Bandit Convex Optimization for Scalable and Dynamic IoT Management

Tianyi Chen and Georgios B. Giannakis

Abstract—The present paper deals with online convex optimization involving both time-varying loss functions, and time-varying constraints. The loss functions are not fully accessible to the learner, and instead only the function values (a.k.a. bandit feedback) are revealed at queried points. The constraints are revealed after making decisions, and can be instantaneously violated, yet they must be satisfied in the long term. This setting fits nicely the emerging online network tasks such as fog computing in the Internet-of-Things (IoT), where online decisions must flexibly adapt to the changing user preferences (loss functions), and the temporally unpredictable availability of resources (constraints). Tailored for such human-in-the-loop systems where the loss functions are hard to model, a family of bandit online saddle-point (BanSaP) schemes are developed, which adaptively adjust the online operations based on (possibly multiple) bandit feedback of the loss functions, and the changing environment. Performance here is assessed by: i) dynamic regret that generalizes the widely used static regret; and, ii) fit that captures the accumulated amount of constraint violations. Specifically, BanSaP is proved to simultaneously yield sub-linear dynamic regret and fit, provided that the best dynamic solutions vary slowly over time. Numerical tests in fog computation offloading tasks corroborate that our proposed BanSaP approach offers competitive performance relative to existing approaches that are based on gradient feedback.

Index Terms—Online learning, bandit convex optimization, saddle-point method, Internet of Things, mobile edge computing.

I. INTRODUCTION

Internet-of-Things (IoT) envisions an intelligent infrastructure of networked smart devices offering task-specific monitoring and control services [2]. Leveraging advances in embedded systems, contemporary IoT devices are featured with small-size and low-power designs, but their computation and communication capabilities are limited. A prevalent solution during the past decade was to move computing, control, and storage resources to the remote cloud (a.k.a. data centers). Yet, the cloud-based IoT architecture is challenged by high latency due to directly communicating with the cloud, which certainly prevents real-time applications [3]. Along with other features of IoT, such as extreme heterogeneity and unpredictable dynamics, the need arises for innovations in network design and management to allow for adaptive online service provisioning, subject to stringent delay constraints [4].

From the network design vantage point, fog is viewed as a promising architecture for IoT that distributes computation, communication, and storage closer to the end IoT users, along the cloud-to-things continuum [3]. In the fog computing paradigm, service provisioning starts at the network edge, e.g., smartphones, and high-tech routers, and only a portion of tasks will be offloaded to the powerful cloud for further processing (a.k.a. computation offloading) [5]–[7]. Existing approaches for computation offloading either focus on time-invariant static settings, or, rely on stochastic optimization approaches such as Lyapunov optimization to deal with time-varying cases; see [8] and references therein. Nevertheless, static settings cannot capture the changing IoT environment, and the stationarity commonly assumed in stochastic optimization literature may not hold in practice, especially when the stochastic process involves human participation as in IoT. From the management perspective, online network control, which is robust to non-stationary dynamics and amenable to low-complexity implementations, remains a largely uncharted territory [6], [8].

Indeed, the primary goal of this paper is an algorithmic pursuit of online network optimization suitable for emerging tasks in IoT. Focusing on such algorithmic challenges, online convex optimization (OCO) is a promising methodology for sequential tasks with well-documented merits, especially when the sequence of convex costs varies in an unknown and possibly adversarial manner [9]. Aiming to empower traditional fog management policies with OCO, most available OCO works benchmark algorithms with a static regret, which measures the difference of costs (a.k.a. losses) between the online solution and the best static solution in hindsight [10], [11]. However, static regret is not a comprehensive performance metric in dynamic settings such as those encountered with IoT [12].

Recent works extend the analysis of static to that of dynamic regret [12], [13], but they deal with time-invariant constraints that cannot be violated instantaneously. Tailored for fog computing setups that need flexible adaptation of online decisions to dynamic resource availability, OCO with time-varying constraints was first studied in [14], along with its adaptive variant in [15], and the optimal regret bound in this setting was first established in [16]. Yet, the approaches in [14]–[16] remain operational under the premise that the loss functions are explicitly known, or, their gradients are readily available. Clearly, none of these two assumptions can be easily satisfied in IoT settings, because i) the loss function capturing user dissatisfaction, e.g., service latency or reliability, is hard to model in dynamic environments; and, ii) even if modeling is possible in theory, the low-power IoT devices may not afford the complexity of running statistical learning tools such as deep neural networks online.

In this context, targeting a gradient-free efficient solution,
alternative online schemes have been advocated leveraging point-wise values of loss functions (partial-information feedback) rather than their gradients (full-information feedback). They are termed bandit convex optimization (BCO) in machine learning [17]–[20], or referred as zeroth-order schemes in optimization circles [21], [22]. While [17]–[22] employed BCO with time-invariant constraints that cannot be violated instantaneously, the long-term effect of such instantaneous violations was studied in [23], where the focus is still on static regret and time-invariant constraints. Nevertheless, [17]–[22] cannot be implemented without knowing the instantaneous constraints, and the performance guarantees relative to the best dynamic benchmark have not been characterized in [17]–[23]. Building on full-information precursors [14]–[16], the present paper broadens the scope of BCO to the regime with time-varying constraints, and proposes a class of online algorithms termed online bandit saddle-point (BanSaP) approaches. Also worth mentioning is that the regret-fit tradeoff of BanSaP markedly improves that in [15] for the special case with full-information feedback, and that in [23] for the special case with time-invariant constraints. With an eye on managing IoT with limited information, our contribution is the incorporation of long-term and time-varying constraints to expand the scope of BCO; see a summary in Table I.

In a nutshell, relative to existing works, the main contributions of the present paper are summarized as follows.

c1) We generalize the standard BCO framework with only time-varying costs [17], [18], to account for both time-varying costs and constraints. Performance here is established relative to the best dynamic benchmark, via metrics that we term static regret and time-invariant constraints. Nevertheless, [17]–[22] cannot be implemented without knowing the instantaneous constraints, and the performance guarantees relative to the best dynamic benchmark have not been characterized in [17]–[23]. Building on full-information precursors [14]–[16], the present paper broadens the scope of BCO to the regime with time-varying constraints, and proposes a class of online algorithms termed online bandit saddle-point (BanSaP) approaches. Also worth mentioning is that the regret-fit tradeoff of BanSaP markedly improves that in [15] for the special case with full-information feedback, and that in [23] for the special case with time-invariant constraints. With an eye on managing IoT with limited information, our contribution is the incorporation of long-term and time-varying constraints to expand the scope of BCO; see a summary in Table I.

c2) We develop a class of BanSaP algorithms to tackle this novel BCO problem, and analytically establish that BanSaP solvers yield simultaneously optimal sub-linear dynamic regret and fit, given that the accumulated variations of per-slot minimizers are known to grow sub-linearly with time (Section IV).

c3) Our BanSaP algorithms are applied to computational offloading tasks emerging in IoT management, and simulations under various network sizes further demonstrate that the BanSaP solvers outperform the popular algorithm with bandit feedback, and have comparable performance relative to full-information alternatives (Section V).

