Distributed no-regret edge resource allocation with limited communication

Saad Kriouile, Dimitrios Tsilimantos, and Theodoros Giannakas
Huawei Paris Research Centre, France, first.last@huawei.com

***Abstract*—To accommodate low latency and computation-intensive services, such as the Internet-of-Things (IoT), 5G networks are expected to have cloud and edge computing capabilities. To this end, we consider a generic network setup where devices, performing analytics-related tasks, can partially process a task and offload its remainder to base stations, which can then reroute it to cloud and/or to edge servers. To account for the potentially unpredictable traffic demands and edge network dynamics, we formulate the resource allocation as an online convex optimization problem with service violation constraints and allow limited communication between neighboring nodes. To address the problem, we propose an online distributed (across the nodes) primal-dual algorithm and prove that it achieves sublinear regret and violation; in fact, the achieved bound is of the same order as the best known centralized alternative. Our results are further supported using the publicly available Milano dataset.**

***Index Terms*—Online convex optimization, edge computing, resource allocation, distributed algorithms.**

I. Introduction

A. Motivation

It is envisioned that globally more than 29.3 billion networked devices will be connected to the Internet of Things (IoT) by 2023 [1], offering automation and real-time monitoring of machine and human-driven processes. A main challenge in IoT deployment lies with the massive amount of connected devices; and in particular with the device heterogeneity (e.g., different computational capabilities) and the diverse and potentially stringent service (task) requirements [2]. To host the unprecedented IoT data traffic, the edge computing paradigm has recently gained a lot of momentum as complementary to that of the cloud and it has been deemed as a key enabler to what ultimately will be the so-called "Cloud-to-Things Continuum" [3], [4].

In that framework, a plethora of spatially distributed devices collect data from sensors and perform low-latency impromptu computation (e.g., machine-learning inference) using energy-limited resources. The envisaged "Cloud-to-Things Continuum" allows flexible task offloading *from* an IoT device, via base stations, *towards* more computationally powerful edge servers and, if needed, to the cloud. Although this architecture is promising, allocating resources for IoT computations has two distinct fundamental challenges: (a) the resources are allocated in the presence of highly unpredictable and non-stationary request patterns (demands) and network conditions; (b) the network nodes handling those tasks, namely devices, base stations and servers, are distributed and should act in the absence of a centralized entity with full observability. Naturally, the following question arises:

"Can we offer an efficient distributed data-driven algorithm for resource allocation in the IoT context?"

In order to address this question, in this paper we consider a distributed setting with nodes of different capabilities and employ Online Convex Optimization (OCO) [5]. The use of OCO is suitable for problems that are difficult to model due to unpredictable dynamics and provides concrete theoretical guarantees for such problems even in the worst-case.

B. Related Work

The related literature can be split into two categories. The first corresponds to studies of *IoT network optimization* in the edge-cloud setting, using similar system model and assumptions to ours. An offline version of the problem is formulated and then decomposed across its different domains (fog and cloud) in [6], resulting in convex subproblems. In [7], the authors consider the latency minimization and develop an algorithm based on the online secretary framework—however, no constraints are used in their formulation. Closer to our work, [2], [4] formulate the resource allocation as an OCO problem, and model the service violations as long-term constraints. The learning rate is adapted to the different IoT tasks in [2]; and further extending the notion of constraint violation (see [8]), the number of violations is also considered in [4]. Unlike our work, both approaches are centralized.

The second one deals with works on *OCO with constraints in generic settings*, such as [9]–[11]. Although generic, these works do not apply to our problem as they develop centralized algorithms. Delayed feedback on the cost and constraint functions arrive to a centralized agent in [12]; our approach adopts a different feedback model based on limited exchange of information between nodes.

This work was supported by the CHIST-ERA LeadingEdge project, grant CHIST-ERA-18-SDCDN-004 through ANR grant number ANR-19-CHR3-0007-06. The three authors contributed equally; the order is random.

Finally, a distributed OCO algorithm in an environment with time-varying constraints is presented and analyzed in [13], but, unlike our approach, the nodes/actors use synchronous information and make consensus steps.

C. Contributions and Structure

In this work, we approach the resource allocation in an edge-cloud scenario in a distributed way; our main contributions can be summarized as follows.

(C.1) We model the resource allocation as a distributed constrained OCO problem. The network nodes (devices, base stations, edge servers) are cast as individual OCO agents that must collectively optimize a given network performance metric and satisfy service requirement guarantees (modeled as long-term constraints). To this end, we define a limited communication model between neighboring agents that allows them to exchange crucial information, related to their coupling constraints.

(C.2) We propose an online primal-dual algorithm, based on projected gradient descent, with a sub-linear regret bound $\mathcal{O}(T^{1/2})$ and a sub-linear constraint violation bound $\mathcal{O}(T^{3/4})$. These bounds are equivalent to a centralized approach besides a multiplicative factor. We validate our theoretical results with numerical simulations on a commonly used dataset, and compare the performance of our algorithm to benchmarks.

The remainder of this paper is organized as follows. We summarize our system model and assumptions in Section II. Then, we introduce the OCO formulation of the problem and present our proposed algorithm and its theoretical guarantees in Section III. We show our numerical results in Section IV and conclude the paper in Section V.

II. Problem Setup

In this section we present our network model and main assumptions. In particular, we start by the edge computing components that are available in our IoT application; then we present the control (optimization) variables, and finally we discuss our performance objectives and system constraints.

In what follows, we use bold fonts for vectors and matrices, and $\mathcal{A}$ for a set with $|\mathcal{A}| = A$ elements.

