)$'th commodity, and a commodity of type $i \in \{1,\dots,r-1\}$ can be transformed into commodity r at any stage (stopping in-network processing) and then routed to the sink node(s). We write the *augmented flow conservation constraints* as the system of equations

$$Ax + Bx + \lambda = 0, \tag{4}$$

where $A, B \in \{-1,0,1\}^{rm \times r(n+2m)}$ are routing and processing matrices respectively.

We describe first matrix A. Let $\mathcal{I} \in \{-1,0,1,\}^{m \times n}$ be the incidence matrix where the (i,j)'th entry is given by

$$\mathcal{I}(i,j) = \begin{cases} 1 & \text{if link } j \text{ starts at node } i \\ -1 & \text{if link } j \text{ ends at node } i \\ 0 & \text{otherwise} \end{cases}$$

The incidence $\mathcal{I}$ captures that the same amount of commodity that leaves a node must enter another node [18, Chap. 1]. Next, since the routing is not affected by the processing, we define $I := [\mathcal{I}, \mathbf{0}_{m\times m}, \mathbf{0}_{m\times m}]$ where $\mathbf{0}_{m\times m}$ is a $m \times m$ zero matrix. The routing matrix of all the commodities in the network is

$$A = \begin{bmatrix} I_1 & 0 & \dots & 0 \\ 0 & I_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & I_r \end{bmatrix},$$

where $I_i = I$ for all $i \in \{1,\dots,r\}$. Vector $(Ax) \in \mathbf{R}^{rm}$ is known as the nodes' divergence and captures the net increment of commodities in each of the nodes (see [18, Sec. 1E]).

Similarly for the processing matrix, we model how data changes type, as if we had "virtual" links. Let

$$\begin{aligned} J_i &= [\mathbf{0}_{m\times n}, -\mathrm{diag}(w^{(i)}), -\mathrm{diag}(w^{(i)})], \\ K_i &= [\mathbf{0}_{m\times n}, \mathrm{diag}(w^{(i)}), \mathbf{0}_{m\times m}], \\ L_i &= [\mathbf{0}_{m\times n}, \mathbf{0}_{m\times m}, \mathrm{diag}(w^{(i)})], \end{aligned}$$

and observe that $J_i, K_i, L_i \in \mathbf{R}^{m\times(n+2m)}$ since $\mathrm{diag}(w^{(i)})$ is an $m \times m$ matrix. Matrix J_i captures the amount of commodity i processed, and K_i, L_i the commodity i that is transformed into a commodity of type $i+1$ and r, respectively. By arranging the matrices accordingly, we can write the processing matrix that captures the dynamics in the service chain as follows

$$B = \begin{bmatrix} J_1 & 0 & \dots & \dots & 0 \\ K_1 & J_2 & & & \vdots \\ 0 & K_2 & \ddots & & \vdots \\ \vdots & & \ddots & J_{r-1} & 0 \\ L_1 & \dots & L_{r-2} & K_{r-1}+L_{r-1} & J_r \end{bmatrix}.$$

Our model is built using minimal assumptions and, as we explain in Sec. IV, it is readily extensible.

D. Cost Functions

Our objective is to maximize the analytics performance by using the nodes that yield the highest task precision (or, other service-specific metrics), while minimizing the data routing and processing costs. Therefore, we define the function

$$f : C \times \Theta \to \mathbf{R}^{r(n+2m)} \tag{5}$$

which takes as input an eligible policy vector, $x \in C$, and outputs a *cost* vector[4] with a component for each decision in x. Namely, the first n elements of $f(\cdot)$ correspond to the routing cost parameters (one per link, capturing, e.g., the delay); the next m components to the *net reward*, i.e., analytics gain minus processing cost (due to, e.g., energy consumption of VNFs); and the last m components model the impact for terminating early the in-network processing. The randomness of rewards and costs is captured by θ which modulates function $f(\cdot,\theta)$. This parameter is drawn from $\Theta := \{\theta_1,\dots,\theta_p\}$ that collects the p different possible system states, e.g., VNF configurations[5], link costs, etc. We consider

[4]To streamline exposition, f is referred as cost function, and we will be minimizing the negative rewards (hence, maximizing performance).

[5]For instance a node offers different accuracy in an image classification function based on whether it uses AlexNet [10] or DenseNet [11]

the most general case where θ is allowed to vary arbitrarily. Finally, we define the *network cost* function as:

$$\Phi(x, \theta) := \langle \mathbf{1}, f(x, \theta) \rangle, \tag{6}$$

where $\mathbf{1}$ is the all ones vector and $\langle \cdot, \cdot \rangle$ the inner product.

Our model is general and imposes minimal assumptions on the system operation, and Θ can be defined in any way that suits the specific application. In case of a static system and complete information, the problem to solve is

$$\min_{x \in C} \Phi(x, \theta) \quad \text{s.t.} \quad Ax + Bx + \lambda = 0. \tag{7}$$

However, in practice the requests and system configurations change with time, and moreover might not observable when we select x. In the next section, we present an online learning algorithm that addresses both of these issues.

III. Online Resource Allocation

We formalize here the *online* resource allocation problem, introduce our optimality criterion, and present our algorithm.

A. Online problem

The network operation is time slotted. Data analytic requests arrive at nodes according to a stochastic process $\{\lambda_t\}$ with mean $\lambda \in \mathbf{R}^{rm}$, and the system state θ_t changes following an arbitrary random process. At the beginning of slot t, the network controller (NC) makes a resource allocation decision $x_t \in C \subset \mathbf{R}^{r(n+2m)}$ which induces a (randomized) cost vector F_t drawn from $f(x_t, \theta_t)$. Our goal is to find a policy that minimizes the network cost, overcoming these key issues:

(I1) The statistics of the arrivals $\{\lambda_t\}$ are *not* known. Also, each λ_t might be revealed after decision x_t is made.[6]

(I2) The cost vector F_t associated with a resource allocation decision x_t is not known before implementing x_t.

(I3) The system configuration $\{\theta_t\}$ might vary in an arbitrary fashion. Therefore, the distribution from which the reward vectors F_t are drawn might as well change.

Issue (I1) renders impossible to design an offline policy that satisfies the network constraints. Hence, we replace the "static" augmented flow conservation constraint in Eq. (4) with their "relaxed" dynamic version:

$$\lim_{T \to \infty} \mathbf{E} \left[\frac{1}{T} \sum_{t=1}^{T} (Ax_t + Bx_t + \lambda_t) \right] = 0. \tag{8}$$

That is, we now enforce that all traffic is served asymptotically.[7] Observe that we can write Eq. (8) as $\mathbf{E}[A\hat{x} + B\hat{x} + \lambda] = 0$ where $\hat{x} := \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} x_t$.