Notation. $(\cdot)^\top$ stands for vector and matrix transposition, and $\|x\|$ denotes the $\ell_2$-norm of a vector $x$. Inequalities for vectors $x > 0$, and the projection $[a]^+ := \max\{a, 0\}$ are entry-wise.

II. BANDIT ONLINE LEARNING WITH CONSTRAINTS

In this section, a generic BCO formulation with long-term and time-varying constraints will be introduced, along with its real-world application in IoT management.

### A. Online learning with constraints under partial feedback

Before introducing BCO with long-term constraints, we begin with the classical BCO setting, where constraints are time-invariant, and must be strictly satisfied [17], [18], [20]. Akin to its full-information counterpart [9], [10], BCO can be viewed as a repeated game between a learner and nature. Consider that time is discrete and indexed by $t$. Per slot $t$, a learner selects an action $x_{t}$ from a convex set $X \subseteq \mathbb{R}^{d}$, and subsequently nature chooses a loss function $f_{t}(\cdot): \mathbb{R}^{d} \rightarrow \mathbb{R}$ through which the learner incurs a loss $f_{t}(x_{t})$. The convex feasible set $X$ has a-priori known and fixed over the entire time horizon. Different from the OCO setup, at the end of each slot, only the value of $f_{t}(x_{t})$ rather than the form of $f_{t}(x)$ is revealed to the learner in BCO. Although this standard BCO setting is appealing to various applications such as online end-to-end routing [24] and task assignment [25], it does not account for potential variations of (possibly unknown) constraints, and does not deal with constraints that can possibly be satisfied in the long term rather than a slot-by-slot basis [14], [16], [23].

Online optimization with time-varying and long-term constraints is well motivated for applications from power control in wireless communication [26], geographical load balancing in cloud networks [14], [27], to computation offloading in fog computing [28], [29]. Motivated by these dynamic network management tasks, our recent works [14], [15] studied OCO with time-varying constraints in full information setting, where the gradient feedback is available. Complementing [14] and [15], the present paper broadens the applicability of BCO to the regime with time-varying long-term constraints.

Specifically, we consider that per slot $t$, a learner selects an action $x_{t}$ from a known and fixed convex set $X \subseteq \mathbb{R}^{d}$, and then nature chooses not only a loss function $f_{t}(\cdot): \mathbb{R}^{d} \rightarrow \mathbb{R}$, but also a time-varying penalty function $g_{t}(\cdot): \mathbb{R}^{d} \rightarrow \mathbb{R}^{N}$. The later gives rise to the time-varying constraint $g_{t}(x_{t}) \preceq 0$, which is driven by the unknown application-specific dynamics. Similar to the standard BCO setting, only the value of $f_{t}(x_{t})$ at the queried point $x_{t}$ is revealed to the learner here; but different from the standard BCO setting, besides $X$, the constraint $g_{t}(x_{t}) \preceq 0$ needs to be carefully taken care of. And the fact that $g_{t}$ is unknown to the learner when performing her/his decision, makes it impossible to satisfy in every time slot. Hence, a more realistic goal here is to find a sequence of solutions $\{x_{t}\}$ that minimizes the aggregate loss, and ensures that the constraints $\{g_{t}(x_{t}) \preceq 0\}$ are satisfied in the long term on average. Specifically, extending the BCO framework [17]–[19] to accommodate such time-varying constraints, we consider the following online optimization problem

$$\begin{align*}
\min_{\{x_{t} \in X, y_{t}\}} \quad & \sum_{t=1}^{T} f_{t}(x_{t}) \quad \text{s. to} \quad \sum_{t=1}^{T} g_{t}(x_{t}) \preceq 0
\end{align*}$$

where $T$ is the entire time horizon, $x_{t} \in \mathbb{R}^{d}$ is the decision variable, $f_{t}$ represents the cost function, $g_{t} := [g_{t}^{1}, \ldots, g_{t}^{N}]^{\top}$ denotes the constraint function with $n$th entry $g_{t}^{n}(\cdot): \mathbb{R}^{d} \rightarrow \mathbb{R}$, and $X \subseteq \mathbb{R}^{d}$ is a convex set. In the current setting, we assume that only the values of loss function are available at queried points since e.g., its complete form related to user experience is hard to approximate, but the constraint function

<table>
<thead>
<tr>
<th>Reference</th>
<th>Benchmark</th>
<th>Constraints</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>[9]–[11]</td>
<td>Static</td>
<td>Fixed and strict</td>
<td>Gradient</td>
</tr>
<tr>
<td>[12], [13]</td>
<td>Dynamic</td>
<td>Fixed and strict</td>
<td>Gradient</td>
</tr>
<tr>
<td>[16]</td>
<td>Static</td>
<td>Varying and long-term</td>
<td>Gradient</td>
</tr>
<tr>
<td>[14], [15]</td>
<td>Dynamic</td>
<td>Varying and long-term</td>
<td>Gradient</td>
</tr>
<tr>
<td>[23]</td>
<td>Static</td>
<td>Fixed and long-term</td>
<td>Gradient, Value</td>
</tr>
<tr>
<td>[17]–[22]</td>
<td>Static</td>
<td>Varying and long-term</td>
<td>Function value</td>
</tr>
</tbody>
</table>

This work Dynamic Varying and long-term Function value

IEEE INTERNET OF THINGS JOURNAL (TO APPEAR) 2

2327-4662 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
is revealed to the learner as it represents measurable physical requirements e.g., power budget, and data flow conservation constraints. Before the algorithm development in Section III and performance analysis in Section IV, we will introduce a motivating example of fog computing in IoT.

B. Motivating setup: mobile fog computing in IoT

The online computational offloading task of fog computing in IoT [5], [6], [8] takes the form of BCO with long-term constraints (1). Consider a mobile network with a sensor layer, a fog layer, and a cloud layer [3], [4]. The sensor layer contains heterogeneous low-power IoT devices (e.g., wearable watches and smart cameras), which do not have enough computational capability, and usually offload their collected data to the local fog nodes (e.g., smartphones and high-tech routers) in the fog layer for further processing [30]. The fog layer consists of \( N \) nodes in the set \( \mathcal{N} := \{1, \ldots, N\} \) with moderate processing capability; thus, part of workloads will be collaboratively processed by the local fog servers to meet the stringent latency requirement, and the rest will be offloaded to the remote data center in the cloud layer [6]; also see Fig. 1.