A. Topology and Computational Requests

We consider a layer of IoT sensors, which receive computational requests (e.g., analytics tasks) that need to be executed, similarly to [2], [4], [6], [7], [14], [15]. Time is slotted and at every timeslot t those requests arrive to a set of devices $\mathcal{D}$. We denote the vector of requests as $\mathbf{r}^t \in \mathbb{R}_+^D$. Before $\mathbf{r}^t$ is revealed, the Network Operator (NO) has to reserve resources across its network infrastructure in order to accommodate them.

We assume that the network consists of the following nodes, as shown in Fig. 1.

- Devices at the edge, denoted by $\mathcal{D}$;

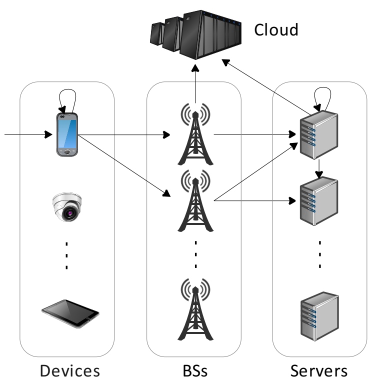


Fig. 1: Topology of our edge computing setting

- Base Stations (BSs) at the edge, denoted by $\mathcal{B}$;
- Servers at the edge, denoted by $\mathcal{S}$;
- A cloud server at the core network, denoted by C.

Throughout the rest of this work, we will refer to the above entities (except for the cloud) as “nodes” or “agents”. We denote by $\mathcal{N}$ the set of all nodes across the network with $N = D + B + S$.

Each device can process locally part of the computation and can also offload tasks via a wireless channel to the BSs. Then, the BSs can forward an incoming task either to the edge servers (wirelessly) or to the cloud. Finally, each edge server can process the received tasks locally, reroute to other edge servers or forward to the cloud; the latter only executes tasks. For simplicity, we assume full connectivity between nodes, but our methodology applies to any connectivity graph.

B. Distributed Control Variables

The NO wishes to optimize a set of performance metrics in a *distributed* manner. Therefore, we wish to design a system where each agent decides its own actions. At every t, the control variables for every device $d \in \mathcal{D}$, BS $b \in \mathcal{B}$ and server $s \in \mathcal{S}$ are

$$\mathbf{x}_d^t = [w_{d0}^t, w_{d1}^t, \ldots, w_{dB}^t, p_{d1}^t, \ldots, p_{dB}^t] \in \mathbb{R}_+^{2B+1} \quad (1)$$
$$\mathbf{x}_b^t = [y_{bC}^t, y_{b1}^t, \ldots, y_{bS}^t, q_{b1}^t, \ldots, q_{bS}^t] \in \mathbb{R}_+^{2S+1} \quad (2)$$
$$\mathbf{x}_s^t = [z_{sC}^t, z_{s1}^t, \ldots, z_{sS}^t] \in \mathbb{R}_+^{S+1}. \quad (3)$$

For each device d, w_{d0}^t is the locally executed tasks and w_{db}^t, p_{db}^t are the offloaded tasks and the respective transmission power to each available BS. For each BS b, $y_{b,C}^t$ denotes the tasks offloaded to the cloud and y_{bs}^t, q_{bs}^t are the offloaded tasks and the respective transmission power to each available server $s \in \mathcal{S}$. Finally, for each server s, z_{sC}^t is the offloaded tasks to the cloud and $z_{ss'}^t$ is the offloaded tasks to each available server $s' \in \mathcal{S}$, including z_{ss}^t that denotes the locally processed tasks.

For every t, it must hold $\mathbf{x}_d^t \in \Omega_d, \mathbf{x}_b^t \in \Omega_b, \mathbf{x}_s^t \in \Omega_s$, where $\Omega_d, \Omega_b, \Omega_s$ are time-invariant box constraints of the form $\{\mathbf{x} : \mathbf{0} \le \mathbf{x} \le \bar{\mathbf{x}}\}$. Note that these constraints are local, meaning that an agent needs no external information in order to satisfy them. To denote the *collective* control variable of all nodes, we use the following notation:

$$\mathbf{x}_{\mathcal{D}}^t = \{\mathbf{x}_d^t\}_{\forall d \in \mathcal{D}}, \quad \mathbf{x}_{\mathcal{B}}^t = \{\mathbf{x}_b^t\}_{\forall b \in \mathcal{B}}, \quad \mathbf{x}_{\mathcal{S}}^t = \{\mathbf{x}_s^t\}_{\forall s \in \mathcal{S}}. \quad (4)$$

Finally, we use $\mathbf{x}^t = \{\mathbf{x}_{\mathcal{D}}^t, \mathbf{x}_{\mathcal{B}}^t, \mathbf{x}_{\mathcal{S}}^t\} \in \Omega \subset \mathbb{R}_+^V$ to denote all the variables across the network, with $V = D(2B+1) + B(2S+1) + S(S+1)$.

C. Performance Objectives

Our main target is to choose the resources $\mathbf{x}^t$, such that a cumulative cost for the total delay and transmit power across all nodes is minimized. The cost function at each node depends on its control variables and it is considered time-varying in order to capture the unpredictable network dynamics at every timeslot, e.g. the randomness of the wireless channels or the network congestion levels. More precisely, we have the following cost functions per node:

$$f_d^t(\mathbf{x}_d^t) = c_d^t(w_{d0}^t) + \sum_{b \in B} c_{db}^t(w_{db}^t, p_{db}^t) \quad (5)$$

$$f_b^t(\mathbf{x}_b^t) = \sum_{s \in S} c_{bs}^t(y_{bs}^t, q_{bs}^t) + c_{bC}^t(y_{bC}^t) \quad (6)$$

$$f_s^t(\mathbf{x}_s^t) = \sum_{s' \in S} c_{ss'}^t(z_{ss'}^t) + c_{sC}^t(z_{sC}^t). \quad (7)$$

