Towards no regret with no service outages in online resource allocation for edge computing

Ayman Chouayakh
Huawei Technologies, France
aymanchouayakh90@gmail.com

Apostolos Destounis
Huawei Technologies, France
apostolos.destounis@huawei.com

***Abstract*—This paper presents a new framework to deal with the unpredictable nature of demands in Edge Computing. We formulate the edge resource reservation problem as a constrained Online Convex Optimization (OCO) problem with time-varying cost and constraint functions and introduce a new metric, termed Outage Fit, to capture the fact that constraint violation by even a small amount can lead to service outages. Based on this notion, an online algorithm which provably learns to avoid service outages and constraint violations while resulting in a good latency cost is proposed and numerical results, based on real data traces showcase the improvement with respect to the state of the art algorithms.**

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

I. Introduction

Internet of Things (IoT) plays a remarkable role in improving the quality of life and growing the world's economy by introducing new use cases [1]. In many use cases such as driverless vehicles, the level of latency is vital [2]. Traditional centralized cloud-based network are unfit for these applications due to latency that results from cloud processing. In order to support the deployments of those systems, Edge Computing (EC) was proposed [3]. EC deploys servers nears the devices in order to extend the capabilities of computation and network connection from the cloud to the edge of the network with the ultimate objective of bringing intelligence to the edge systems to support the changing IoT environment.

In this context, to deal with uncertainty in dynamic environments, online learning has emerged as a promising solution [4]. More specifically, the framework of Online Convex Optimization (OCO) is an ideal fit for edge computing settings where the network topology and tasks can vary unpredictably and have unpredictable returns due to the variation of wireless channel conditions and service demands. In this paper, we use OCO to make online resource reservation decisions that lead to low latency. Since resource reservations are made before the demands arrive, a challenge, in addition to achieving low latency, is to learn how to avoid service outages that result from possible under-allocation of resources with respect to the demand.

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.

In general, OCO seeks to find a sequence of decisions, that minimizes the sum of convex loss functions over time. The challenge is that the function itself is revealed to the decision maker only after the action is chosen and may vary arbitrarily as if an adversary selected it to disrupt system control decisions. Due to the lack of information about the current loss function, the decision maker cannot select an optimal decision at each time slot. Instead, the decision maker aims at minimizing the regret [5], i.e., the performance gap between the online decision sequence and some performance benchmark.

Most works on OCO benchmark algorithms with *static regret* which measures the difference of costs between the online solution and best static solution in hindsight [4]. However, static regret is not necessarily a sufficient metric in dynamic IoT environments. A more suitable metric is the dynamic regret in which the benchmark is formed via a sequence of best dynamic solutions for the instantaneous cost minimization. Recent works [6], [7] benchmark algorithms with dynamic regret under time-invariant constraints. However, in real environments we often have to deal with time-varying constraints. Since the constraint functions are also revealed after making the decision, it is impossible to satisfy them at each time slot. Therefore a more realistic goal is to find a sequence of online decisions that minimizes the regret and ensures that the constraints are satisfied in the long term on average [8]. Most of the papers in this line of work suppose that constraint violations at some times could be compensated by strictly feasible decisions at other times. This is appropriate only for constraints that have a cumulative nature, such as in applications with long term energy budgets. But for other applications, see e.g. [9], constraint violations at some steps can not be compensated by other times; this type of constraints is referred to as clipped constraints.

The use of OCO algorithms for online routing and resource reservation in networks has recently received attention from the research community. Indeed, treating the traffic demands as controlled by an adversary is a means to bypass its unpredictable nature and its difficulty to model. Chen et al. [10] study online resource allocation in a more general setting, which is applied in [11] for resource allocation in IoT systems. These works characterize the

static and dynamic regret, as well as constraint violations, under the model that underprovisioning at some time instances can be compensated by overprovisioning in other time instances. A similar setting is examined in [12] for CPU and memory reservations in a cloud server, where the cloud resource reservations must satisfy a time-average constraint for violations (as opposed to instantaneous ones). Finally, Donassolo et al. [13] focus on the migration of the components of fog computing tasks and compare multiple service migration algorithms.