Per time \( t \), each fog node \( n \) collects data requests \( b^n_t \) from all its nearby sensors. Once receiving these requests, node \( n \) has three options: i) offloading the amount \( z^n_t \) to the remote data center; ii) offloading the amount \( y_t^{nk} \) to each of its nearby node \( k \) for collaborative computing; and, iii) locally processing the amount \( y_t^{nm} \) according to its resource availability. The optimization variable \( x_t \) in this case consists of the cloud offloading, local offloading, and local processing amounts; i.e., \( x_t := \{z_t^n, z_t^{nk}, y_t^{nm}\} \). Assuming that each fog node has a data queue to buffer unserved workloads, the instantaneously served workloads (offloading plus processing) is not necessarily equal to the data arrival rate. Instead, a long-term constraint is common to ensure that the cumulative amount of served workloads is no less than the arrived amount at each node \( n \) over time [26]

\[
\sum_{t=1}^{T} g^n_t(x_t) := \sum_{t=1}^{T} \left( b^n_t + \sum_{k \in \mathcal{N}_n^\text{in}} y_{kn}^t - \sum_{k \in \mathcal{N}_n^\text{out}} y_{kn}^t - z^n_t - y_t^{nm} \right) \leq 0
\]

(2)

where \( \mathcal{N}_n^\text{in} \) and \( \mathcal{N}_n^\text{out} \) represent the sets of fog nodes with incoming links to node \( n \) and those with outgoing links from node \( n \), respectively. The bandwidth limit of communication link (e.g., wireline) from fog node \( n \) to the remote cloud is \( z^n; \) the limit of the transmission link (e.g., wireless) from node \( n \) to its neighbor \( k \) is \( y_{nk}^t \), and the computation capability of node \( n \) is \( y_{nm}^t \). With \( \mathcal{X} \) collecting all the aforementioned limits, the feasible region can be expressed by \( x_t \in \mathcal{X} := \{0 \leq x_t \leq \bar{x}\} \).

Performance is assessed by the user dissatisfaction of the online processing and offloading decisions, e.g., aggregate delay [2], [4]. Specifically, as the computation delay is usually negligible for data centers with thousands of high-performance servers, the latency for cloud offloading amount \( z^n_t \) is mainly due to the communication delay, which is denoted as a time-varying cost \( c^n_t(z^n_t) \) depending on the unpredictable network congestion during slot \( t \). Likewise, the communication delay of the local offloading decision \( y_t^{nk} \) from node \( n \) to a nearby node \( k \) is denoted as \( c_t^{nk}(y_t^{nk}) \), but its magnitude is much lower than that of cloud offloading. Regarding the processing amount \( y_t^{nm} \), its latency comes from the computation delay due to its limited computational capability, which is presented as a time-varying function \( h_t^t(y_t^{nm}) \) capturing the dynamic CPU capability during the computing processes. Per slot \( t \), the network delay \( f_t(x_t) \) aggregates the computation delay at all nodes plus the communication delay at all links, namely

\[
f_t(x_t) := \sum_{n \in \mathcal{N}} \left( c_t^n(z^n_t) + \sum_{k \in \mathcal{N}_t^\text{out}} c_t^{nk}(y_t^{nk}) + h_t^t(y_t^{nm}) \right). \]

(3)

Clearly, the explicit form of functions \( c_t^n(\cdot), c_t^{nk}(\cdot), \) and \( h_t^t(\cdot) \) is unknown to the network operator due to the unpredictable traffic patterns [24]; but they are convex (thus \( f_t(x_t) \) is convex) with respect to their arguments, which implies that the marginal computation/communication latency is increasing as the offloading/processing amount grows.

Aiming to minimize the accumulated network delay while serving all the IoT workloads in the long term, the optimal offloading strategy in this mobile network is the solution of the following online optimization problem (cf. (3))

\[
\min_{\{x_t \in \mathcal{X}, \forall t\}} \sum_{t=1}^{T} f_t(x_t), \text{ s. to (2) for } n = 1, \ldots, N. \quad (4)
\]

Comparing to the generic form (1), we consider an online fog computing problem in (4), where the loss (network latency) function \( f_t(\cdot) \) and the data requests \( \{b^n_t\} \) within slot \( t \) are not known when making the offloading and local processing decision \( x_t \); after performing \( x_t \), only the value of \( f_t(x_t) \) (a.k.a. loss) as well as the measurements \( \{b^n_t\} \) are revealed to the network operator. In other words, knowledge of the network operator is fully causal, meaning that before deciding \( x_t \) at time \( t \), the operator knows only \( \{f_t(x_{\tau}), \{b^n_t\}\}_{\tau=1}^{t-1} \). Note that in this example, measuring \( \{b^n_t\} \) is tantamount to knowing the function \( g^n_t(\cdot) \) in (2). Therefore, (4) is in the form of (1).

III. ONLINE BANDIT SADDLE-POINT METHODS

To solve the problem in Section II, an online saddle-point method is revisited first, before developing its bandit variants for network optimization with only partial feedback.
A. Online saddle-point approach with gradient feedback

Several works have studied the OCO setup with time-varying long-term constraints (cf. (1)), including [14], [16], and the recent variant [15] incorporating with adaptive stepsizes. Consider now the per-slot problem (1), which contains the current objective \( f_t(x) \), the current constraint \( g_t(x) \leq 0 \), and a time-invariant feasible set \( \mathcal{X} \). With \( \lambda \in \mathbb{R}_+^n \) denoting the Lagrange multiplier associated with the time-varying constraint, the online Lagrangian of (1) can be expressed as

\[
L_t(x, \lambda) := f_t(x) + \lambda^T g_t(x). \tag{5}
\]

Serving as a basis for developing the bandit approaches, we next revisit the online saddle-point scheme with full-information [16]. Specifically, given the primal iterate \( x_t \) and the dual iterate \( \lambda_t \) at each slot \( t \), the next decision \( x_{t+1} \) is generated by

\[
x_{t+1} = \arg\min_{x \in \mathcal{X}} \nabla_x^T L_t(x_t, \lambda_t)(x - x_t) + \frac{1}{2\alpha} \|x - x_t\|^2 \tag{6}
\]

where \( \alpha \) is a pre-defined constant, and

\[
\nabla_x^T L_t(x_t, \lambda_t) = \nabla f_t(x_t) + \nabla^T g_t(x_t) \lambda_t \]

is the gradient of \( L_t(x, \lambda) \) with respect to (w.r.t.) the primal variable \( x \) at \( x = x_t \). The minimization (6) admits the closed-form solution, given by

\[
x_{t+1} = P_{\mathcal{X}}(x_t - \alpha \nabla_x L_t(x_t, \lambda_t)) \tag{7}
\]

where \( P_{\mathcal{X}}(y) := \arg\min_{x \in \mathcal{X}} \|x - y\|^2 \) denotes the projection operator. In addition, the dual update takes the modified online gradient ascent form

\[
\lambda_{t+1} = [\lambda_t + \mu (g_t(x_t) + \nabla^T g_t(x_t)(x_{t+1} - x_t))]^+ \tag{8}
\]

where \( \mu \) is a positive stepsize, \([\cdot]^+\) represents projection to the positive orthant, and \( \nabla_\lambda L_t(x_t, \lambda_t) = g_t(x_t) \) is the gradient of \( L_t(x_t, \lambda_t) \) w.r.t. \( \lambda = \lambda_t \). Note that (8) is a modified gradient update since the dual variable is updated along the first-order approximation of \( g_t(x_{t+1}) \) at the previous iterate \( x_t \) rather than commonly used \( g_t(x_t) \), which will be critical in our subsequent analytical derivations.