The functions $c_d^t(.), c_{ss}^t(.)$ represent local processing delay cost, $c_{db}^t(.), c_{bs}^t(.)$ capture both delay and power cost for the wireless links between nodes, $c_{bC}^t(.), c_{sC}^t(.)$ are used for the offloading delay cost to the cloud and $c_{ss'}^t(.)$ introduces the delay cost for the wired links between servers. At every timeslot t, the total cost is expressed as

$$f^t(\mathbf{x}^t) = \sum_{d \in \mathcal{D}} f^t(\mathbf{x}_d^t) + \sum_{b \in \mathcal{B}} f^t(\mathbf{x}_b^t) + \sum_{s \in \mathcal{S}} f^t(\mathbf{x}_s^t). \quad (8)$$

Similar to our model, most related works, e.g., [2], [4], [7], assume that local processing delay and communication delay, often specified via standard queuing models, are expressed by functions of only local control variables. As we will see next, this is not the case for the problem constraints, where there exist constraints that couple agents to preserve the flow of tasks inside the network.

D. Constraints

We now focus on the constraints imposed by our application. In practice, a good decision must first ensure that the incoming tasks are either processed locally or offloaded to other nodes (*flow conservation constraint*). Moreover, the transmission data rate of a wireless link must be sufficient for the assigned offloaded tasks (*rate constraint*). We model these rates using the well known Shannon capacity. All constraint functions are considered time-varying in order to model the unknown dynamics of incoming tasks and channel gains. Given that the agents first reserve resources and then the tasks are revealed, it is possible to have service violations, i.e. the provisioned resources are not adequate or cannot be realized.

For each device $d \in \mathcal{D}$, the constraint functions are

$$g_{d0}^t(\mathbf{x}_d^t) = r_d^t - w_{d0}^t - \sum_{b \in \mathcal{B}} w_{db}^t \quad (9)$$

$$g_{db}^t(\mathbf{x}_d^t) = w_{db}^t - b_w \log_2\left(1 + \alpha_{db}^t p_{db}^t\right), \forall\, b \in \mathcal{B}, \quad (10)$$

where b_w is a constant for the transmission bandwidth and α_{db}^t is an unknown variable for the channel gain, including the effect of interference and noise. In a similar way, for each BS $b \in \mathcal{B}$ we have

$$g_{b0}^t(\mathbf{x}_b^t; \mathbf{x}_{\mathcal{D}}^t) = \sum_{d \in \mathcal{D}} w_{db}^t - y_{bC}^t - \sum_{s \in \mathcal{S}} y_{bs}^t \quad (11)$$

$$g_{bs}^t(\mathbf{x}_b^t) = y_{bs}^t - b_w \log_2\left(1 + \alpha_{bs}^t q_{bs}^t\right), \forall\, s \in \mathcal{S}, \quad (12)$$

where α_{bs}^t is defined as α_{db}^t above. Notice that (11) is a *coupling constraint*, i.e. to evaluate the function at BS b, we need to know the external variables $\{w_{db}^t\}_{d \in \mathcal{D}}$ of devices. Conventionally, we denote this dependency with conditional arguments to distinguish from locally available variables, as denoted by $g_{b0}^t(\mathbf{x}_b^t; \mathbf{x}_{\mathcal{D}}^t)$. Finally, for each server $s \in \mathcal{S}$, we have only the following flow conservation constraint function

$$g_{s0}^t(\mathbf{x}_s^t; \mathbf{x}_{\mathcal{B}}^t, \mathbf{x}_{\mathcal{S}_{-s}}^t) = \sum_{b \in \mathcal{B}} y_{bs}^t + \sum_{s' \in \mathcal{S}_{-s}} z_{s's}^t - \sum_{s' \in \mathcal{S}} z_{ss'}^t - z_{sC}^t \quad (13)$$

where $\mathcal{S}_{-s}$ is used to denote the set of edge servers $\mathcal{S}$ excluding s. (13) is also a coupling constraint, since server s needs to know the external variables $\{y_{bs}^t\}_{b \in \mathcal{B}}$ of BSs and $\{z_{s's}^t\}_{s' \in \mathcal{S}_{-s}}$ of other servers.

To denote the collective set of constraints per device d and per BS b, we use the notation

$$\mathbf{g}_d^t(\mathbf{x}_d^t) = g_{d0}^t(\mathbf{x}_d^t) \cup \{g_{db}^t(\mathbf{x}_d^t)\}_{\forall b \in \mathcal{B}} \quad (14)$$

$$\mathbf{g}_b^t(\mathbf{x}_b^t; \mathbf{x}_{\mathcal{D}}^t) = g_{b0}^t(\mathbf{x}_b^t; \mathbf{x}_{\mathcal{D}}^t) \cup \{g_{bs}^t(\mathbf{x}_b^t)\}_{\forall s \in \mathcal{S}} \quad (15)$$

and write $\mathbf{g}^t(\mathbf{x}^t) = \{\{\mathbf{g}_d^t\}_{d \in \mathcal{D}}, \{\mathbf{g}_b^t\}_{d \in \mathcal{B}}, \{g_{s0}^t\}_{s \in \mathcal{S}}\} : \mathbb{R}_+^V \to \mathbb{R}^M$ to denote all constraints across the network, where $M = B(D+S) + N$.

E. Exchange of Information

Due to the coupling constraints, a fully distributed solution is no longer possible. To circumvent this, we allow each node to exchange information with other nodes, so that it can take into account the coupling constraints in its local decisions. The exact messages to exchange is part of the algorithm design, which ideally should achieve a performance close to a centralized approach with a minimal exchange of information. To limit the communication overhead, we further consider that each node can send information to other nodes only once during a timeslot, i.e. at the beginning of a timeslot. As we will see in the next section, this practical assumption introduces delayed feedback between nodes, as not all of the required information is available for a single transmission step within a timeslot.