The present paper studies a framework with clipped time-varying constraints and provides new guarantees for constraint violations by introducing a new metric termed *Outage Fit*. The intuition behind this new metric is that most of the existing metrics that capture constraint violations in the literature are not sufficient: we may have good performances with respect to those metrics even when constraints are violated at each time slot; see Section IV for more details. Relative to existing works, the main contributions of this paper are:

1) We formulate online edge resource reservation as a constrained online convex problem with time-varying costs and constraints, by defining the delays resulting from a resource allocation decision from queueing-theoretic models. Contrary to prior work, the constraints should be satisfied at every time instance, and underprovisioning cannot be compensated by later overprovisioning.
2) We introduce a new metric called outage fit in order to capture the fact that even small constraint violations can be undesirable as they lead to service outages due to unacceptable long delays caused by overloading some parts of the system.
3) We propose an online algorithm that provably learns to avoid service outages and achieve asymptotically zero amount of constraint violations while keeping the "no regret" property for the latency cost of the computation tasks.

II. System model and Problem Formulation

This section presents the system model and formalizes the resource allocation problem as an OCO problem with constraints.

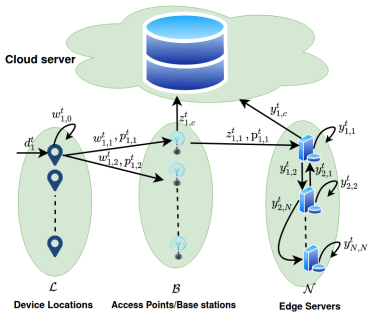


Figure 1: Illustration of the edge computing framework.

A. Model components

We consider a network composed of a set $\mathcal{L}$ ($|\mathcal{L}| = L$) of device locations (LOCs), a set $\mathcal{B}$ ($|\mathcal{B}| = B$) of Base Stations (BSs), a set $\mathcal{N}$ ($|\mathcal{N}| = N$) of Edge Servers (ESs) and a remote data center in the cloud. Time is slotted but each slot is long enough to assume that a large number of devices and computing tasks have arrived in its duration. Per time slot t, each location l has an arrival rate of computing tasks d_l^t. We make no assumptions on how the computing demands vary, in order to capture the highly dynamic and unpredictable conditions that is envisaged in future IoT and edge computing environments. Devices at location l can process computing tasks at rate c_l, while Edge Server n can process computation tasks at rate c_n. The objective is to reserve at the beginning of t and for the rest of the duration of the slot, without a prior knowledge of traffic demands and channel conditions, the following resources (see Fig. 1 and Table I):

- Computation tasks from LOC l to BS b: $w_{l,b}^t \leq d_l^{\max}$.
- Computation tasks at LOC l to be processed locally: $w_{l,0}^t \leq c_l$.
- Computation tasks from BS b to be routed to ES n: $z_{b,n}^t \leq \bar{z}_{b,n}$.
- Computation tasks from BS b to be routed to the cloud $z_{b,c}^t \leq \bar{z}_{b,c}$.
- Computation tasks from ES n to be rerouted to ES m: $y_{n,m}^t \leq \bar{y}_{n,m}$.
- Computation tasks from ES n to be rerouted to the cloud: $y_{n,c}^t \leq \bar{y}_{n,m}$.
- Computation tasks at EC n to be processed locally: $y_{n,n}^t \leq c_n$.
- Transmission power from LOC l to BS b: $p_{l,b}^t \leq \bar{p}_{l,b}$.
- Transmission power from BS b to ES n: $\mathrm{p}_{b,n}^t \leq \bar{\mathrm{p}}_{b,n}$.

In addition, the resource allocation decision must satisfy the following flow conservation constraints

$$\sum_{l\in\mathcal{L}} w_{l,b}^t \leq z_{b,c}^t + \sum_{n\in\mathcal{N}} z_{b,n}^t \forall b \in \mathcal{B}$$

$$\sum_{b\in\mathcal{B}} z_{b,n}^t + \sum_{m\in\mathcal{N}_n^{\text{in}}} y_{m,n}^t \leq \sum_{m\in\mathcal{N}_n^{\text{out}}} y_{n,m}^t + y_{n,n}^t + y_{n,c}^t \forall n \in \mathcal{N},$$

where $\mathcal{N}_n^{\text{in}}$ and $\mathcal{N}_n^{\text{in}}$ represent the sets of edge servers with in-coming links to edge server n and those with out-going links from edge server n.

For the rest of the paper, we will denote $\boldsymbol{x}^t = (\boldsymbol{w}^t, \boldsymbol{z}^t, \boldsymbol{y}^t, \boldsymbol{p}^t, \mathbf{p}^t)^\top$ be the allocation vector. Then, $\boldsymbol{x}^t \in \Omega$, where the latter set is defined as the set of all allocation vectors with non-negative elements $\boldsymbol{x} \geq 0$ satisfying the box and flow conservation constraints above.