To perform the online saddle-point recursion (7)-(8) however, the gradient \( \nabla f_t(x) \) and the constraint \( g_t(x) \) should be known to the learner at each slot \( t \). When the gradient of \( f_t(x) \) (or its explicit form) is unknown as it is in our setup, additional effort is needed. In this context, the systematic design of the online bandit saddle-point (BanSaP) methods will be leveraged to extend the online saddle-point method to the regime where gradient information is unavailable or computationally costly.

B. BanSaP with one-point partial feedback

The key idea behind BCO is to construct (possibly stochastic) gradient estimates using the limited function value information [17]-[19], [21], [22]. Depending on system variability, the online learner can afford one or multiple loss function evaluations (partial-information feedback) per time slot [18], [21], [22]. Intuitively, the performance of a bandit algorithm will improve if multiple evaluations are available per time slot; see Fig. 2 for a comparison of full- versus partial-information feedback settings.

To begin with, we consider the case where the learner can only observe the function value of \( f_t(x) \) at a single point per slot \( t \). The crux here is to construct a (possibly unbiased) estimate of the gradient using this single piece of feedback. Interestingly though, a stochastic gradient estimate of \( f_t(x) \) can be obtained by one point random function evaluation [17]. The intuition can be readily revealed from the one-dimensional case \((d=1)\): For a binary random variable \( u \) taking values \{-1, 1\} equiprobable, and a small constant \( \delta > 0 \), the idea of forward differentiation implies that the derivative \( f'_t(x) \) at \( x \) can be approximated by

\[
f'_t(x) \approx \frac{f_t(x + \delta) - f_t(x - \delta)}{2\delta} = \mathbb{E}_u \left[ \frac{u}{\delta} f_t(x + \delta u) \right] \tag{9}
\]

where the approximation is due to \( \delta > 0 \), and the equality follows from the definition of expectation. Hence, \( f_t(x + \delta u)/\delta \) can serve as a stochastic estimator of \( f'_t(x) \) based only single function evaluation \( f_t(x + \delta u) \). Generalizing this approximation to high dimensions, with a random vector \( u \) drawn from the unit sphere (a.k.a. the surface of a unit ball), the scaled function evaluation at a perturbed point \( x + \delta u \) yields an estimate of the gradient \( \nabla f_t(x) \), given by [17]

\[
\nabla f_t(x) \approx \mathbb{E}_u \left[ \frac{d}{\delta} f_t(x + \delta u) u \right] := \mathbb{E}_u \left[ \nabla^1 f_t(x) \right] \tag{10}
\]

where we define one-point gradient \( \nabla^1 f_t(x) := \frac{d}{\delta} f_t(x + \delta u) u \).

Building upon this intuition, consider a bandit version of the online saddle-point iteration, for which the primal update becomes (cf. (7))

\[
x_{t+1} = P_{\{1-\gamma\} \mathcal{X}} \left( x_t - \alpha \nabla^1 L_t(x_t, \lambda_t) \right) \tag{11}
\]

where \((1-\gamma)\mathcal{X} := \{ (1-\gamma) x : x \in \mathcal{X} \}\) is a subset of \( \mathcal{X} \), \( \gamma \in [0,1) \) is a pre-selected constant depending on \( \delta \), and the one-point Langrangian gradient is given by (cf. (10))

\[
\nabla^1 L_t(x_t, \lambda_t) := \nabla^1 f_t(x_t) + \nabla^T g_t(x_t) \lambda_t. \tag{12}
\]

In the full-information case, \( x_t \) in (7) is the learner’s action, but in the bandit case the learner’s action is \( x_{t,t} := x_t + \delta u_t \), which is the point for function evaluation but not \( x_t \) in (11). Note that while the random perturbation \( u_t \) is assumed to lie on the surface of a unit ball, we do not confine the actual IoT decision \( x_t \) to follow any specific distribution.

Furthermore, the projection is performed on a smaller convex
Algorithm 1 BanSaP for OCO with time-varying constraints

1: **Initialize:** primal iterate \( \bar{x}_1 \), dual iterate \( \lambda_1 \), parameters \( \delta \) and \( \gamma \), and proper step sizes \( \alpha \) and \( \mu \).

2: **for** \( t = 1, 2, \ldots \) **do**

3: The learner plays the perturbed actions \( \{x_{m,t}\}_{m=1}^{M} \) based on the learning iterate \( \bar{x}_t \).

4: The nature reveals the losses \( \{f_i(x_{m,t})\}_{m=1}^{M} \) at queried points, and the constraint function \( g_i(x) \).

5: The learner updates the primal variable \( \bar{x}_{t+1} \) by (11) with the gradient estimated by (12) for \( M = 1 \), or (15) for \( M = 2 \), otherwise, by (17).

6: The learner updates the dual variable \( \lambda_{t+1} \) via (13).

7: **end for**

---

The learner plays the perturbed actions \( \{x_{m,t}\}_{m=1}^{M} \) based on the learning iterate \( \bar{x}_t \).

with one-point bandit feedback do not increase computation than the actual decision \( x_t \) is used in this update. Compared with the gradient-based recursions \( (7)-(8) \), the updates \( (11)-(13) \) with one-point bandit feedback do not increase computation or memory requirements, and thus provide a light-weight surrogate for gradient-free online bandit network optimization.

C. BanSaP with multipoint partial feedback

Featuring a simple update given minimal information, the BanSaP with one-point bandit feedback is suitable for fast-varying environments, where multiple function evaluations are impossible. As shown later in Sections IV and V, the theoretical and empirical performance of BanSaP with single-point evaluation is degraded relative to the full-information case.

To improve the performance of BanSaP with one-point feedback, we will first rely on two-point function evaluation at each slot [21], and then generalize to multipoint evaluation. Intuitively, this approach is justified when the underlying dynamics are slow, e.g., when the load and price profiles in power grids are piece-wise stationary. In this case, each slot can be further divided into multiple mini-slots, and one query is performed per mini-slot, over which the loss function and the constraints do not change. Compared to \( (11)-(13) \), the key difference is that the one-point estimate in \( (12) \) is replaced by

\[
\hat{\nabla}^2 f_i(\bar{x}_t) := \frac{d}{2\delta} (f_i(\bar{x}_t + \delta u_t) - f_i(\bar{x}_t - \delta u_t)) u_t
\]

(d = 1), where the expectation of the differentiation term in \( (14) \) approximates well the derivative of \( f_t \) at \( \bar{x}_t \); that is,

\[
E_u \left[ \frac{d}{2\delta} (f_t(\bar{x}_t + \delta u_t) - f_t(\bar{x}_t - \delta u_t)) \right] = \frac{1}{2\delta} (f_t(\bar{x}_t + \delta) - f_t(\bar{x}_t - \delta)) \approx f'_t(\bar{x}_t)
\]

where the equality follows because the random variable \( u_t \) takes values \( \{-1, 1\} \) equiprobable.