III. OCO Formulation and Decentralized Algorithm

The goal of this section is to formulate the resource allocation, described in the previous section, as an Online Convex Optimization (OCO) problem [5] and propose an algorithm to solve it. Importantly, we explain how we adapt a centralized algorithm to transcend towards one which can run in a distributed fashion; finally, we present the performance guarantees of our solution.

A. OCO Preliminaries

We start with the basics of OCO in a centralized algorithm for two reasons: (a) it helps to introduce useful concepts and metrics; and (b) we benchmark our distributed algorithm with respect to a centralized one. To formulate the resource allocation as an OCO problem, we first need to define the sequence of events taking place for the central agent during every timeslot t.

1) The agent implements an action $\mathbf{x}^t$.
2) The environment reveals all the unknown variables, e.g. computation requests $\mathbf{r}^t$ and channel gains α_{ij}^t, which in the OCO framework can be random variables or even controlled by an adversary. Using these, the functions $f^t(.)$ and $\mathbf{g}^t(.)$ become known.
3) The agent receives or evaluates cost and constraint violations, i.e. the values $f^t(\mathbf{x}^t)$ and $\mathbf{g}^t(\mathbf{x}^t)$.
4) The agent updates its action to $\mathbf{x}^{t+1}$.

Below, we define the benchmark actions and the metrics to evaluate an algorithm that produces a sequence of actions $\{\mathbf{x}^t\}_{t=1,\dots,T}$.

Definition 1 (Static Regret)**.** *The fixed optimal action* $\mathbf{x}_*$ *and the static regret are defined as*

$$\mathbf{x}_* = \underset{x\in\Omega}{\text{argmin}} \sum_{t=1}^{T} f^t(\mathbf{x}), \text{s.t. } \mathbf{g}^t(\mathbf{x}) \le 0,\ t = 1, \dots, T \quad (16)$$

$$\text{Reg}_S(T) = \sum_{t=1}^{T} f^t(\mathbf{x}^t) - \sum_{t=1}^{T} f^t(\mathbf{x}_*). \quad (17)$$

Definition 2 (Dynamic Regret)**.** *The per slot optimal action* $\{\mathbf{x}_*^t\}_{t=1,\dots,T}$ *and the dynamic regret are defined as*

$$\mathbf{x}_*^t = \underset{x\in\Omega}{\text{argmin}} f^t(\mathbf{x}), \text{s.t. } \mathbf{g}^t(\mathbf{x}) \le 0 \quad (18)$$

$$\text{Reg}_D(T) = \sum_{t=1}^{T} f^t(\mathbf{x}^t) - \sum_{t=1}^{T} f^t(\mathbf{x}_*^t). \quad (19)$$

Definition 3 (Fit)**.** *Using the clipped constraint function* $h_m^t(\mathbf{x}^t) := [g_m^t(\mathbf{x}^t)]^+ = \max\{0, g_m^t(\mathbf{x}^t)\}$*, the fit is defined as*

$$\text{Fit}(T) = \sum_{t=1}^{T} \sum_{m=1}^{M} h_m^t(\mathbf{x}^t). \quad (20)$$

The static regret is a standard metric for evaluating OCO-based algorithms; however, there is also a growing interest recently for the dynamic regret [16], which is in principle a much harder metric. For the fit we use a clipped version of the constraint, i.e. we do not allow negative and positive values of the constraints to average out in the long run. This is a suitable modeling approach, as it would be unrealistic to assume that overprovisioning in certain timeslots can compensate for missing resources or channel rate violations in other timeslots. Finally, similarly to [8], we define the gradient of $h_m(\mathbf{x})$ as

$$\nabla h_m(\mathbf{x}) = \nabla[g_m(\mathbf{x})]^+ = \begin{cases} \mathbf{0} & \text{if } g_m(\mathbf{x}) \le 0 \\ \nabla g_m(\mathbf{x}) & \text{if } g_m(\mathbf{x}) > 0 \end{cases}$$

In the remainder, we use $\mathbf{h}^t$ for our analysis with subscripts that have the same meaning as for $\mathbf{g}^t$ in (9)-(15).

Overall, the objective of an algorithm is to establish that $\text{Reg}_S(T)$, $\text{Reg}_D(T)$ and $\text{Fit}(T)$ are all sublinear in the time horizon T [5].

B. Decomposition Across the Agents

A centralized algorithm can use the Lagrange function

$$\mathcal{L}^t(\mathbf{x}^t, \boldsymbol{\lambda}^t) = f^t(\mathbf{x}^t) + \sum_{m=1}^{M} h_m^t(\mathbf{x})\lambda_m^t \quad (21)$$

and apply the primal-dual updates [8] as follows

$$\boldsymbol{\lambda}^t = \frac{\mathbf{h}^t(\mathbf{x}^t)}{\eta\sigma},\ \mathbf{x}^{t+1} = \mathcal{P}_\Omega\left(\mathbf{x}^t - \eta\nabla_{\mathbf{x}}\mathcal{L}^t(\mathbf{x}^t, \boldsymbol{\lambda}^t)\right), \quad (22)$$

where λ_m^t is the Lagrange multiplier for constraint m, η is the gradient step size, σ is a constant and $\mathcal{P}_\Omega(\mathbf{x})$ is the projection of $\mathbf{x}$ onto Ω. Ideally, we want to perform these updates in a distributed way so that they are as close as possible to the updates of the centralized algorithm. For any node $n \in \mathcal{N}$ (device, BS or server), we have

$$\boldsymbol{\lambda}_n^t = \frac{\mathbf{h}_n^t(\mathbf{x}_n^t; \mathbf{x}_{\mathcal{E}_n}^t)}{\eta\sigma} \quad (23)$$

$$\mathbf{x}_n^{t+1} = \mathcal{P}_{\Omega_n}\left(\mathbf{x}_n^t - \eta\nabla_{\mathbf{x}_n}\mathcal{L}^t(\mathbf{x}^t, \boldsymbol{\lambda}^t)\right), \quad (24)$$

where $\mathcal{E}_n$ is the set of nodes with variables required for node n, i.e. $\mathcal{E}_d = \emptyset$, $\mathcal{E}_b = \mathcal{D}$ (11), $\mathcal{E}_s = \mathcal{B} \cup \mathcal{S}_{-s}$ (13), and $\boldsymbol{\lambda}^t$ uses the same subscripts for constraints as $\mathbf{g}^t, \mathbf{h}^t$.