B. Cost function

The components of the cost are as follows:

- $c_{l,0}^t(w_{l,0}^t)$: the mean computation delay of the amount of traffic $w_{l,0}^t$ processed locally at location l. Since we assume that the duration of a slot is large enough

for many computation jobs to arrive to the system, these delays can be given by a standard queueing theoretic model as proportional to $\frac{1}{w^t_{l,0}} \frac{w^t_{l,0}/c_l}{1-w^t_{l,0}/c_l}$. This is well defined only for $w^t_{l,0} < c_l$, thus, we will use the following convex extension for $c_l \geq w^t_{l,0}$:

$$c^t_{l,0}(w^t_{l,0}) = \begin{cases} \frac{1}{c_l - w^t_{l,0}} & , w^t_{l,0} \leq (1-a)c_l \\ \frac{1}{ac_l} + \frac{\left(w^t_{l,0} - (1-a)c_l\right)}{a^2} & , w^t_{l,b} \geq (1-a)c_l \end{cases}.$$

- $c^t_{l,b}(w^t_{l,b}, p^t_{l,b})$: the transmission delay of the offloaded jobs from location l to the base station b. Similar to the previous cost, the mean transmission delay of a job is given as $\frac{1}{R^t_{l,b}} \frac{w^t_{l,b}/R^t_{l,b}}{1-w^t_{l,b}/R^t_{l,b}}$, where $R^t_{l,b}$ is the channel capacity between LOC l and BS b. It is given by $R^t_{l,b} = b_w \log_2\left(1 + \alpha^t_{l,b} p^t_{l,b}\right)$, where $\alpha^t_{l,b}$ is a parameter that captures interference and noise and b_w is the system bandwidth. Since the channel can vary unpredictably, we may have that $R^t_{l,b} > w^t_{l,b}$, in which cases there will be a service outage due to acceptably high delays (or a flow control mechanism will start dropping jobs to ensure acceptable delays to the rest of the transmitted jobs). We will use the convex extension of this function, $c^t_{l,b}(w^t_{l,b}, p^t_{l,b}) =$

$$\begin{cases} \frac{1}{R^t_{l,b} - w^t_{l,b}} & , \frac{w^t_{l,b}}{R^t_{l,b}} \leq (1-a) \\ \frac{1}{aR^t_{l,b}} + \frac{\left(w^t_{l,0} - (1-a)R^t_{l,b}\right)^2}{a^2} & , \frac{w^t_{l,b}}{R^t_{l,b}} \geq (1-a) \end{cases}$$

  and handle the service outage caused in cases with overloads with the constraints defined in II.C.
- $c^t_{b,n}(z^t_{b,n}, \mathrm{p}^t_{b,n})$: the transmission delay $z^t_{b,n}$ under the transmission power $\mathrm{p}^t_{b,n}$. The delay depends on the channel capacity given by $R^t_{b,n} = b_w \log_2\left(1 + \alpha^t_{b,n}\mathrm{p}^t_{b,n}\right)$, where $\alpha^t_{b,n}$ is a parameter that captures noise and channel conditions on the link. $c^t_{b,n}(z^t_{b,n}, \mathrm{p}^t_{b,n})$ is similar to $c^t_{l,b}(w^t_{l,b}, p^t_{l,b})$.
- $c^t_{b,c}(z^t_{b,c})$: the mean delay of the offloaded traffic $z^t_{b,c}$ from BS b to the cloud. Specifically, as the computation delay is usually negligible for data centers, the latency for cloud offloading is mainly due to the communication delay [11]. Following [14], we suppose that the transmission time to the cloud is given by $d^t_{b,c} z^t_{b,c}$, where $d^t_{b,c}$ denotes the delay of the transmission path from BS b to the cloud. $c^t_{b,c}(z^t_{b,c}) = d^t_{b,c} z^t_{b,c}$
- $c^t_{n,n}(y^t_{n,n})$: the mean processing delay of computation demands $y^t_{n,n}$ at ES n is similar to $c^t_{l,0}(w^t_{l,0})$ (replacing c_l by c_n, c_n is the processing capacity of ES n).
- $c^t_{n,c}(y^t_{n,c})$: the mean delay of the offloaded amount $y^t_{n,c}$ from ES n to the cloud. Similarly to $c^t_{b,c}(z^t_{b,c})$, it cost is given by: $c^t_{n,c}(y^t_{n,c}) = d^t_{n,c} y^t_{n,c}$