Relative to the one-point feedback case, the advantage of the two-point feedback is variance reduction in the gradient estimator. Specifically, the second moment of the stochastic gradient can be uniformly bounded, \( E[\frac{d}{2\delta} (f_t(\bar{x}_t + \delta u_t) - f_t(\bar{x}_t - \delta u_t)) u_t^2] \leq d^2 F^2/\delta^2 \), with \( F \) denoting an upper-bound of \( f_t(x) \). The proof of this argument can be found in the Appendix (Lemma ??). In fact, a bias-variance tradeoff emerges in the one-point case, but not in the two-point case. This subtle yet critical difference will be responsible for an improved performance of BanSaP with two-point feedback, and its stable empirical performance, as will be seen later.

With the insights gained so far, the next step is to endow the BanSaP with more than two function evaluations [18]. With \( M > 2 \) points, the gradient estimator is obtained by querying the function values over \( M \) points in the neighborhood of \( \bar{x}_t \). These points include \( x_{m,t} := \bar{x}_t + \delta u_{m,t}, 1 \leq m \leq M - 1 \), and the learning iterate \( x_{m,t} := \bar{x}_t \), where \( u_{m,t} \) is independently drawn from \( \mathbb{S} \). Specifically, the gradient becomes (cf. \( (11) \))

\[
\hat{\nabla}_x^M L_t(\bar{x}_t, \lambda_t) := \frac{d}{\delta(M - 1)} \sum_{m=1}^{M-1} (f_t(\bar{x}_t + \delta u_{m,t}) - f_t(\bar{x}_t)) u_{m,t} + \nabla^\top g_t(\bar{x}_t) \lambda_t
\]

where we define the \( M \)-point stochastic gradient as \( \hat{\nabla}^M f_t(\bar{x}_t) := \frac{d}{\delta(M - 1)} \sum_{m=1}^{M-1} (f_t(\bar{x}_t + \delta u_{m,t}) - f_t(\bar{x}_t)) u_{m,t} \).

At the price of extra computations, simulations will validate that the BanSaP with multipoint feedback enjoys improved performance. The family of the BanSaP approaches with one-or multiple-point feedback is summarized in Algorithm 1.

**Remark 1** (Sampling schemes). The BanSaP solvers here adopt uniform sampling for gradient estimation, meaning \( u \) is drawn uniformly from the unit sphere. However, other sampling rules can be incorporated without affecting the order of regret bounds derived later. For example, one can sample \( u \) from the canonical basis of a \( d \)-dimensional space uniformly at random [18], or, sample \( u \) from a normal distribution [22]. The effectiveness of these schemes will be tested using simulations.

IV. PERFORMANCE ANALYSIS

In this section, we will introduce pertinent metrics to evaluate BanSaP algorithms in the online bandit learning with long-term constraints, and rigorously analyze the performance of the proposed algorithms.

A. Optimality and feasibility metrics

With regard to performance of BCO schemes, static regret is a common metric, under time-invariant and strictly satisfied
constraints, which measures the difference between the aggregate loss and that of the best fixed solution in hindsight [17], [18]. Extending the definition of static regret to accommodate \( M \)-point function evaluations and time-varying constraints, let us first consider

\[
\text{Reg}_T^d := \frac{1}{M} \sum_{t=1}^{T} \sum_{m=1}^{M} \mathbb{E}[f_t(x_{m,t})] - \sum_{t=1}^{T} f_t(x^*)
\]  

(18)

where the actual loss per slot is averaged over the losses of \( M \) actions (queried points), \( \mathbb{E} \) is taken over the sequence of the random actions \( x_{m,t} \) with randomness induced by \( \{u_{m,t}\} \) perturbations, and the best static solution is \( x^* \in \arg\min_{x \in X} \sum_{t=1}^{T} f_t(x) \); s. to \( g_t(x) \leq 0 \), \( \forall t \). A BCO algorithm yielding a sub-linear regret implies that the algorithm is “on average” no-regret [23]; or, in other words, asymptotically not worse than the best fixed solution \( x^* \). Though widely used, the static regret relies on a rather coarse benchmark, which is not as useful in dynamic IoT settings. Specifically, the gap between the loss of the best static and that of the best dynamic benchmark is as large as \( O(T) \) [31]. In response to the quest for improved benchmarks in this dynamic setup with constraints, two metrics are considered here: **dynamic regret** and **dynamic fit**. The notion of dynamic regret has been recently adopted in [12], [13] to assess performance of online algorithms under time-invariant constraints. For our BCO setting of (1), we adopt

\[
\text{Reg}_T^d := \frac{1}{M} \sum_{t=1}^{T} \sum_{m=1}^{M} \mathbb{E}[f_t(x_{m,t})] - \sum_{t=1}^{T} f_t(x^*_t)
\]  

(19)

where \( \mathbb{E} \) is again taken over the sequence of random actions, and the benchmark is now formed via a sequence of best dynamic solutions \( \{x^*_t\} \) for the instantaneous cost minimization problem subject to the instantaneous constraint, namely

\[
x^*_t \in \arg\min_{x \in X} f_t(x) \quad \text{s. to} \quad g_t(x) \leq 0.
\]  

(20)

Comparing (19) with (18), if \( x^*_t = x^*, \forall t \), then the static regret is equivalent to the dynamic regret. In general, the dynamic regret is larger than the static regret, i.e., \( \text{Reg}_T^d \leq \text{Reg}_T^s \), since \( \sum_{t=1}^{T} f_t(x^*) \) is always no smaller than \( \sum_{t=1}^{T} f_t(x^*_t) \) according to the definitions of \( x^* \) and \( x^*_t \). Hence, a sub-linear dynamic regret implies a sub-linear static regret, but not vice versa.

Regarding feasibility of decisions generated by a BCO algorithm, the notion of **dynamic fit** will be used to measure the accumulated violation of constraints [23], that is

\[
\text{Fit}_T^d := \left\| \frac{1}{M} \sum_{t=1}^{T} \sum_{m=1}^{M} g_t(x_{m,t}) \right\|.
\]  

(21)

Note that the dynamic fit is zero if the accumulated violation \( \frac{1}{M} \sum_{t=1}^{T} \sum_{m=1}^{M} g_t(x_{m,t}) \) is entry-wise less than zero. Hence, enforcing \( \frac{1}{M} \sum_{t=1}^{T} \sum_{m=1}^{M} g_t(x_{m,t}) \leq 0 \) is different from restricting \( x^*_t \) to meet \( \frac{1}{M} \sum_{m=1}^{M} g_t(x_{m,t}) \leq 0 \) in every slot. While the latter implies the former, the long-term constraint implicitly assumes that the instantaneous constraint violations can be compensated by the later strictly feasible decisions.