Due to issue (I2), prior frameworks that decide resource allocation assuming known gradients of cost functions, e.g., [19], [20], or by solving sequences of known programs, e.g., [21], [22], cannot be applied here. On the contrary, our approach works only by observing the specific outcome of each action x_t, and under the following minimal assumption:

Assumption 1. $\mathbf{E}[F_t] = f(x_t, \theta)$ *and* $\mathrm{Var}(F_t) < \infty$ *for all* $t \in \mathbf{N}$. *The expectation is with respect to* x_t *for a fixed* θ.

This is a valid assumption here and holds when, e.g., the information to analyze is in a database, and we can select it in a randomized manner; or when we have streaming applications (e.g., speech recognition) where information (e.g., words) is drawn from the same source (e.g., conversation). Importantly, we do not assume that the expected cost of the objective functions is known, as this depends on the actual data.

Finally, (I3) might arise for various practical reasons. For instance, the computing nodes might be owned by third parties and hence the NC does not have access to their configuration (vector θ_t). Similarly, the nodes configuration can change with time as they update their training data sets (used for the analysis), or even the NC might on purpose try different options in an effort to tailor the system on the received data.

B. Optimality Criterion

We quantify the algorithm's performance using the criterion of "regret". This metric is extensively used in machine learning (e.g., [16], [17], [23]) where the cost of making an action is not known a priori and decided by an adversary. In our case, the "adversary" corresponds to the random and changing system states (due to (I1)-(I3)) that alter its operation, and force us to relearn how to route and process data.

Definition 1 (Regret)**.** *Let* $\{x_t\}$, $t = 1, \ldots, T$ *is a sequence of actions from* C, *and define the* static *feasible set*

$$X := \{x \in C \mid Ax + Bx + \lambda = 0\}. \tag{9}$$

The regret is defined as

$$R_T := \sum_{t=1}^{T} \Phi(x_t, \theta_t) - \min_{x \in X} \sum_{t=1}^{T} \Phi(x, \theta_t) \tag{10}$$

where $\Phi(x_t, \theta_t) = \mathbf{E}[\langle \mathbf{1}, F_t \rangle]$ *(by Assumption 1 and the definition of the network cost in Eq.* (6)*).*

Technically, the regret captures the difference between the expected accumulated cost and the expected cost of the best *fixed* policy designed in hindsight at time $T \in \mathbf{N}$. That is, the best decision if the future events and costs were known in advance, which is a hypothetical solution not available in practice. The goal is to design a policy that obtains sub-linear regret (i.e., $\limsup_{T \to \infty} R_T / T \leq 0$) and sublinear constraint violation. Hence, the online algorithm will perform asymptotically as good as the benchmark policy, while satisfying the constraints.

C. Algorithm

We first relax the constraints and define the Lagrangian:

$$\mathcal{L}_t(x, y) := \Phi(x, \theta) + \langle y_t, Ax + Bx + \lambda_t \rangle,$$

where $y_t \in \mathbf{R}_+^{rm}$ is a non-negative penalty parameter. Also, we make the following standard assumptions.

[6]For example, because x_t is implemented at the beginning of a time slot, and some requests are created *during* the slot. This is important when slicing or SDN is used, where often paths are established in advance. Knowing λ_t before setting x_t is a simpler case that our model also addresses.

[7]The expectation is taken with respect to random variable λ_t, $t \in \mathbf{N}$.

Algorithm 1

1: **Input:** sets C and matrices A, B
2: **Set:** $x_1 = 0$; $y_1 = 0$; $g = 0$, $\rho = 1/\sqrt{T}$
3: **for** $t = 1, 2, \dots$ **do**
4: $\quad x_{t+1} \leftarrow \mathcal{P}_{C_t}(x_t - \rho g_t)$
5: $\quad$ observe cost vector $F_{t+1} \in \mathbf{R}^{r(n+2m)}$
6: $\quad g_{t+1} \leftarrow F \circ \varphi(x_{t+1}) + \langle A + B, y_t \rangle$
7: $\quad y_{t+1} \leftarrow [y_t + \rho(Ax_{t+1} + Bx_{t+1} + \lambda_{t+1})]^+$
8: **end for**

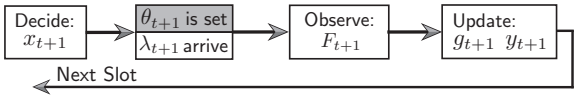

Fig. 3. **Timing of system operation**: (1) the current-slot policy is decided; (2) the system configuration is set (gray box; not observable) and the requests arrive; (3) the current service performance is observed (not the entire function f); (4) the algorithm updates are calculated. Repeat from (1).

Assumption 2. *There exists a $\eta > 0$ and an $x \in C$ such that $Ax + Bx + \lambda + \mathbf{1}\eta \preceq 0$.*

Assumption 3. *$\{\lambda_t\}$ is an i.i.d. process with expected value λ and finite variance.*

Assumption 2 is a "one-sided" Slater condition, requiring that the interior of the network capacity region is non-empty [24].

Algorithm 1 is a primal-dual *stochastic* subgradient method on the Lagrangian with a time-varying action set[8] $C_t := \{u \in C \mid Au + Bu + \sum_{i=1}^{t-1}(Ax_i + Bx_i + \lambda_i) \succeq 0\}$. First, there is a projected stochastic subgradient ascent on the Lagrangian (line 4), where ρ is the learning rate and $\mathcal{P}_{C_t}$ the Euclidean projection onto C_t. Then, we observe the vector of costs (line 5), and compute an *unbiased* estimate of the cost functions gradient (line 6). Finally, we perform a stochastic dual-subgradient descent update (line 7). See also Fig. 3.

The convergence of the algorithm relies on the subgradient estimates being unbiased. For the dual variable update, this holds because of Assumption 3. For the primal update, we rely on the following proposition:

Proposition 1. *Suppose that $f(x, \theta)$ is linear in x for a fixed $\theta \in \Theta$. Define $\varphi(y) = 1/y$ if $y \neq 0$ and $\varphi(y) = 0$ otherwise. For a fixed $\theta \in \Theta$, we have*

$$\nabla\Phi(x, \theta) = \mathbf{E}[F_t \circ \varphi(x)],$$

where $\nabla\Phi(x, \theta_t)$ denotes a subgradient of $\Phi(\cdot, \theta_t)$ at x, and $\circ$ the entry-wise product of two vectors.

The proof is presented in the Appendix, and relies on the fact that the cost of each function is proportional to the amount of resources allocated. Our main technical contribution is the following theorem.