Let $f^t(\boldsymbol{x}^t)$ be the aggregate cost as a result of the resource reservation $\boldsymbol{x}^t$ at slot t. It can be written as

$$\begin{aligned} f^t(\boldsymbol{x}^t) := \beta_1 \sum_{l\in\mathcal{L}} \left(c^t_{l,0}(w^t_{l,0}) + \sum_{b\in\mathcal{B}} c^t_{l,b}(w^t_{l,b}, p^t_{l,b}) \right) \\ + \beta_1 \sum_{b\in\mathcal{B}} \left(c^t_{b,c}(z^t_{b,c}) + \sum_{n\in\mathcal{N}} c^t_{b,n}(z^t_{b,n}, \mathrm{p}^t_{b,n}) \right) \\ + \beta_1 \sum_{n\in\mathcal{N}} \left(c^t_{n,n}(y^t_{n,n}) + c^t_{n,c}(y^t_{n,c}) \right) \\ + \beta_2 \Bigg(\sum_{l\in\mathcal{L}, b\in\mathcal{B}} (p^t_{l,b})^2 + \sum_{b\in\mathcal{B}, n\in\mathcal{N}} (\mathrm{p}^t_{b,n})^2 \Bigg), \end{aligned}$$

where β_1 and β_2 are weights that reflect the relative importance of the delay and energy consumption.

C. Constraints

In addition to minimizing the cost, a good resource allocation decision must also ensure that, first, each of the incoming computation demands will be either served locally or offloaded to the edge/cloud network, that is

$$g^t_l(\boldsymbol{x}^t) = d^t_l - w^t_{l,0} - \sum_{b\in\mathcal{B}} w^t_{l,b}, \forall l \in \mathcal{L}$$

and, second, that the rates in the wireless access and channels between the BSs and ESs are enough to carry all the traffic assigned, that is

$$g^t_{l,b}(w^t_{l,b}, p^t_{l,b}) = w^t_{l,b} - b_w \log_2\left(1 + \alpha^t_{l,b} p^t_{l,b}\right), \forall l, b \in \mathcal{L}\times\mathcal{B}$$

$$g^t_{b,n}(z^t_{b,n}, \mathrm{p}^t_{b,n}) = z^t_{b,n} - \mathrm{b}_w \log_2\left(1 + \alpha^t_{b,n}\mathrm{p}^t_{b,n}\right), \forall b, n \in \mathcal{B}\times\mathcal{N}.$$

Notation in summarized in the following table.

$\mathcal{L}$	the set of locations
$\mathcal{B}$	the set of base stations
$\mathcal{N}$	the set of edge servers
d^t_l	demands of computational tasks at LOC l
$w^t_{l,0}$	computational tasks from LOC l processed locally
$w^t_{l,b}$	computational tasks from LOC l to BS b
$z^t_{b,n}$	computational tasks from BS b to be routed to ES n
$z^t_{b,c}$	computational tasks from BS b to be routed to the cloud server
$y^t_{n,m}$	computational tasks from ES n to be routed to ES m
$y^t_{n,c}$	computational tasks from ES n to the cloud server
$y^t_{n,n}$	computational tasks from ES n processed locally
$p^t_{l,b}$	transmission power from LOC l to BS b
$\mathrm{p}^t_{b,n}$	transmission power from BS b to ES n
$\boldsymbol{x}^t$	$\boldsymbol{x}^t = (\boldsymbol{w}^t, \boldsymbol{z}^t, \boldsymbol{y}^t, \boldsymbol{p}^t, \mathbf{p}^t)$ allocation vector
$\mathbf{g}^t(.)$	time-varying constraint function
$f^t(.)$	cost function

Table I: Notations

III. Design Objectives

Since the computing requests can vary arbitrarily, we make no assumptions and, instead, treat them if controlled by an adversary who tries to harm the performance of the system. In this setting, meaningful metrics are notions of regret [4], [5], which quantify the performance of the algorithm against a benchmark action and fit, which

quantifies how well the constraints are satisfied. We detail these concepts next.