Under this broader BCO setup, an ideal online algorithm is the one that achieves both sub-linear dynamic regret and sub-linear dynamic fit. A sub-linear dynamic regret implies “no-regret” relative to the clairvoyant dynamic solution on the long-term average; i.e., \( \lim_{T \to \infty} \text{Reg}_T^d / T = 0 \); and a sub-linear dynamic fit indicates that the online strategy is also feasible on average; i.e., \( \lim_{T \to \infty} \text{Fit}_T^d / T = 0 \). Unfortunately, the sub-linear dynamic regret is not achievable under arbitrary underlying dynamics, even when the time-varying constraint in (1) is absent [31]. Therefore, we aim at designing an online strategy that generates a sequence \( \{x_{m,t}\} \) ensuring sub-linear dynamic regret and fit, under the suitable **regularity conditions** on the underlying dynamics.

### B. Main results

Before formally analyzing the dynamic regret and fit for BanSaP, we assume that the following conditions are satisfied.

\textbf{(as1)} For every \( t \), the functions \( f_t(x) \) and \( g_t(x) \) are convex.

\textbf{(as2)} Function \( f_t(x) \) is bounded over the set \( X \), meaning \( |f_t(x)| \leq F, \forall x \in X \); while \( f_t(x) \) and \( g_t(x) \) have bounded gradients; that is, \( \|\nabla f_t(x)\| \leq G \), and \( \max_{x \in X} \|\nabla g_t(x)\| \leq G \).

\textbf{(as3)} For a small constant \( \gamma \), there exists a constant \( \eta > 0 \), and an interior point \( \hat{x} \in (1-\gamma)X \) such that \( g_t(\hat{x}) \leq -\eta 1, \forall t \).

\textbf{(as4)} With \( \mathbb{B} := \{x \in \mathbb{R}^d : \|x\| \leq 1\} \) denoting the unit ball, there exist constants \( 0 < r < R \) such that \( r \mathbb{B} \subseteq X \subseteq R \mathbb{B} \).

Assumptions (as1)-(as2) are typical in OCO with both full- and partial-information feedback [10], [17], [23]; (as3) is Slater’s condition modified for our BCO setting, which guarantees the existence of a bounded Lagrange multiplier [32] in the constrained optimization; and, (as4) requires the action set to be bounded within a ball that contains the origin. When (as4) appears to be restrictive, it is tantamount to assuming \( X \) is compact and has a nonempty interior, because one can always apply an affine transformation (a.k.a. reshaping) on \( X \) to satisfy (as4); see [17, Section 3.2].

Under these assumptions, we are on track to first provide upper bounds for the dynamic regret, and the dynamic fit of the BanSaP solver with one-point feedback.

**Theorem 1** (one-point feedback). Suppose that (as1)-(as4) are satisfied, and consider the parameters \( \alpha, \mu, \delta, \gamma \) defined in (11)-(13), and constants \( F, G, r, R \) defined in (as2)-(as4). If the dual variable is initialized by \( \lambda_1 = 0 \), then the BanSaP with one-point feedback in (7)-(8) has dynamic regret bounded by

\[
\text{Reg}_T^d \leq \frac{R}{\alpha} + \frac{R^2}{2\alpha} + \frac{d^2G^2R^2T}{\delta^2} + 2G\delta T + \gamma \mu G T (1 + \|\lambda\|) + 2\mu G^2 R^2 T
\]  

(22)

where \( \|\lambda\| := \max_{t} \|\lambda_t\| \), and the accumulated variation of the per-slot minimizers \( x^*_t \) in (20) is given by

\[
V(x^{1:T}) := \sum_{t=1}^{T} \|x^*_t - x^*_{t-1}\|.
\]  

(23)

In addition, the dynamic fit defined in (21) is bounded by

\[
\text{Fit}_T^d \leq \frac{\|\lambda\|}{\mu} + \frac{G^2 \sqrt{NT}}{2\beta} + \frac{\delta G \sqrt{NT}}{\beta} \left( \frac{\alpha^2 d^2 F^2}{\delta^2} + \alpha^2 G^2 \|\lambda\|^2 \right)
\]  

(24)

where \( \beta > 0 \) is a pre-selected constant. Furthermore, if we choose the stepsizes as \( \alpha = \mu = O(T^{-\frac{5}{7}}) \), and the parameters
\[ \delta = \Theta(T^{-\frac{1}{2}}), \beta = T^{\frac{1}{2}} \text{ and } \gamma = \delta/r, \text{ then the online decisions generated by BanSaP are feasible, i.e., } \mathbf{x}_{t^*}, \mathbf{t} \in \mathcal{X} ; \text{ and also yield the following dynamic regret and fit} \]

\[ \text{Reg}_{dT} = \Theta\left( V(\mathbf{x}_{1:T}^*) T^{\frac{1}{2}} \right) \text{ and } \text{Fit}_{dT} = \Theta\left( T^{\frac{1}{2}} \right). \tag{25} \]

**Proof:** See the online version [33].

For BanSaP with one-point feedback, Theorem 1 asserts that its dynamic regret and fit are upper-bounded by some constants depending on the those parameters, the time horizon, and the accumulated variation of per-slot minimizers. Interestingly, the crucial constant \( \delta \) controlling the perturbation of random actions appears in both the denominator and numerator of (22) and (24), which correspond to the variance and bias of the gradient estimator. Therefore, simply setting a small \( \delta \) will not only reduce the bias, but it will also boost the variance - a clear manifestation of the that is known as bias-variance tradeoff in BCO [19]. Optimally choosing parameters implies that the dynamic fit is sub-linearly growing, and the dynamic regret is sub-linearly growing, and the dynamic regret is bounded by

\[ \text{Reg}_{dT} = \Theta\left( T^{\frac{1}{2}}(\rho)^{3} \right) \text{ and } \text{Fit}_{dT} = \Theta\left( T^{\frac{1}{2}}(\rho)^{3} \right). \tag{29} \]

Likewise, if the stepsizes of BanSaP with two-point feedback are chosen such that \( \alpha = \Theta(4^{\rho} - 1), \) and the parameters are \( \delta = \Theta(T^{\frac{1}{2}}(\rho)^{2}) \), \( \beta = T^{\frac{1}{2}}(\rho)^{2} \), and \( \gamma = \delta/r, \) then the dynamic regret and fit in (25) become

\[ \text{Reg}_{dT} = \Theta\left( T^{\frac{1}{2}}(\rho)^{3} \right) \text{ and } \text{Fit}_{dT} = \Theta\left( T^{\frac{1}{2}}(\rho)^{3} \right). \tag{30} \]

**Remark 2** (Optimal regret). As a special case of Theorems 1 and 2, by confining \( \mathbf{x}_t = \cdots = \mathbf{x}_{t^*} \), so that \( V(\mathbf{x}_{1:T}^*) = 0 \), the dynamic regret bounds (25) and (28) reduce to the static ones, which correspond to \( \Theta(T^2) \) in the one-point feedback case, and to \( \Theta(\sqrt{T}) \) in the two-point case. This pair of bounds markedly improves the regret versus fit tradeoff in [23], and matches the order of regret in [17], and [18], [21], which are the best possible ones that can be achieved by efficient algorithms even in the BCO setup without the long-term constraints. Considering the full-information setting in [15] as a special case, the regret and fit of BanSaP outperform those in [15].