Theorem 1 (Online Primal-Dual SGD)**.** *Define constants σ, D, G such that $\sigma^2 \geq \max_{u \in C} \|Au + Bu + \lambda_t\|^2 \ \forall t \in \mathbf{N}$, $D := \max_{u,v \in C} \|u - v\|$ and $\|F_t \circ \varphi(x_t)\| \leq G$. If Assumptions 1, 2 and 3 are satisfied, Algorithm 1 ensures that*

$$\text{(i)} \quad R_T \leq \left(G + D^2 + \frac{\sigma^2}{2}\right)\sqrt{T} = O(\sqrt{T})$$

$$\text{(ii)} \quad \mathbf{E}\left[\left\|\frac{1}{T}\sum_{t=1}^{T}(Ax_t + Bx_t + \lambda_t)\right\|\right] \leq \frac{\Gamma}{\sqrt{T}} = O\left(\frac{1}{\sqrt{T}}\right)$$

$$\Gamma := \sqrt{(\tfrac{1}{rm\eta}(\sigma^2 + 2GD + D^2/2))^2 + 2\sigma^2 + 4GD + D^2}.$$

Claim (i) shows that the regret increases at rate of $R_T \leq O(\sqrt{T})$, and therefore, the gap between the average performance of Algorithm 1 and the best fixed decision reduces sub-linearly, i.e., $\limsup_{T\to\infty} R_T/T \leq 0$. Constants G, D, σ^2 capture how the regret scales with the network size, the functions in the chain, and the variance of the arrival process.

Claim (ii) states that the constraint violation decreases with T. Observe that $\Gamma/\sqrt{T} \to 0$ as $T \to \infty$, and hence the "relaxed" dynamic constraint in Eq. (8) will be satisfied. We note also that constant Γ depends on $\eta > 0$ (see Assumption 2), which captures, qualitatively, the network capacity region size. And the larger η is, the better.[9] Finally, the bounds in Theorem 1 hold for a fixed time horizon T, but they can be readily extended using the "doubling trick"; see [25, Sec. 2.3].

IV. Extensions & Discussion

Our framework can be generalized for different system architectures and different types of service chains. Due to lack of space, we discuss briefly some representative cases.

1) Multiple users: The model is readily extensible to scenarios of multiple users. To see this, let $u \in \mathbf{N}$ be the number users, and let us use subscripts to indicate the user number. For example, x_i indicates the decision variables of user $i \in \{1, \dots, u\}$ as described in Sec. II-B. We can redefine the variables and parameters as follows $x = (x_1, \dots, x_u)$, $\lambda = (\lambda_1, \dots, \lambda_u)$, $A = \text{diag}(A_1, \dots, A_u)$, $B = \text{diag}(B_1, \dots, B_u)$[10] and $f = (f_1, \dots, f_u)$.[11] Recall that f_i is a vector. The augmented flow conservation constraints can be written as in Eq. (4). For the capacity constraints, we must simply take into account that the sum of *all* commodities needs to respect the capacity of each link/node.

2) Processing costs: Eq. (2) considers that all algorithms have the same computational needs. However, we can model the different requirements by replacing it with $\sum_{i=1}^{r} \gamma^{(i)}(v_k)x^{(i)}(e_j) \leq c(e_j)$, where $\gamma^{(i)}(v_k) > 0$ captures the "cost" of processing a unit of commodity $i \in \{1, \dots, r\}$ at node v_k, $k \in \{1, \dots, m\}$.

3) Flow augmentations: In Sec. II, the processing of each commodity i generates a unit of commodity $i+1$. But, we can relax this assumption and allow the flow volume to change after each processing stage (e.g., to model data compression, etc.). This extension is as simple as adding a multiplicative

[8]Note that we use $\succeq$, unlike other stochastic formulations, see [24], in order to ensure we only process actual data (and not dummy data) and this way avoid creating artificial/pseudo reward gains. This problem does not arise in networks that do not have processing nodes.

[9]η can be seen as the additional data we could send over a link or process in a node while still satisfying the augmented flow conservation constraints.

[10]Here, diag is a diagonal matrix where each diagonal entry is a matrix.

[11]The dimensions of f_i, A_i and B_i depend on the number of function in the service chain of each user.

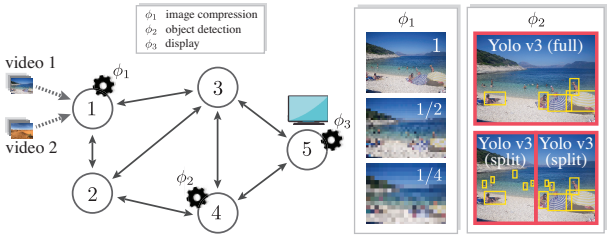

Fig. 4. Illustrating the network of the example in Sec. V. The first function in the chain compresses the video frames; the second analyzes images with a video frame; and the third display the results on a monitor. The functions in the service chain can be executed in multiple modes.

factor to matrix K_i in the matrix B. For instance, if the first "commodity" column of B is $(J_1, \kappa_1 K_1, \ldots, L_1)$ with $\kappa_1 > 0$, that means that κ_1 units of commodity 2 are generated as a result of processing a unit of commodity 1 at node 1.

4) Wireless constraints: In case of a wireless network, we might need to amend our model so as to handle time varying link capacities and, possibly, interference. The former can be incorporated in the changing action set C_t, under some assumptions about the Slater vector. The interference, on the other hand, requires the calculation of link activation schedules, which can be included in C_t. Finding such schedules might be challenging for large networks and/or requires centralized calculations, but our framework can use any of the existing solutions proposed for this problem; see [24].

5) Static Requests and Configurations: Algorithm 1 can be directly applied when the requests and/or the system are static.

Corollary 1. *Assume that $\lambda_t = \lambda$ for all $t \in \mathbf{N}$. Replace C for X in Algorithm 1. Then, $R_T \leq \left(G + D^2\right)\sqrt{T}$.*

To see why this holds, note that if $x_t \in X$ and $\lambda_t = \lambda$, $\forall t$, then the constraint violation is always zero. Also, we get:

Corollary 2. *Assume that $\theta_t = \theta \in \Theta$ for all $t \in \mathbf{N}$. Then,*

$$\Phi_\theta(\bar{x}_T) - \min_{x \in X} \Phi_\theta(x) \leq (G^2 + D^2 + \sigma^2/2)/\sqrt{T}$$

$\bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$. The constraint violation is as in Theorem 1.

6) Multiple Grades & Stages: In our model, we use the term "grade" to refer to the different precision (quality, in general) in which the analytics are executed by each VNF. However, our model captures also the case of *horizontal grades*, where the number of chain stages can increase to improve the extracted information, at the expense of additional resource costs. For instance, consider an ensemble learning problem [26], where deciding to parse the data from one more classifier amounts to adding one more stage in our chain—but also paying more routing and computing costs. The early-exit option in our model can directly capture this scenario, and note that we can model (through f) cases where the remaining functions are executed at the sink node (or a nearby server), *and* cases where the remaining functions are omitted.

V. Performance Evaluation

We illustrate the results in Sec. III with a video analytics example. Our goal is to show how the service chain and system parameters (network size, stochastic, arrivals, varying costs) affect the bounds of Theorem 1.

TABLE I
Illustrating the mean Average Precision (mAP) (http://cocodataset.org/#detection-eval) of Yolo v3 depending on the compression and the the object-detection model.