- **Static regret:** Static regret is adopted as a metric by standard OCO schemes. It measures the difference between the loss incurred by the decisions of an online algorithm and the best static decision in hindsight $\boldsymbol{x}_*$:

$$\mathrm{Reg}_s^T = \sum_{t=1}^{T} f^t(\boldsymbol{x}^t) - \min_{\boldsymbol{x}\in\Omega:\boldsymbol{g}^t(\boldsymbol{x})\leq 0,\forall t} \sum_{t=1}^{T} f^t(\boldsymbol{x})$$

- **Dynamic regret:** Here the performance of the online algorithm is compared to the sequence of best dynamic decisions $\{\boldsymbol{x}_*^t\}$ for the instantaneous cost minimization, that is:

$$\mathrm{Reg}_d^T = \sum_{t=1}^{T} f^t(\boldsymbol{x}^t) - \sum_{t=1}^{T} \min_{\boldsymbol{x}\in\Omega:\boldsymbol{g}^t(\boldsymbol{x})\leq 0} f^t(\boldsymbol{x}) \quad (1)$$

  The dynamic regret is always larger than the static regret, as the latter is more restricted.

- **Fit:** The fit is introduced to measure the accumulated violation of constraints. In our scenario, for the constraints related to the transmission rates and channel capacities in Sec. III.C, constraint violations from one step can not be compensated by strictly feasible decisions (e.g. overprovisioning) in other steps because the slots last for a relatively long time and a violation leads to a service outage due to overloading some nodes. The most relevant definition of the constraint fit is therefore an extension of the definition in [9] for time varying constraints

$$\mathrm{Fit}^{\mathrm{T}} := \sum_{t=1}^{T}\sum_{i=1}^{M} [\mathrm{g}_i^t(\boldsymbol{x}^t)]^+ \quad (2)$$

  so that these cancellation effects cannot occur.

The goal of online learning in the literature is to obtain algorithms with the "no regret" property, that is algorithms such that the regret and fit scale sublinearly with the horizon T. However, in our case, even having $\mathrm{Fit}^T = o(T)$ does not necessarily reflect a good performance. Indeed, an algorithm may have a fit that is sublinear in T but some (or even all) constraints are violated by a small amount at every slot, for example if $g_i^t(\boldsymbol{x}^t) = \mathcal{O}(1/\log(T))$. Since a constraint violation even by a small amount leads to service outages due to unacceptable delays, this is clearly an undesirable operating point. We, therefore introduce the corresponding notion of outage fit:

Definition 1 (Outage Fit)**.** *We define the outage fit of an online resource allocation algorithm as the total number of constraints that have been violated, i.e.*

$$\mathrm{OutageFit}^T = \sum_{t=1}^{T}\sum_{i} 1_{\{g_i^t(x^t)\geq 0\}}$$

Having an outage fit that is sublinear with T would lead to learning to avoid service outages altogether In addition, the fit is still a meaningful performance metric because an algorithm with a good outage fit may lead to few but very severe overflows, which is also undesirable. In addition, $\mathbb{1}_{\mathbf{g}^t\geq 0}$ is not convex. We design an algorithm that resolves both issues in the next Section.

IV. Online Algorithm

The first step in order to deal with the non-convexity of the outage fit is to define the function

$$\mathbf{g}_\epsilon^t(\boldsymbol{x}) := \frac{1}{\epsilon}(\mathbf{g}^t(\boldsymbol{x}) + \epsilon\mathbf{1}), \quad (3)$$

which has the following convenient property:

Lemma 1. *For any* $0 < \epsilon \leq 1$*, if* $\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}[\frac{1}{\epsilon}(g^t+\epsilon)]^+ = 0$ *then* $\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}[g^t]^+ = 0$ *and* $\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\mathbb{1}_{g^t>0} = 0$

Proof. The result is a consequence of the following:

- $[\frac{1}{\epsilon}\mathbf{g}^t + 1]^+ > \frac{1}{\epsilon}[\mathbf{g}^t]^+ \geq [\mathbf{g}^t]^+$
- $[\frac{1}{\epsilon}\mathbf{g}^t + 1]^+ \geq [\frac{1}{\epsilon}\mathbf{g}^t + 1]^+\mathbb{1}_{\mathbf{g}^t>0} \geq \mathbb{1}_{\mathbf{g}^t>0}$