**Remark 3** (Dynamic regret). Theorems 1, 2 and Corollary 1 extend the dynamic regret analysis in [12]–[14] to the regime of bandit online learning with long-term time-varying constraints.

Interestingly though, in the BCO setting of our interest, sub-linear dynamic regret and fit are possible to achieve when the per-slot minimizer does not vary on average, that is, \( V(\mathbf{x}_{1:T}^*) \) is sub-linearly growing with \( T \).

**V. NUMERICAL TESTS**

In this section, we demonstrate how the fog computation offloading task can benefit from our novel BanSaP solvers.

**A. BanSaP for fog computation offloading**

Recall that the computation offloading problem (4) is in the form of (1). Therefore, the BanSaP solver of Section III can
Algorithm 2 BanSaP for fog computation offloading

1: **Initialize:** primal iterates \{\hat{y}^n_{nk}\} and \{\hat{z}^n_t\}, dual iterate \(\lambda_t\), parameters \(\delta\) and \(\gamma_t\), and proper step sizes \(\alpha\) and \(\mu\).
2: for \(t = 1, 2, \ldots\) do
3: \hspace{1em} for \(n = 1, \ldots, N\) do
4: \hspace{2em} Fog nodes perform perturbed offloading decisions to cloud \{\hat{z}^n_t\}, to neighbor edges \{\hat{y}^n_{nk}\}, and locally process \{\hat{y}^n_{nk}\} based on \(x_t\).
5: \hspace{1em} end for
6: Fog nodes observe the (possibly multiple) losses to update (31) with stochastic gradients obtained via (32).
7: Fog nodes observe the actual user demands from IoT devices to update the dual variables (33).
8: end for

be customized to solve (4) in an *online* fashion, with provable performance and feasibility guarantees.

Specifically, with \(g_t(x_t)\) as in (2) and \(f_t(x_t)\) as in (3), the primal update (7) boils down to a simple closed-form gradient update amenable to decentralized implementation; the cloud offloading amount at node \(n\) is

\[
\hat{z}^n_{t+1} = \left[ \hat{z}^n_t - \alpha \left( \nabla c^n_t(\hat{z}^n_t) - \lambda^n_t \right) \right]_0
\]

and the offloading amount from node \(n\) to node \(k\) is given by

\[
\hat{y}^n_{tk} = \left[ \hat{y}^n_{tk} - \alpha \left( \nabla c^n_t(\hat{y}^n_{tk}) - \lambda^n_t + \lambda^k_t \right) \right]_0
\]

while the local processing decision at node \(n\) is generated by

\[
\hat{y}^n_{tn} = \left[ \hat{y}^n_{tn} - \alpha \left( \nabla h^n_t(\hat{y}^n_{tn}) - \lambda^n_t \right) \right]_0
\]

where \(\alpha\) is chosen according to Theorems 1 and 2. Using two-point feedback (\(M = 2\)) as an example, the gradients involved in (31) can be estimated as

\[
\nabla^2 c^n_t(\hat{z}^n_t) := \frac{d}{2\delta} (f_t(\hat{x}_{1:t} + \delta u_t) - f_t(\hat{x}_{1:t} - \delta u_t))_{u_t(\hat{z}^n_t)}
\]

and with respect to the offloading variable, as

\[
\nabla^2 c^n_t(\hat{y}^n_{tk}) := \frac{d}{2\delta} (f_t(\hat{x}_{1:t} + \delta u_t) - f_t(\hat{x}_{t-1:t} - \delta u_t))_{u_t(\hat{y}^n_{tk})}
\]

and with respect to the local processing variable, as

\[
\nabla^2 h^n_t(\hat{y}^n_{tn}) := \frac{d}{2\delta} (f_t(\hat{x}_{1:t} + \delta u_t) - f_t(\hat{x}_{1:t} - \delta u_t))_{u_t(\hat{y}^n_{tn})}
\]

where \(u_t(\hat{z}^n_t), u_t(\hat{y}^n_{tk}),\) and \(u_t(\hat{y}^n_{tn})\) represent the corresponding entries of the random vector \(u_t \in \mathbb{R}^{|E|}\) at slot \(t\).

The dual update (8) at each node \(n\) reduces to

\[
\lambda^n_{t+1} = \lambda^n_t + \mu \left( b^n_{t+1} + \sum_{k \in \mathbb{N}^n_{tn}} y^n_{tk} - \sum_{k \in \mathbb{N}^n_{tn}} y^n_{tk} - \hat{z}^n_{t+1} - \hat{y}^n_{tn} \right)
\]

where \(\mu\) is chosen according to Theorems 1 and 2. Intuitively, to guarantee completion of the service requests, the dual variable increases (increasing penalty) when there is instantaneous service residual, and decreases when over-serving incurs in the mobile-edge computing systems. Following its generic form in Algorithm 1, BanSaP for online fog computation offloading tasks, is summarized in Algorithm 2.
is convex and its point-wise value can be evaluated. Therefore, the online cost (a.k.a. aggregate service latency) in (3) is specified by

$$f_t(x_t) := \sum_{n \in \mathcal{N}} \left( e^{p_t^ny_t^n} + \sum_{k \in \mathcal{N}^\text{out}} t^nk y_t^k + t^nn (y_t^n)^2 \right)$$

where $p_t^n = 0.015 \sin(\pi t/96) + 0.05$, $n \in \mathcal{N}\setminus\{4,5\}$, $p_t^3 = 0.045 \sin(\pi t/96) + 0.15$, $n \in \{4,5\}$, and the local coefficients are set to $t^nk = 8/n^k$ and $t^nn = 8/n^n$. Regarding the data arrival rate $b_t^n$, it is generated according to $b_t^n=q^n \sin(\pi t/96)+\nu_t^n$, with $q^n$ and $\nu_t^n$ uniformly distributed over $[40,50]$ and $[45,55]$ for $n \in \mathcal{N}\setminus\{1,2,3\} \cup \{4,5\}$, and $q^3 \in [32,40]$, $\nu_t^3 \in [36,44]$, $n \in \{1,2,3\}$, and $q^\in \{20,25\}$, $\nu_t^\in \{22.5,27.5\}$, $n \in \{4,5\}$. Notice that the periods of $p_t^n$ and $b_t^n$ vary between nodes, mimicking heterogeneity of IoT sensors such as motion sensors and thermostat sensors [37]. It is also worth mentioning that our BanSaP algorithm as well as its performance analysis are model-free, which means they can incorporate more complex models so long as the loss function is convex and its point-wise value can be evaluated.