We now focus on the gradients in (24) and write

$$\nabla_{\mathbf{x}_n}\mathcal{L}^t(\mathbf{x}^t, \boldsymbol{\lambda}^t) = \mathbf{H}_{nL}^t + \mathbf{H}_{nE}^t \quad (25)$$

where $\mathbf{H}_{nL}^t$ denotes the part that depends only on local cost and constraint functions at node n,

$$\mathbf{H}_{nL}^t = \nabla_{\mathbf{x}_n} f_n^t(\mathbf{x}_n^t) + \boldsymbol{\lambda}_n^{t\top}\nabla_{\mathbf{x}_n}\mathbf{h}_n^t(\mathbf{x}_n^t; \mathbf{x}_{\mathcal{E}_n}^t), \quad (26)$$

and $\mathbf{H}_{nE}^t$ describes the part that depends on external constraints from other nodes. For clarity, we provide the explicit expression for each type of node:

$$\mathbf{H}_{dE}^t = \sum_{b\in\mathcal{B}} \lambda_{b0}^t \nabla_{\mathbf{x}_d} h_{b0}^t(\mathbf{x}_b^t; \mathbf{x}_{\mathcal{D}}^t) \quad (27)$$

$$\mathbf{H}_{bE}^t = \sum_{s\in\mathcal{S}} \lambda_{s0}^t \nabla_{\mathbf{x}_b} h_{s0}^t(\mathbf{x}_s^t; \mathbf{x}_{\mathcal{B}}^t, \mathbf{x}_{\mathcal{S}_{-s}}^t) \quad (28)$$

$$\mathbf{H}_{sE}^t = \sum_{s'\in\mathcal{S}_{-s}} \lambda_{s'0}^t \nabla_{\mathbf{x}_s} h_{s'0}^t(\mathbf{x}_{s'}^t; \mathbf{x}_{\mathcal{B}}^t, \mathbf{x}_{\mathcal{S}_{-s'}}^t). \quad (29)$$

Notice that in (27)-(29), only one dimension of the gradients can be non-zero, as each node has only a single variable involved in coupling constraints of another node, i.e. w_{db}, y_{bs} and $z_{ss'}$ for devices, BSs and servers.

Remark 1. *According to the definition of* $\mathbf{h}^t(.)$ *and the flow conservation constraints, the summation terms in* (27)-(29) *simplify to* λ^t_{n0}.

C. Distributed Algorithm

As one can already notice, a distributed version of the algorithm requires exchange of information at two different steps. Specifically, a node n needs to send its variables to nodes that need them in their coupling constraints and then, receive the gradient-related feedback $\mathbf{H}^t_{nE}$. We now turn our focus on the adopted communication model and examine how this message exchange can be implemented.

An ideal scenario is shown in Fig. 2(a), where nodes can send messages at any moment during a time slot, focusing for simplicity in the link between a device and a BS. As we can see, the BS b needs to collect $\mathbf{x}^t_{\mathcal{E}_b} = \{w^t_{db}\}_{d\in\mathcal{D}}$ and perform few processing steps before it can send its feedback to the device, which can then perform the primal update $\mathbf{x}^{t+1}_d$. In practice, this is not possible in our model as nodes are allowed to send messages only once at the beginning of a time slot. We address this limitation by allowing nodes to send their feedback at the next time slot, as shown in Fig. 2(b). As a result, the device uses the outdated feedback $\mathbf{H}^{t-1}_{dE}$ for its updates.

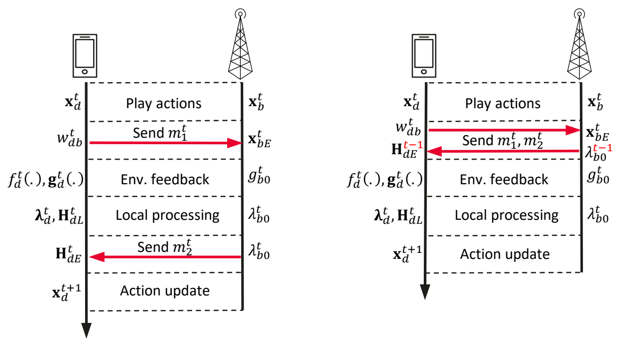
Overall, we define the following messages between nodes n, v, where $n \in \mathcal{E}_v$, and summarize them in Table I.

1) $m^t_{1,n\to v} := \mathbf{x}^t_n \in \mathbf{x}^t_{\mathcal{E}_v}$; required to evaluate the coupling constraint $h^t_{v0}(\mathbf{x}^t_v; \mathbf{x}^t_{\mathcal{E}_v})$ at node v, which is then used for the dual update of λ^t_{v0} in (23) and the local term $\mathbf{H}^t_{vL}$.
2) $m^t_{2,v\to n} := \lambda^{t-1}_{v0}$; feedback required to evaluate $\mathbf{H}^{t-1}_{nE}$ for the primal update at node n.