□

Note that $\mathbf{g}_\epsilon^t$ are convex (since $\mathbf{g}^t$ are). In addition, defining $\mathrm{Fit}_\epsilon^T := \sum_{t=1}^{T}[\mathbf{g}_\epsilon^t]^+$, Lemma 1 states that if Fit_ϵ^T is sub-linear in T then both the outage fit and the fit are also sublinear in T. We therefore propose to use $[\mathbf{g}_\epsilon(\boldsymbol{x}^t)]^+$ as the constraint functions in the online learning problem. We define the corresponding Lagrangian $\mathcal{L}_\epsilon^t(\boldsymbol{x}^t, \boldsymbol{\lambda}^t) = f^t(\boldsymbol{x}^t) + [\mathbf{g}_\epsilon^t(\boldsymbol{x}^t)]^+\boldsymbol{\lambda}^t$, whose (sub)gradient is given by

$$\nabla_{\boldsymbol{x}}\mathcal{L}_\epsilon^t(\boldsymbol{x}^t, \boldsymbol{\lambda}^t) = \nabla_{\boldsymbol{x}} f^t(\boldsymbol{x}^t) + \sum_i \lambda_i^t \nabla_{\boldsymbol{x}}[g_{i,\epsilon}^t(\boldsymbol{x}^t)]^+ \quad (4)$$

We then update the primal and dual variables as

$$\boldsymbol{x}^{t+1} = \mathcal{P}_\Omega\Big(\boldsymbol{x}^t - \alpha\nabla_{\boldsymbol{x}}\mathcal{L}_\epsilon^t(\boldsymbol{x}^t, \boldsymbol{\lambda}^t)\Big), \quad (5)$$

$$\boldsymbol{\lambda}^{t+1} = \frac{[\big(\mathbf{g}_\epsilon^t(\boldsymbol{x}^{t+1})\big]^+}{\alpha\sigma}, \quad (6)$$

where σ is a parameter chosen such that $\sigma > \frac{MG^2}{\epsilon^2}$. The procedure is summarized in Algorithm 1.

Algorithm 1 Online learning algorithm with no regret and no outages

1: **Inputs:** Parameters ϵ, σ and step size α.
2: Initialize primal iterate $\boldsymbol{x}^1$ and dual iterates $\boldsymbol{\lambda}^1$.
3: **for** $t = 1, \cdots, T$ **do**
4: Take resource allocation decision $\boldsymbol{x}^t$.
5: The loss $f^t(\boldsymbol{x}^t)$ and the constraint function $\mathbf{g}^t(\boldsymbol{x}^t)$ are revealed.
6: Update the primal variables $\boldsymbol{x}^{t+1}$ according to (5)
7: Update the dual variables $\boldsymbol{\lambda}^{t+1}$ according to (6)
8: **end for**

In the rest of this Section, we provide the bounds for the regret and the fit under Alg. 1. First, we can show

that if the incoming computation demands are bounded the standard assumptions in the OCO literature, hold:

Lemma 2. *Assume the* $\boldsymbol{d}^t \leq d^{\max} < \infty$. *Then:*

1. *For every t, the functions* $f^t(\boldsymbol{x})$ *and* $\boldsymbol{g}^t(\boldsymbol{x})$ *are convex and Lipschitz continuous.*
2. *Function* $f^t(\boldsymbol{x})$ *is bounded over the set* Ω*, and has bounded gradients meaning* $|f^t(\boldsymbol{x})| \leq F$, $\|\nabla f^t(\boldsymbol{x})\| \leq G$ *and* $\max_n \|\nabla g_n^t(\boldsymbol{x})\| \leq G$
3. *The radius of the convex set* Ω *is bounded; i.e.,* $\|\boldsymbol{x} - \boldsymbol{y}\| \leq R, \forall \boldsymbol{x}, \boldsymbol{y} \in \Omega$.

In addition, to provide bounds for the outage fit, we need an additional assumption, which is essentially a Slater condition on the constraints:

Assumption 1. *Define* $\mathcal{E} := \{x \in \Omega$ *such that* $g_i^t(x) < -\epsilon \ \forall i = 1, .., M \ \forall t = 1..T\}$. *Then, given* T *and* ϵ, $\mathcal{E}$ *is not empty.*

The performance of Algorithm 1 can be then characterized as follows (proof omitted due to lack of space):