	Yolo v3 (full)			**Yolo v3 (split)**		
Obj. size	1	1/2	1/4	1	1/2	1/4
small	0.12	0.07	0.09	0.20	0.19	0.20
medium	0.32	0.24	0.28	0.41	0.40	0.41
large	0.46	0.44	0.40	0.11	0.10	0.10

A. System Setup

Consider the network in Fig. 4, which consists of 5 nodes and 14 links. Node 1 receives two video streams that must be processed by a three-stage chain. The first function resizes/compresses the video frames; the second executes an object-detection algorithm; and the third displays the results on a monitor (the sink node). The functions can run in different modes, i.e., have multiple grades. The first function compresses the frames using three ratios, 1, 1/2, and 1/4 (indicating the portion of input), and the second runs the object detection application Yolo v3 [8] either in *full* or *split* mode. The *full* mode analyzes the entire image at once, with a neural network of size 512×512,[12] and the *split* mode cuts an image in half and analyzes each part with a 256×256 neural network, see example in Fig. 4.[13] In Table I we present our measurements showing how Yolo performs with images from the COCO data set [12] for different compression and detection modes. Observe that the *full* mode outperforms the *split* mode for large objects (and vice versa), and the prediction accuracy decreases with compression.[14]

We normalize the network parameter values to facilitate presentation. Video frames are samples from the COCO data set and arrive at node 1 at an average rate of 1 unit per time slot, and each image has size of 1 unit. To diversify the information in the video streams, we assume that the frames of the first and second stream contain *only* small and large objects respectively. Hence, the prediction accuracy of Yolo v3 will vary across these streams. We model the prediction accuracy (i.e., the rewards) of the object-detection algorithm as random variables that take values in $[0, 1]$ and have expected value as indicated in Table I. Nodes 1 and 5 can process 2 units of flow/commodity per time slot, while node 4 only 1 unit (i.e., node 4 can analyze at most one image per time slot).

B. Results

We illustrate the performance of Algorithm 1 when images arrive at a constant rate of 1 frame per second (deterministic) and in bursts of 10 frames according to a Poisson process

[12] $s \times s$, $s \in \mathbf{N}$ indicates the size of the input vector in the neural network. See [8] and https://pjreddie.com/darknet/yolo/ for more details.

[13] That happens because splitting an image into two parts may result in some objects being cut. This is particularly important for large objects.

[14] These values were measured for images in the COCO data set, and therefore, do not represent the performance of Yolo v3 for any type of image.

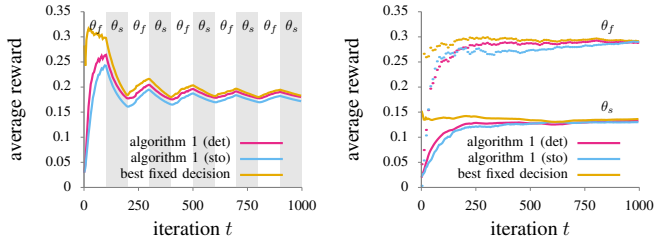

Fig. 5. Illustrating the average reward when the function in the service chain switches between θ_f and θ_s every 100 time slots (left). The figure on the right shows the rewards of the service chains when these are fixed.

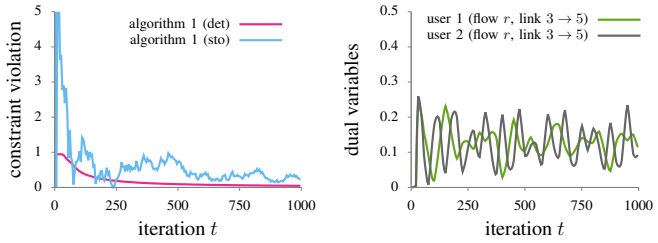

Fig. 6. Illustrating (left) the constraint violation of Algorithm 1 with a time-varying service chain; Fig. 5 (left). The figure on the right-hand-side shows the Lagrange multipliers associated to the commodity of type r in the link from node 3 to node 5.

(stochastic). To keep the exposition simple, we fix the compression ratio in the service chain to $1/2$, and play with the object-detection mode in the second function. We use θ_f and θ_s to denote the service chains that use Yolo in full and split mode, respectively. Also, to facilitate the interpretation of the figures, we assume there are no costs associated with routing and processing data. Hence, the objective function in Eq. (6) represents the value of executing the data analytics, and thus the results are shown in maximization terms (higher is better).

1) Regret convergence: Fig. 5 (left) shows the average reward of Algorithm 1 when the service chain switches between θ_f and θ_s every 100 slots. Observe that Algorithm 1 tracks the performance of the best fixed decision for both cases (*det* and *sto*), and that the gap (i.e., the regret) reduces with time. Note also that the reward of the best fixed decision oscillates because the service chains have different expected performances. The latter is shown in Fig. 5 (right), where we run the simulations with fixed service chains θ_f, θ_s (we show two simulations in the same plot). It is easy to see from Fig. 5 (right) that the reward of the best fixed decision varies over time since the output of the object-detection algorithm is randomized. Finally, we recall that Algorithm 1 does not know that images come from the COCO data set or when the service chains switch in the network, and yet, it is able to constantly "relearn" which is the best video stream to process by using observations F_t. **Key message:** Algorithm 1 has no-regret and can operate with minimal system information.

2) Constraint violation: Fig. 6 (left) shows the constraint violation obtained by Algorithm 1 in the previous simulation. Observe that in both cases (*det* and *sto*), $\|\frac{1}{T}\sum_{t=1}^{T}(Ax_t + Bx_t + \lambda_t)\|$ gets smaller as t increases. Note also that the stochastic case has a bursty behavior due to the nature of the arrivals. Fig. 6 (right) shows the dual variables or Lagrange multipliers of three commodities.[15] Note that those remain bounded and fluctuate due to the time-varying arrivals. **Key message:** Algorithm 1 satisfies the constraints, which guarantee that all traffic is served.

3) Network size and reward randomness: Now we show how Algorithm 1 is affected by the system parameters. We start by considering a larger network (Tutte graph [28] with 46 nodes and 138 directed links). We place the service chain functions in three random nodes and select the rest of the parameters as indicated at the beginning of the section. Fig.

[15]Lagrange multipliers can be seen as scaled queue backlogs [27, pp. 305].

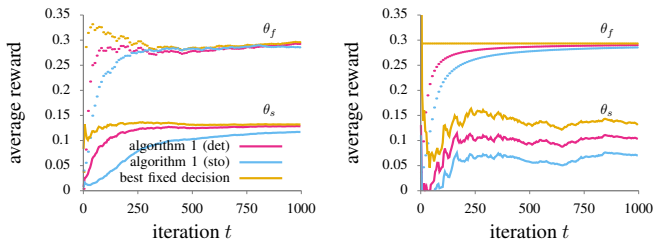

Fig. 7. Illustrating the average regret with a network of 46 nodes and 138 directed links. The figure on the right shows, in addition, the impact of the randomness in the service chain functions.