Finally, BanSaP is benchmarked by: i) the full-information modified online saddle-point method (MOSP) in [14] that takes gradient-based update for primal-dual variables; ii) the heuristic cloud-only approach that offloads all data requests to the remote cloud; iii) the heuristic fog-only approach that processes all data requests locally without collaboration; and, iv) the partial-information perturbed online primal-dual method in [23]. For both cloud-only and fog-only approaches, unoffloaded and unprocessed requests are buffered at the fog nodes for later processing; thus, these amounts are measured by their fit. Regarding the perturbed primal-dual method in [23], it comes with two-point bandit feedback, and the perturbation constant is chosen as $0.06$ to satisfy the technical conditions therein.

As different stepsizes of BanSaP and MOSP lead to different behaviors, we manually optimized stepsizes in each test so that they have similar fit, and focus on their cost comparison. When the parameters of BanSaP need to be slightly adjusted in each setting with $N=5$ nodes, the fit and average cost are compared among the BanSaP variants with $M$-point feedback under different sampling schemes in Figs. 3 and 4. Clearly, for both sampling schemes, the cost and fit of BanSaP solvers decrease as the amount of bandit feedback increases. However, such performance gain vanishes when feedback increases; e.g., $M \geq 4$. Regarding the sampling schemes, Fig. 3 demonstrates...
that when all the BanSaP variants have low dynamic fit, the uniform sampling-based BanSaP with one-point feedback has large initial fit; and Fig. 4 confirms that for $M = 1$, the coordinate sampling-based BanSaP outperforms that with uniform sampling; and, for $M \geq 2$, the BanSaP solvers with uniform sampling incur lower cost. Therefore, to optimize empirical performance in the subsequent tests, coordinate sampling is adopted by BanSaP with $M = 1$, while uniform sampling is used in BanSaP with $M \geq 2$.

Optimality and feasibility. With optimized sampling schemes for BanSaP solvers, the dynamic fit and average cost are then compared among three BanSaP variants, MOSP, the perturbed primal-dual method in [23], and two heuristic schemes in Figs. 5 and 6. Without queueing at the fog side, the cloud-only scheme has much lower dynamic fit since all user demands are offloaded to the remote cloud. However, it incurs a much higher average cost (service latency) as the network latency between fog and cloud becomes high due to the large offloading amount. By increasing the amount of feedback, the BanSaP solver tends to have a lower fit and a lower average cost, both of which are comparable to those of MOSP when $M \geq 2$. On the other hand, the BanSaP with only one-point bandit feedback still has a similar fit relative to the fog-only scheme, but enjoys much lower cost. Interestingly enough, when the variance (cf. the shaded area in Fig. 6) of the one-point BanSaP's cost is high, it markedly vanishes when multiple function values become available, which corroborates our claims in Theorems 1-2. For the perturbed method in [23], while its average cost is similar or slightly better than that of BanSaP, its dynamic fit is much higher than all BanSaP variants, which is aligned with its $O(T^{2})$ fit (cf. $O(T^{2})$ in our corresponding case).

Effect of network size. The third test evaluates the performance of all schemes under different number of fog nodes (i.e., network size). For each algorithm, the fit averaged over all fog nodes and time is presented in Fig. 7, and the cost averaged over the time is shown in Fig. 8. Clearly, the one-point BanSaP has lower average fit than the fog-only approach in most scenarios, and also incurs less average cost in all tested settings. Similar to those in Figs. 5 and 6, the average fit and cost of BanSaP with multiple function evaluations is relatively comparable to that of the full-information MOSP as the network size grows. On the other hand, the method in [23] enjoys slightly lower average cost as the network size grows, but its dynamic fit is again much higher than all BanSaP variants. An interesting observation here is that as the number of fog nodes increases, the performance gain of the BanSaP solver with a large $M$ becomes more evident; see e.g., Fig. 8. This implies that for a larger network, BanSaP benefits from more bandit information to learn and track the network dynamics.

VI. CONCLUSIONS AND THE ROAD AHEAD

Bandit convex optimization (BCO) in dynamic environments was studied in this paper. Different from existing works in bandit settings, the focus was on a broader setting where part of the constraints are revealed after taking actions, and are also tolerable to instantaneous violations but have to be satisfied on average. The novel BCO setting fits well the emerging fog computing tasks in IoT. A class of online bandit saddle-point (BanSaP) approaches were proposed, and their online performance was rigorously analyzed. It was shown that the resultant regret bounds match those attained in BCO setups without long-term constraints. Furthermore, the BanSaP solvers can simultaneously yield sub-linear dynamic regret and fit, if the dynamic solutions vary slowly over time.

Our algorithmic and theoretical results serve as an exciting first step to innovate online bandit learning tailored for dynamic network management tasks, emerging from contemporary IoT applications. Interesting future directions include designing asynchronous variants of BanSaP, and incorporating predictable models in online optimization. To fully assess its practical impact, extensive experiment validations on real testbeds are also of great importance.

REFERENCES

Tianyi Chen (S’14) received the B. Eng. degree (with highest honors) in Communication Science and Engineering from Fudan University, and the M.Sc. degree in Electrical and Computer Engineering (ECE) from the University of Minnesota (UMN), in 2014 and 2016, respectively. Since July 2016, he has been working toward his Ph.D. degree at UMN. His research interests lie in online convex optimization, reinforcement learning and stochastic network optimization with applications to Internet-of-Things (IoT). He was in the Best Student Paper Award finalista of the Asilomar Conference on Signals, Systems, and Computers. He received Best Demo Award in the International Conference on IoT, and the National Scholarship from China in 2013, UMN ECE Department Fellowship in 2014, and the UMN Doctoral Dissertation Fellowship in 2017.

Georgios B. Giannakis (F’97) received his Diploma in Electrical Engineering from the NTL. Techn. Univ. of Athens, Greece, 1981. From 1982 to 1986 he was with the Univ. of Southern California (USC), where he received his MSc. in Electrical Engineering, 1983, MSc. in Mathematics, 1986, and Ph.D. in Electrical Engr., 1986. He was with the University of Virginia from 1987 to 1998, and since 1999 he has been a professor with the Univ. of Minnesota, where he holds an Endowed Chair in Wireless Telecommunications, a University of Minnesota McKnight Presidential Chair in ECE, and serves as director of the Digital Technology Center.

His general interests span the areas of communications, networking and statistical signal processing - subjects on which he has published more than 400 journal papers, 700 conference papers, 25 books chapters, two edited books and two research monographs (h-index 131). Current research focuses on learning from big data, wireless cognitive radios, and network science with applications to social, brain, and power networks with renewables. He is the (co-) inventor of 30 patents issued, and the (co-) recipient of 8 best paper awards from the IEEE Signal Processing (SP) and Communications Societies, including the G. Marconi Prize Paper Award in Wireless Communications. He also received Technical Achievement Awards from the SP Society (2000), from EURASIP (2005), a Young Faculty Teaching Award, the G. W. Taylor Award for Distinguished Research from the University of Minnesota, and the IEEE Fourier Technical Field Award (2015). He is a Fellow of EURASIP, and has served the IEEE in a number of posts, including that of a Distinguished Lecturer for the IEEE-SP Society.