Let us now consider the primal/dual updates of the proposed approach. First, the dual update (23) remains the same and thus, identical to the centralized case. Then, for the primal update (24), only the gradient term is modified and approximated by

$$\nabla_{\mathbf{x}_n} \hat{\mathcal{L}}(\mathbf{x}^t, \boldsymbol{\lambda}^t) = \mathbf{H}^t_{nL} + \mathbf{H}^{t-1}_{nE}. \tag{30}$$

The steps of our proposed distributed algorithm are presented in Algorithm 1 for any node n.



(a) Ideal case (b) Proposed approach

Fig. 2: Exchanged messages during time slot t between device d and BS b.

TABLE I: Messages (m^t_1, m^t_2) between nodes

From\To	**Device** d'	**BS** b'	**Server** s'
Device d	–	$w^t_{db'}$	–
BS b	λ^{t-1}_{b0}	–	$y^t_{bs'}$
Server s	–	λ^{t-1}_{s0}	$z^t_{ss'}$, λ^{t-1}_{s0}

Algorithm 1 Distributed OCO for node n

```
1: Initialize: parameters $\sigma$, $\eta$
2: Set first action $\mathbf{x}^1_n \in \Omega_n$ and define $\boldsymbol{\lambda}^0_n = \mathbf{0}$.
3: for $t = 1, \ldots, T$ do
4:     Play action $\boldsymbol{x}^t_n$
5:     Send messages $m^t_{1,n\to v}$ to nodes $v : n \in \mathcal{E}_v$
6:     Receive from environment functions $f^t_n(.)$ and $\boldsymbol{g}^t_n(.)$
7:     Receive feedback messages $m^t_{2,v\to n}$ from nodes $v$
8:     Compute $\boldsymbol{\lambda}^t_n$ with (23)              ▷ Dual update
9:     Update $\boldsymbol{x}^{t+1}_n$ with (24),(30)          ▷ Primal update
10: end for
```

D. Performance Guarantees

We make the following standard assumptions, widely used in online learning literature (e.g., see [8], [11]), and then formally state our main theorem.

Assumption 1.

- *(i) Set* Ω_n *is bounded and convex; specifically it holds that* $\|\mathbf{x}_n - \mathbf{y}_n\| \le R$, $\forall \mathbf{x}_n, \mathbf{y}_n \in \Omega_n$, *for* $n \in \mathcal{D}, \mathcal{B}, \mathcal{S}$.
- *(ii) For* $t = 1, \ldots, T$, *functions* f^t_n *and* $g^t_{n,i}$ *are convex and Lipschitz with* $\|\nabla_{\mathbf{x}_n} f^t_n\| \le F'$ *and* $\left\|\nabla_{\mathbf{x}_n} g^t_{n,i}\right\| \le G'$, *for* $n \in \mathcal{D}, \mathcal{B}, \mathcal{S}$ *and* $\mathbf{g}^t_n = \{g^t_{n,i}\}_{i=1,\ldots,M_n}$ *(with* M_n *the number of constraints at node* n*).*

Below we write a list of implications that we use for the proof of our theorem. First, f^t_n and $g^t_{n,i}$ are both bounded, i.e., $|f^t_n| \le F$, $|g^t_{n,i}| \le G''$. Second, since $\left\|\nabla g^t_{n,i}\right\| \le G'$, then $\left\|\nabla h^t_{n,i}\right\| \le G'$ (comes from the definition of gradient of h). Third, since $g^t_{n,i}$ is bounded, $h^t_{n,i}$ is also bounded by definition; hence $|h^t_{n,i}| \le G''$ and $\|\mathbf{h}^t\| \le G$. For simplicity we write that $|f^t_n|, \|\nabla f^t_n\| \le F$ and $|h^t_{n,i}|, \left\|\nabla h^t_{n,i}\right\|, \|\mathbf{h}^t_n\| \le G$. Fourth, since $g^t_{n,i}$ is convex, then $h^t_{n,i}$ is as well.

Proofs for the first and fourth implications can be found in Appendix B of the extended version of our paper [17].

Theorem 1. *Given Assumption 1, and $\sigma > 3KG^2$, Algorithm 1 guarantees that*

$$\mathrm{Reg}_S(T) \le \frac{R^2}{2\eta} + \frac{2REG^2}{\eta\sigma} + \frac{5}{2}\eta N F^2 T \triangleq U_{sr}, \qquad (31)$$

$$\mathrm{Reg}_D(T) \le U_{sr} + \frac{R}{\eta} V(\boldsymbol{x}_*^{1:T}), \qquad (32)$$

$$\mathrm{Fit}(T) \le \sqrt{\frac{\eta\sigma}{\beta} MT(U_{sr} + 2NFT)}, \qquad (33)$$

where E is the number of edges in the network topology, $K = D(4B + \frac{3}{2}) + B(4S + \frac{3}{2}) + S(\frac{5}{2}S - 1)$, $\beta = 1 - \frac{3KG^2}{\sigma}$ and $V(\mathbf{x}_*^{1:T}) = \sum_{t=1}^{T} \left\|\mathbf{x}_*^t - \mathbf{x}_*^{t-1}\right\|$.

Proof. See Appendix A of the extended version of our paper [17]. □

An immediate implication of Theorem 1 is that, for step size $\eta = \mathcal{O}(T^{-1/2})$, we have

- $\mathrm{Reg}_S(T) = \mathcal{O}(T^{1/2})$
- $\mathrm{Reg}_D(T) = \mathcal{O}(\max\{T^{1/2}, T^{1/2}V(\mathbf{x}_*^{1:T})\})$
- $\mathrm{Fit}(T) = \mathcal{O}(T^{3/4})$

We conclude that our distributed algorithm, although using outdated Lagrange multipliers, achieves sublinear static regret and fit; if additionally, $V(\mathbf{x}_*^{1:T}) = o(T^{1/2})$, then dynamic regret is also sublinear. In fact, we achieve the same order of bounds as the centralized algorithm in [4] (in the case where the authors ignore the outages), which tackles the same setting. Note that choosing $\eta = \mathcal{O}(T^{-1/2})$ yields the minimum regret while preserving a sublinear fit.