7 (left) shows the simulations results when the service chains are fixed. Observe that Algorithm 1 has the same behavior as in Fig. 5 (right); however, with slower convergence. Especially for the stochastic case. The reason for that is that constant σ^2 in the regret bound (Theorem 1) increases with the bustiness of the arrivals and the network size—matrices A and B, and the variance of the arrival process $\{\lambda_t\}$.

Next, we show that the randomness in the service chain functions affects the convergence time. We illustrate this by making the service chain θ_f deterministic (no randomness), and increasing four times the variance of the randomness in the θ_s service chain utilities. Observe from the Fig. 7 (right) that with θ_f, the convergence is smooth,[16] whereas with θ_s the convergence degrades heavily. In the last case, the loss of performance is because it is more difficult to learn the utilities. Also, by increasing the randomness, we are affecting constant D in the bound of Theorem 1. **Key message:** The randomness in the utilities slows down the learning, thus the convergence.

VI. Related Work

NFV Service Chains. VNFs are today the key building blocks of services networks and data centers [29]. Prior works have proposed heuristics or approximation algorithms for their deployment [7], [30], [31], and optimized routing [5]. However, in practice the network state and requests are unknown and vary with time. Using a learning approach, [32], [33] identify min-cost chains where the challenge lies in learning the flow volume, and [34] handles VNF migration costs. Yet, in analytic service chains: the performance depends

[16]That is because the algorithm obtains the "expected" performance of the functions in the service chain executing them once.

on the data and VNFs configuration; each stage might change the flow volume; and decisions are made under unknown demands, costs and performance observations.

In-Network Processing. More general, the idea of in-network processing gains significant traction, as more and more services involve both routing and (multi-stage) data processing. This has spurred efforts to extend classical network optimization algorithms. In [4] the authors summarized static multicommodity flow and middlebox placement problems with complete information, while [3] assumed random request arrivals and proposed a joint routing and processing backpressure-type policy (fixed and known objectives). Recently, [2], [6], proposed max-throughput processing and routing policies for unicast/multicast flows, while [14] considered also energy cost criteria. Here, we make the next step by maximizing the analytics performance, despite the unknown/time-varying requests and system state.

Online Convex Optimization. OCO was introduced for problems with changing functions and fixed inelastic constraints [16], and later extended for elastic/budget constraints [35], [36], where one can trade off regret for constraint violation [37]. Recently, [38] achieved $O(\sqrt{T})$ regret for inelastic i.i.d.-changing constraints; [39] considered elastic arbitrarily-changing constraints; and [40] proved that sublinear regret is not attainable for inelastic and arbitrary-changing constraints. Here, we consider the practical case where the constraints are elastic and changing, and we can observe only the outcome of x_t, F_t, and not function $f_t(x, \theta_t)$. By incorporating an effective sampling technique [17], we manage to obtain $O(\sqrt{T})$ regret and $O(1/\sqrt{T})$ average constraint violation.

OCO in Networks. OCO has shown promise for network optimization. Considering wireless networks, [41] designed a power control algorithm that adapts to fast-fading channels; [42] studied the effect of channel feedback on throughput; and [43] employed a UCB bandit algorithm to learn link activation schedules. Similarly for fog/cloud computing, [44], [45] proposed learning-based task offloading and service migration rules, while [46] used OCO to optimize reservation of resources. Our problem is different as it involves multiple processing stages, and we advance the algorithmic toolbox with a unifying primal-dual approach that handles time-varying unknown constraints *and* limited observations.

VII. Conclusions

As the number of network services requiring data processing increases, it becomes imperative to develop the necessary algorithmic tools for enabling the design of operational policies. In a first step towards this direction, we proposed an analytical framework that fuses online learning and resource allocation algorithms, and maximizes the performance of data analytics under very general network conditions; unknown request arrival statistics; VNFs with multiple grades; and limited information about their performance. Our algorithms achieve, asymptotically, zero optimality gap and constraint violation, as we demonstrated also in a series of experiments. This work, we believe, paves the road for exciting future studies, either focusing on the technical aspects of data analytics service chains, or to extending this optimization framework.

Appendix

To keep notation short, we define $E := A + B$ and let $d := r(n + 2m)$. Also, to streamline exposition, we use C instead of the time-varying set C_t as not allowing to transmit dummy data is equivalent to adding slack variables [47, pp. 131]. The difference, however, is that the slack variables do not have to satisfy the augmented flow conservation constraints. Also, there is not a cost associated with these variables, and therefore, they do not affect the network utility objective.

A. Preliminaries

Lemma 1 presents an inequality related to projected gradient descent, and Lemma 2 a relation between vectors in ℓ_2.

Lemma 1. *Consider update $x_{t+1} = \mathcal{P}_C(x_t - \rho v_t)$ where $x_t \in C$, $v_t \in \mathbf{R}^d$, and $\rho > 0$. For any $z \in C$, it holds*

$$\langle v_t, x_{t+1} - z\rangle \leq \frac{1}{\rho}\left(\|z - x_t\|^2 - \|z - x_{t+1}\|^2 - \|x_{t+1} - x_t\|^2\right)$$

Proof: Let $I_C(u)$ be the indicator function, i.e., $I_C(u) = 0$ if $u \in C$ and $+\infty$ otherwise. It is well-known (see, for example, [48, Sec. 2]) that

$$\begin{aligned} x_{t+1} &= \mathcal{P}_C(x_t - \rho v_t) \\ &= \arg\min_{u \in C}\{\langle v_t, u\rangle + (1/\rho)\|u - x_t\|^2\} \\ &= \arg\min_{u \in \mathbf{R}^d}\{\langle v_t, u\rangle + (1/\rho)\|u - x_t\|^2 + I_C(u)\} \quad (11) \end{aligned}$$

That is, the projected gradient descent update can be regarded as solving an unconstrained minimization problem. Since x_{t+1} is a minimizer of the last equation, we have that

$$\begin{aligned} 0 &\in \partial_{x_{t+1}}(\langle v_t, x_{t+1}\rangle + (1/\rho)\|x_{t+1} - x_t\|^2 + I_C(x_{t+1})) \\ &= v_t + (2/\rho)(x_{t+1} - x_t) + s \end{aligned}$$

where s is a vector in the normal cone of C at x_{t+1}, i.e., $s \in N_C(x_{t+1}) := \{s \in \mathbf{R}^d \mid \langle s, z - x_{t+1}\rangle \leq 0, \forall z \in C\}$. Next, multiply across by $(z - x_{t+1})$ and use the fact that $\langle s, z - x_{t+1}\rangle \leq 0$ to obtain $0 \leq \langle v_t, z - x_{t+1}\rangle + (2/\rho)\langle x_{t+1} - x_t, z - x_{t+1}\rangle$. Finally, observe that $\langle x_{t+1} - x_t, z - x_{t+1}\rangle = \langle z, x_{t+1}\rangle - \langle z, x_t\rangle + \langle x_{t+1}, x_t\rangle - \|x_{t+1}\|^2 = \frac{1}{2}\left(\|z - x_t\|^2 - \|z - x_{t+1}\|^2 - \|x_{t+1} - x_t\|^2\right)$ and so the stated result. ∎

Lemma 2. *Let $u, v \in \mathbf{R}^d$ and $\mu > 0$. It holds $\langle v, u\rangle - \frac{1}{\mu}\|u\|^2 \leq \mu\|v\|^2$.*

Proof: The proof follows from the definition of the convex conjugate function and the fact that the ℓ_2-norm is convex and self dual. That is, $\langle v, u\rangle - \frac{1}{\mu}\|u\|^2 \leq \sup_{u \in \mathbf{R}^d}\langle v, u\rangle - \frac{1}{\mu}\|u\|^2 := (1/\mu)\|\mu v\|_*^2 = \mu\|v\|^2$. See, for example, [49, Sec. 11-A] and [47, Ex. 3.27]. ∎

B. Proof of Theorem 1

We start by presenting the proof of Proposition 1. Let $\kappa : \mathbf{R}_+ \to \mathbf{R}$ be a linear function with $\kappa \in \mathbf{R}$, and let $K(x)$ be a random variable with $\mathbf{E}(K(x)) = \kappa(x) = \kappa x$. Trivially, $\kappa = \mathbf{E}[K/x]$ (i.e., the gradient of κ) when $x \neq 0$. If $x = 0$, then $\kappa(x)$ is not differentiable and we cannot compute a (sub)gradient by dividing by x. Instead, we use the fact that the cost/reward of *not-processing* a flow is 0, and so the all-zeroes vector is a subgradient of κ at $x = 0$.

Proof outline. We start by establishing the upper bound on the regret by using the stochastic primal and dual updates (Lemmas 3 and 4). In Lemma 5, we prove the second claim in the theorem, and in Lemma 6, characterize constant Γ.

Lemma 3 (Primal online gradient descent)**.** *The regret is upper bounded by* $R_T \leq G^2 T\rho + \frac{D^2}{\rho} + \sum_{t=1}^{T}\langle y_t, E(z - x_{t+1})\rangle$.

Proof: Let $v_t = \nabla\Phi_t + \langle y_t, E\rangle = \mathbf{E}[F_t \circ \varphi(x_t) + \langle y_t, E\rangle]$ in Lemma 1. Rearranging terms and adding $\langle \nabla\Phi_t, x_t - x_{t+1}\rangle$ to both sides of the bound in Lemma 1 we have $\langle \nabla\Phi_t, x_t - z\rangle \leq \frac{1}{\rho}(\|z - x_t\|^2 - \|z - x_{t+1}\|^2 - \|x_{t+1} - x_t\|^2) + \langle \nabla\Phi_t, x_t - x_{t+1}\rangle + \langle y_t, E(z - x_{t+1})\rangle$. By Lemma 2, $\langle \nabla\Phi_t, x_t - x_{t+1}\rangle - (1/\rho)\|x_{t+1} - x_t\|^2 \leq \rho\|\nabla\Phi_t\|^2$ and therefore $\langle \nabla\Phi_t, x_t - z\rangle \leq \langle y_t, E(z - x_{t+1})\rangle + \rho\|\nabla\Phi_t\|^2 + (1/\rho)(\|z - x_t\|^2 - \|z - x_{t+1}\|^2)$. Finally, since $\langle \nabla\Phi_t, x_t - z\rangle = \Phi_t(x_t) - \Phi_t(z)$, we can sum from $t = 1, \ldots, T$ and use the fact that $\max_{u,v\in C}\|u - v\| \leq D$ and $\|\nabla\Phi_t\| \leq G$ to obtain the stated result. ∎

Lemma 4 (Dual online gradient descent)**.** *For any vector* $z \in X$*, it holds* $\mathbf{E}[\sum_{t=1}^{T}\langle y_t, E(z - x_{t+1})\rangle] \leq \rho\frac{\sigma^2}{2}T$.

Proof: The third term in the RHS of the bound in Lemma 3 can be written as $\sum_{t=1}^{T}\langle y_{t+1}, E(z - x_{t+1})\rangle = \sum_{t=1}^{T}\langle y_t, Ez + \lambda_t\rangle - \sum_{t=1}^{T}\langle y_{t+1}, Ex_{t+1} + \lambda_t\rangle$. The first term in RHS of last equation is upper bounded by zero in expectation. Observe that since $z \in X$ (i.e., it is a feasible point), then $\mathbf{E}[\sum_{t=1}^{T}\langle y_t, Ez + \lambda_t\rangle] = \sum_{t=1}^{T}\langle y_t, Ez + \mathbf{E}[\lambda_t]\rangle \leq 0$. Hence, $\mathbf{E}[\sum_{t=1}^{T}\langle y_t, E(z - x_{t+1})\rangle] \leq -\mathbf{E}[\sum_{t=1}^{T}\langle y_t, Ex_{t+1} + \lambda_{t+1}\rangle]$.

We proceed to upper bound the RHS of the last equation. Observe that

$$\begin{aligned}
&\|y_{t+1}\|^2 - \|y_t\|^2 \qquad (12)\\
&\quad = \|[y_t + \rho(Ex_{t+1} + \lambda_{t+1})]^+\|^2 - \|y_t\|^2\\
&\quad \leq \|y_t + \rho(Ex_{t+1} + \lambda_{t+1})\|^2 - \|y_t\|^2\\
&\quad \leq \rho^2\|Ex_{t+1} + \lambda_{t+1}\|^2 + 2\rho\langle y_t, Ex_{t+1} + \lambda_{t+1}\rangle
\end{aligned}$$

Rearrange terms, divide across by 2ρ, and use the fact that the gradients of the dual function are bounded $-\langle y_t, Ex_{t+1} + \lambda_{t+1}\rangle \leq \frac{1}{2\rho}\left(\|y_t\|^2 - \|y_{t+1}\|^2\right) + \rho\frac{\sigma^2}{2}$. Finally, sum from $t = 1, \ldots, T$ to obtain $-\sum_{t=1}^{T}\langle y_t, Ex_{t+1} + \lambda_{t+1}\rangle \leq (2\rho)^{-1}\left(\|y_1\|^2 - \|y_T\|^2\right) + \rho(\sigma^2/2)T \leq \rho(\sigma^2/2)T$ where the last equation follows since $y_1 = 0$ by assumption and $\|y_t\| \geq 0$ for all $t \in \mathbf{N}$. ∎

Lemma 5 (Feasibility)**.** *Consider update* $y_{t+1} = [y_t + \rho(Ex_{t+1} + \lambda_{t+1})]^+$*. Then,* $\|[\frac{1}{T}\sum_{t=1}^{T}(Ex_t + \lambda_t)]^+\| \leq \frac{\|y_{T+1}\|}{\rho T}$.