IV. Performance Evaluation

A. Simulation Setup

Topology and box constraints. We assume a fully connected setting with nodes $D, B, S = 2$. Moreover, the upper bounds of the control variables are as follows: for every device d, $\overline{w}_{d0} = 2$, $\overline{w}_{db} = 25$, and $\overline{p}_{db} = 25$, for every BS b, $\overline{y}_{bC} = 30$, $\overline{y}_{bs} = 25$, and $\overline{q}_{bs} = 25$ and for every server s, $\overline{z}_{sC} = 50$, $\overline{z}_{ss} = 15$ and $\overline{z}_{ss'} = 10$.

Costs. We model the cost functions using expressions for delay (from $M/M/1$[1]) and power [7]. The local processing delay of node n for x tasks is $c_n(x) = 1/(\overline{x} - x)$, where $\overline{x}$ denotes the capacity of node n; this cost is used to model $c_d^t(w_{d0}^t)$ and $c_{ss}^t(z_{ss}^t)$. Then, the cost related to wireless offloading, i.e., $c_{db}^t(w_{db}^t, p_{db}^t)$ and $c_{bs}^t(y_{bs}^t, q_{bs}^t)$, is modeled as $c_{nn'}^t(x, y) = 1/(R_{nn'}^t(y) - x) + \frac{1}{2}y^2$, where $R_{nn'}^t(y) = b_w \log_2\left(1 + \alpha_{nn'}^t y\right)$ is the channel rate. Finally, the delays for offloading to the cloud, i.e. $c_{bC}^t(y_{bC}^t)$ and $c_{sC}^t(z_{sC}^t)$, are modeled as $c_{nC}^t(x) = d_{nC}^t x$, where d_{nC}^t is a time-varying unknown environment parameter.

Unknown variables. For each time slot t we need: (a) channel gains $\alpha_{db}^t, \alpha_{bs}^t$, (b) cloud delay costs d_{bC}^t, d_{sC}^t, and (c) traffic requests $\mathbf{r}^t$. We model (a) and (b) as random variables sampled from $\mathcal{U}(8, 15)$ and $\mathcal{U}(3, 10)$ respectively.

[1]To avoid numerical instabilities, e.g., $M/M/1$ delay becoming infinite, we use standard convex extensions [4].

For (c), we mainly use the public Milano dataset [18] which includes Internet traffic demand (measured in MBs) that arrives to BSs in the form of timeseries. We extract the traffic of 2 BS, and use it to simulate the computation demand of our $D = 2$ devices. We also provide results using synthetic demands, drawn from $\mathcal{U}(1, 10)$.

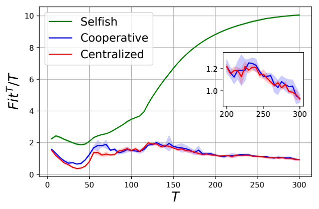
Metrics and Baselines. The performance is evaluated using the static and dynamic regrets (the respective benchmarks are found using CVXPY [19]), and the fit. We plot these metrics for proposed Algorithm 1, which we call *Cooperative*, and for two more *OCO-based* baselines with different level of information each. The first one, *Selfish*, is a distributed algorithm with no information exchange between nodes; i.e. $\mathbf{H}_{nE}^t = \mathbf{0}$ in (25). While the second, *Centralized*, is an algorithm where a centralized controller, with access to all information, makes the exact OCO updates for our problem (i.e., is optimal). The latter describes the State-of-Art algorithm presented in [4].

B. Simulation Results

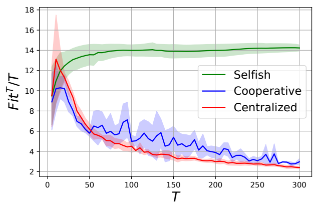
In all our plots, the x-axis represents the horizon T, which we vary from 0 to 300 time slots. For each algorithm, we plot the average value across 4 independent runs and with shade the corresponding standard deviation. Notice that all metrics are normalized by T.

Our setup is challenging for a distributed algorithm, as the flow conservation constraints *couple* different nodes. To this end, we first investigate the $\mathrm{Fit}(T)$ for the Milano and the synthetic datasets in Figs. 3(a), 3(b). The fit of *Centralized* quickly converges to zero, suggesting that it learns to play actions that respect most of the time-varying constraints; the reason is that it performs the best possible primal and dual updates with the freshest information. The fit of (proposed method) *Cooperative* converges almost together with *Centralized* for Milano demands (Fig. 3(a)), and slightly slower for the synthetic ones (Fig. 3(b)). Therefore, the modified gradients used by *Cooperative* suffice in order to satisfy the constraints in the long-run. Finally, *Selfish* exemplifies the necessity of at least some information exchange between the nodes; we can see in both figures the fit increasing. This behavior is expected, as the $\mathrm{Fit}(T)$ of *Centralized* and *Cooperative* is sublinear, whereas the one of *Selfish* can be shown to be linear as its updates totally ignore the coupling (flow conservation) constraints.