Proof: Observe that $y_{t+1} = [y_t + \rho(Ex_{t+1} + \lambda_{t+1})]^+ \succeq y_t + \rho(Ex_{t+1} + \lambda_{t+1})$. Rearrange terms and sum from $t = 1, \ldots, T$ to obtain $y_{T+1} - y_1 \succeq \sum_{t=1}^{T}(Ex_t + \lambda_t)$. Use the fact that $y_1 = 0$, $y_t \succeq 0$ for all $t \in \mathbf{N}$ to write $y_{T+1}/\rho \succeq [\sum_{t=1}^{T}(Ex_t + \lambda_t)]^+$. Taking norms on both sides and dividing by T yields the result. ∎

We show that $\|y_t\|$ is uniformly bounded in expectation.

Lemma 6 (Bounded dual variables)**.** *For any* $T \in \mathbf{N}$ *and* $t \in \{1, \ldots, T\}$*, we have that* $\mathbf{E}[\|y_t\|] \leq \Gamma$ *where*

$$\Gamma := \sqrt{\left(\frac{1}{rm\eta}(\sigma^2 + 2GD + D^2/2)\right)^2 + 2\sigma^2 + 4GD + D^2}.$$

Proof: We proceed to show that y_t is a bounded vector for all $t \in \mathbf{N}$. Observe that

$$\begin{aligned}
&\langle y_t, Ex_{t+1} + \lambda_{t+1}\rangle\\
&\quad = \langle y_t, Ex_{t+1} + \lambda_{t+1}\rangle + \langle \nabla\Phi_t, x_{t+1} - x_{t+1}\rangle\\
&\quad = \langle \nabla\Phi_t + \langle y_t, E\rangle, x_{t+1}\rangle + \langle y_t, \lambda_{t+1}\rangle - \langle \nabla\Phi_t, x_{t+1}\rangle\\
&\text{(a)} \leq \langle \nabla\Phi_t + \langle y_t, E\rangle, z\rangle + \langle y_t, \lambda_{t+1}\rangle - \langle \nabla\Phi_t, x_{t+1}\rangle\\
&\qquad + \rho^{-1}\left(\|z - x_t\|^2 - \|z - x_{t+1}\|^2 - \|x_{t+1} - x_t\|^2\right)\\
&\quad = \langle \nabla\Phi_t, z - x_{t+1}\rangle + \langle y_t, Ez + \lambda_{t+1}\rangle\\
&\qquad + \rho^{-1}\left(\|z - x_t\|^2 - \|z - x_{t+1}\|^2 - \|x_{t+1} - x_t\|^2\right)\\
&\text{(b)} \leq 2GD + \langle y_t, Ez + \lambda_{t+1}\rangle + \tfrac{1}{\rho}\left(\|z - x_t\|^2 - \|z - x_{t+1}\|^2\right)
\end{aligned}$$

where (a) follows from Lemma 1 and holds for any $z \in C$; and (b) by dropping $\rho^{-1}\|x_{t+1} - x\|^2$ and $\langle \nabla\Phi_t, z - x_{t+1}\rangle \leq \|\nabla\Phi_t\|\|z - x_{t+1}\| \leq 2GD$ by the Cauchy-Schwartz inequality. Next, use Eq. (12) and take expectations on both sides w.r.t. λ_t to obtain

$$\begin{aligned}
&\mathbf{E}[\|y_{t+1}\|^2 - \|y_t\|^2]\\
&\quad \leq \rho^2\sigma^2 + 4GD\rho - 2\eta\rho\|y_t\|_1 + \|z - x_t\|^2 - \|z - x_{t+1}\|^2
\end{aligned}$$

where the third term in the RHS of the last equation follows by selecting a $z \in C$ that satisfies the Slater condition. Now consider a span of k iterations. We have $\mathbf{E}[\|y_{t+k}\|^2 - \|y_t\|^2] \leq \rho^2\sigma^2 k + 4\rho GDk - 2\rho\eta\sum_{i=t}^{t+k}\|y_i\|_1 + \|z - x_t\|^2 - \|z - x_{t+k}\|^2$. Hence,

$$\begin{aligned}
&\mathbf{E}[\|y_{t+k}\|^2 - \|y_t\|^2] \qquad (13)\\
&\quad \leq 2\rho^2\sigma^2 k + 4\rho GDk - 2\rho\eta\textstyle\sum_{i=t}^{t+k}\|y_i\|_1 + D^2.
\end{aligned}$$

Next, note that if $\frac{1}{2\eta\rho}(2\rho^2\sigma^2 + 4\rho GD + D^2/k)k \leq \sum_{i=t}^{t+k}\|y_i\|_1$ then $\mathbf{E}[\|y_{t+k}\|^2 - \|y_t\|^2] \leq 0$. The latter will be the case when $\|y_i\|_1 \geq \frac{1}{\eta}(\sigma^2 + 2GD + D^2/(2k\rho))$ for all $i = t, \ldots, t + k$ since $k \geq 1$ and $\rho \leq 1$. Selecting $k := \lceil 1/\rho \rceil$ we have $\|y_i\|_1 \geq \frac{1}{\eta}(\sigma^2 + 2GD + D^2/2)$. Using the equivalence of the norms (i.e., $\|y\|_1 \leq rm\|y\|$ since $y \in \mathbf{R}^{rm}$), we have $\|y_i\| \geq \chi := \frac{1}{rm\eta}(\sigma^2 + 2GD + D^2/2)$. Now, fix a $t \in \{1, \ldots, T\}$ such that $\|y_t\| := \chi$ and suppose that $\|y_i\| \geq \chi$ for all $i = 1, \ldots, k$. Then, from Eq. (13) we can write

$$\mathbf{E}[\|y_{t+k}\|^2] \leq \chi^2 + 2\sigma^2 + 4GD + D^2$$

since $\rho k \leq 1$ and $\rho \leq 1$. Finally, by taking square roots on both sides we obtain the stated result. ∎

ACKNOWLEDGEMENTS

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 795244.

This work was supported by Science Foundation Ireland under Grant No. 17/CDA/4760.

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

This work was sponsored by the NSF Award 1815676.

REFERENCES

[1] E. Siow, T. Tiropanis, and W. Hall, "Analytics for the internet of things: A survey," *ACM Comput. Surv.*, vol. 51, no. 4, 2018.

[2] J. Zhang, A. Sinha, J. Llorca, A. Tulino, and E. Modiano, "Optimal control of distributed computing networks with mixed-cast traffic flows," in *Proc. of IEEE INFOCOM*, 2018, pp. 1880–1888.

[3] A. Destounis, G. S. Paschos, and I. Koutsopoulos, "Streaming big data meets backpressure in distributed network computation," in *Proc. of IEEE INFOCOM*, 2016.