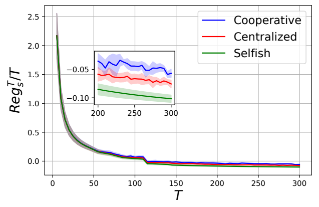
Having discussed the "feasibility" aspect of these algorithms (i.e., how they perform in terms of constraints) we now focus on the objective function and in particular on the regrets in Figs. 3(c), 3(d). We plot these metrics only for Milano dataset; but the respective plots for the synthetic one are similar. A first observation is that the regrets of *Selfish* are the lowest among all three methods. This should not come as a surprise, since by construction, the algorithm solves a more relaxed version of the problem (ignores flow constraints) and therefore can achieve better cost values. *Centralized* has slightly higher regrets, which



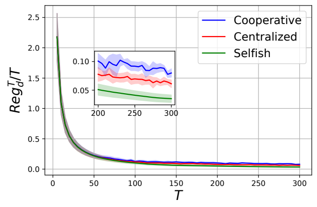
(a) Fit - Milano



(b) Fit - Synthetic

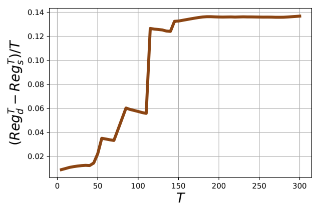


(c) Static regret - Milano

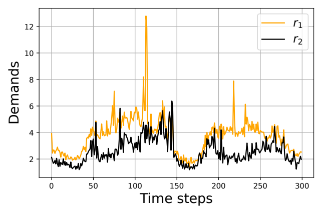


(d) Dynamic Regret - Milano

Fig. 3: Performance metrics vs horizon length T



(a)



(b)

Fig. 4: Milano dataset: (a) Difference of regrets (and benchmarks) vs horizon T; (b) Demands of the devices over time.

is justified by its effort to also satisfy the fit. Finally, *Cooperative* has regrets that also go to zero and are very close to the ones of (the optimal baseline) *Centralized*.

Finally, we comment on the jump at $T \approx 110$ in Fig. 3(c), which is not present in Fig. 3(d). The difference of the two plots has to do with the benchmarks; in Fig. 4(a), the y-axis is in fact the cost gap between them, i.e. $\frac{1}{T}\sum_{t=1}^{T}\left(f^t(x_*) - f^t(x_*^t)\right)$; and in that plot we obviously see that same jump. This behavior can be explained by the change of demand in Fig. 4(b). Essentially, the static benchmark, for $T > 110$, solves an optimization problem by finding a feasible x_* for that extreme demand (at $T \approx 110$), and will be then *more constrained* compared to the dynamic benchmark, which solves the problem for each t individually.

V. Conclusions

We revisit the problem of resource allocation in an IoT network, where devices can process part of the traffic requests, and/or offload the rest to more powerful computational entities (cloud, edge servers). The distributed nature of the setting, as well as the unpredictable environment, motivated us to model the network nodes (devices, BSs, servers) as distributed OCO actors. However, the nodes of the network are naturally coupled by flow conservation constraints at each node, which we model as long-term constraints, and as a result a fully decentralized algorithm is no longer possible.

In order to address this challenge, we propose a distributed OCO algorithm with limited communication between nodes, which practically leads to partially outdated gradient updates. Nevertheless, we show theoretically that our algorithm achieves sub-linear regret bound $\mathcal{O}(T^{1/2})$ and sub-linear constraint violation bound $\mathcal{O}(T^{3/4})$, which is the same order of bounds as a centralized algorithm for this setting. Numerical results based on real data traces confirm our theoretical findings.

References

[1] U. Cisco, "Cisco annual internet report (2018–2023) white paper," *Cisco: San Jose, CA, USA*, vol. 10, 2020.
[2] T. Chen, Y. Shen, Q. Ling, and G. B. Giannakis, "Online learning for "thing-adaptive" fog computing in iot," in *IEEE Asilomar*, 2017.
[3] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of things journal*, vol. 3, 2016.
[4] A. Chouayakh and A. Destounis, "Towards no regret with no service outages in online resource allocation for edge computing," in *IEEE ICC*, 2022.
[5] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *(ICML-03)*, 2003, pp. 928–936.
[6] R. Deng, R. Lu, C. Lai, and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *IEEE ICC*, 2015.
[7] G. Lee, W. Saad, and M. Bennis, "An online secretary framework for fog network formation with minimal latency," in *IEEE ICC*, 2017.
[8] J. Yuan and A. Lamperski, "Online convex optimization for cumulative constraints," in *NIPS*, 2018.
[9] M. Mahdavi, R. Jin, and T. Yang, "Trading regret for efficiency: online convex optimization with long term constraints," *The Journal of Machine Learning Research*, 2012.
[10] M. J. Neely and H. Yu, "Online convex optimization with time-varying constraints," *arXiv preprint arXiv:1702.04783*, 2017.
[11] N. Liakopoulos *et al.*, "Cautious regret minimization: Online optimization with long-term budget constraints," in *ICML*, 2019.
[12] X. Cao, J. Zhang, and H. V. Poor, "Constrained online convex optimization with feedback delays," *IEEE Transactions on Automatic Control*, vol. 66, no. 11, pp. 5049–5064, 2020.
[13] X. Yi, X. Li, L. Xie, and K. H. Johansson, "Distributed online convex optimization with time-varying coupled inequality constraints," *IEEE Transactions on Signal Processing*, vol. 68, pp. 731–746, 2020.
[14] V. B. C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren, and G. Tashakor, "Handling service allocation in combined fog-cloud scenarios," in *IEEE ICC*, 2016.
[15] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: Towards minimizing delay in the internet of things," in *IEEE EDGE*, 2017.
[16] A. Jadbabaie *et al.*, "Online optimization: Competing with dynamic comparators," in *Artificial Intelligence and Statistics.* PMLR, 2015, pp. 398–406.
[17] S. Kriouile, D. Tsilimantos, and T. Giannakas, "Distributed no-regret edge resource allocation with limited communication," *arXiv preprint arXiv:2304.05355*, 2023.
[18] T. Italia, "Telecommunications - SMS, Call, Internet - MI," 2015.
[19] S. Diamond and S. Boyd, "Cvxpy: A python-embedded modeling language for convex optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, 2016.