[4] M. Charikar, Y. Naamad, J. Rexford, and X. K. Zou, "Multi-commodity flow with in-network processing," in *Algorithmic Aspects of Cloud Computing*, Y. Disser and V. S. Verykios, Eds. Cham: Springer International Publishing, 2019, pp. 73–101.

[5] Z. Cao, S. S. Panwar, M. Kodialam, and T. V. Lakshman, "Enhancing mobile networks with software defined networking and cloud computing," *IEEE/ACM Trans. on Networking*, vol. 25, no. 3, 2017.

[6] H. Feng, J. Llorca, A. M. Tulino, and A. F. Molisch, "Optimal dynamic cloud network control," *IEEE/ACM Transactions on Networking*, vol. 26, no. 5, pp. 2118–2131, Oct. 2018.

[7] T. Lukovski, M. Rost, and S. Schmid, "Approximate and incremental network function placement," *Journal of Parallel and Distributed Computing*, vol. 120, 2018.

[8] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. Berg, "Ssd: Single shot multibox detector," in *Proc. of ECCV*, 2016, pp. 21–37.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[11] F. N. Iandola *et al.*, "Densenet: Implementing efficient convnet descriptor pyramids," *CoRR, http://arxiv.org/abs/1404.1869*, 2014.

[12] X. Chen, H. Fang, T. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick, "Microsoft COCO captions: Data collection and evaluation server," *CoRR*, vol. abs/1504.00325, 2015.

[13] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proc. of IEEE CVPR*, 2015.

[14] V. Valls, G. Iosifidis, and T. Salonidis, "Maximum lifetime analytics in iot networks," in *Proc. of IEEE INFOCOM*, 2019, pp. 1369–1377.

[15] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *Proc. of IEEE INFOCOM*, 2018, pp. 1421–1429.

[16] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. of ICML*, 2003.

[17] A. D. Flaxman, A. T. Kalai, and H. B. McMahan, "Online convex optimization in the bandit setting: Gradient descent without a gradient," in *Proc. of ACM-SIAM SODA*, 2005.

[18] R. Rockafellar, *Network flows & monotropic optimization*. Wiley, 1984.

[19] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *IEEE/ACM Trans. on Networking*, vol. 15, no. 6, 2007.

[20] Y. Nesterov and V. Shikhman, "Dual subgradient method with averaging for optimal resource allocation," *European Journal of Operational Res.*, vol. 270, no. 3, 2018.

[21] V. Valls and D. J. Leith, "Max-weight revisited: Sequences of nonconvex optimizations solving convex optimizations," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2676–2689, 2016.

[22] V. Valls and D. J. Leith, "A convex optimization approach to discrete optimal control," *IEEE Transactions on Automatic Control*, vol. 64, no. 1, pp. 35–50, Jan 2019.

[23] E. Hazan, "Introduction to online convex optimization," *Foundations and Trends in Optimization*, 2016.

[24] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*. Foundations and Trends in Networking, 2006.

[25] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[26] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1-2, 2010.

[27] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on cross-layer congestion control in wireless networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 14, no. 2, pp. 302–315, April 2006.

[28] "Tutte graph," *[Online]: http://mathworld.wolfram.com/TuttesGraph.html*.

[29] F. Esposito, I. Matta, and V. Ishakian, "Slice embedding solutions for distributed service architectures," *ACM Comp. Surv.*, vol. 46, no. 1, 2013.

[30] S. Vassilaras *et al.*, "The algorithmic aspects of network slicing," *IEEE Communications Magazine*, vol. 55, no. 8, 2017.

[31] A. Fischer *et al.*, "Virtual network embedding: A survey," *IEEE Commun. Surveys & Tutorials*, vol. 15, no. 4, 2013.

[32] X. Chen *et al.*, "Multi-timescale online optimization of network function virtualization for service chaining," *IEEE Trans. on Mobile Comp.*, 2019.

[33] X. Wang, C. Wu, F. Le, and F. C. M. Lau, "Proactive vnf provisioning with multi-timescale cloud resources: Fusing online learning and online optimization," in *Proc. of IEEE INFOCOM*, 2017.

[34] M. Shi, X. Lin, S. Fahmy, and D. H. Shin, "Competitive online convex optimization with switching costs and ramp constraints," in *Proc. of IEEE INFOCOM*, 2018.

[35] M. Mahdavi, T. Yang, R. Jin, S. Zhu, and J. Yi, "Stochastic gradient descent with only one projection," in *Proc. of NIPS*, 2012, pp. 494–502.

[36] A. Cotter, M. Gupta, and J. Pfeifer, "A light touch for heavily constrained sgd," in *29th Annual Conference on Learning Theory*, ser. Proceedings of Machine Learning Research, vol. 49, 2016, pp. 729–771.

[37] R. Jenatton, J. C. Huang, and C. Archambeau, "Adaptive algorithms for online convex optimization with long-term constraints," in *ICML*, 2016.

[38] H. Yu, M. Neely, and X. Wei, "Online convex optimization with stochastic constraints," in *Proc. of NIPS*, 2017, pp. 1428–1438.

[39] N. Liakopoulos, A. Destounis, G. Paschos, T. Spyropoulos, and P. Mertikopoulos, "Cautious regret minimization: Online optimization with long-term budget constraints," in *Proc. of ICML*, 2019.

[40] S. Mannor, J. N. Tsitsiklis, and J. Yuan Yu, "Online learning with sample path constraints," *Journal of Machine Learning Research*, Mar 2009.

[41] H. Yu and M. J. Neely, "Learning aided optimization for energy harvesting devices with outdated state information," in *Proc. of IEEE INFOCOM*, 2018.

[42] A. Marcastel, E. V. Belmega, P. Mertikopoulos, and I. Fijalkow, "Online power optimization in feedback-limited, dynamic and unpredictable iot networks," *IEEE Trans. Signal Processing*, vol. 67, no. 11, 2019.

[43] T. Stahlbuhk, B. Shrader, and E. Modiano, "Learning algorithms for scheduling in wireless networks with unknown channel statistics," in *Proc. of ACM Mobihoc*, 2018.

[44] T. Chen, Q. Ling, Y. Shen, and G. B. Giannakis, "Heterogeneous online learning for thing-adaptive fog computing in iot," *IEEE Internet of Things Journal*, vol. 6, no. 5, 2018.

[45] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *Proc. of IEEE INFOCOM*, 2018.

[46] N. Liakopoulos *et al.*, "No regret in cloud resources reservation with violation guarantees," in *Proc. of IEEE INFOCOM*, 2019.

[47] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[48] A. Beck and M. Teboulle, "Mirror descent and nonlinear projected subgradient methods for convex optimization," *Operations Research Letters*, vol. 31, no. 3, pp. 167 – 175, 2003.

[49] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*. Berlin: Springer: Grundlehren der Math. Wissenschaften., 1998, vol. 317.