Theorem 1. *Suppose that the traffic is bounded, assumption A1 is satisfied and that resource allocation decisions are made according to Algorithm 1 with* $\beta_\epsilon = (1 - M\frac{(G+\epsilon)^2}{\epsilon^2\sigma})$. *Then the dynamic regret is bounded as:*

$$\mathrm{Reg}_d^T \leq \epsilon MCT + \frac{RV(\boldsymbol{x}_*^{1:T})}{\alpha} + \frac{R^2}{2\alpha} + \alpha\frac{(G+\epsilon)^2}{\epsilon^2}T,$$

where $V(\boldsymbol{x}_*^{1:T}) = \frac{1}{2}\sum_{t=1}^{T} \left\|\boldsymbol{x}_*^t - \boldsymbol{x}_*^{t-1}\right\|$ *is the accumulated variation of the per-slot minimizers in the definition of dynamic regret* (1). *In addition the outage fit and the fit are bounded by:*

$$\mathrm{OutageFit}^T \leq M\frac{T^{1/2}}{\beta_\epsilon}\sqrt{2\alpha\sigma FT + \frac{R^2}{2}\sigma + \alpha^2\sigma\frac{(G+\epsilon)^2}{\epsilon^2}T}$$

$$\mathrm{Fit}^T \leq \epsilon M\frac{T^{1/2}}{\beta_\epsilon}\sqrt{2\alpha\sigma FT + \frac{R^2}{2}\sigma + \alpha^2\sigma\frac{(G+\epsilon)^2}{\epsilon^2}T}$$

Furthermore, if we choose $\alpha = \mathcal{O}(T^{-3/4})$, $\epsilon = \mathcal{O}(T^{-1/4})$ *and* $\sigma = \mathcal{O}(T^{1/2})$, *then by applying Algorithm 1 we obtain:*

$$\boxed{\mathrm{Reg}_d^T = \mathcal{O}(\max\{T^{3/4}, T^{3/4}V(\boldsymbol{x}_*^{1:T})\})} \quad \boxed{\mathrm{Reg}_s^T = \mathcal{O}(T^{3/4})}$$

$$\boxed{\mathrm{OutageFit}^T = \mathcal{O}(T^{7/8})} \quad \boxed{\mathrm{Fit}^T = \mathcal{O}(T^{5/8})}$$

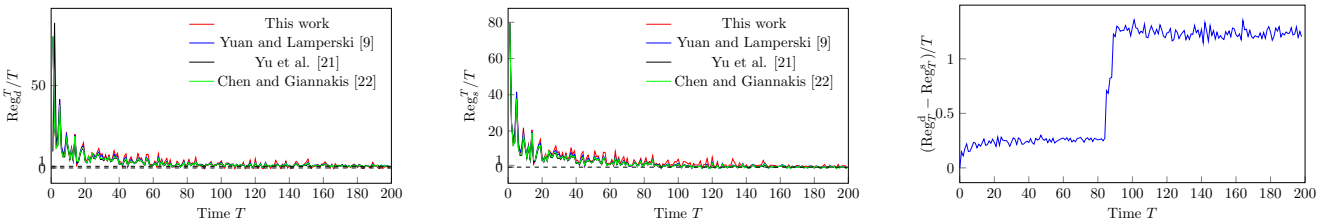
An online primal-dual algorithm with $[\mathbf{g}^t(\boldsymbol{x}^t)]^+$ in the Lagrangian (therefore no guarantees on service outages) would have regret and constraint fit (2) in the order or $\mathcal{O}(\sqrt{T})$ and $\mathcal{O}(T^{3/4})$, respectively. Therefore, introducing the outage fit constraint makes the regret of our proposed algorithm scale worse with T, while the fit scales better. This is natural since the constraint here is more severe than (2). In addition, note that in order to obtain a sub-linear regret, ϵ must be set as a function of T. For long horizons T this would mean that ϵ would be small, so, in practice, Assumption 1 would be satisfied.

V. Numerical experiments

In this section we test the performances of the proposed algorithm based on a real dataset published as Open Data by Telecom Italia in 2014 [15]: The area of Milan was composed of a grid overlay of 10000 squares. For each square, the dataset contains mobile users' activities such as Internet data connectivity from November 1st, 2013 to January 1st, 2014 [16]. The dataset was created with a temporal aggregation of time slots of ten minutes. We take 2 squares as two locations. We generate from this dataset the demand in each location at each time slot, as shown in Fig. 4(a). We suppose that there are 2 BSs and 2 ESs. The delay of the transmission path from BS b to the cloud ($d_{b,c}^t$) is chosen randomly between 1 and 5 ($d_{n,c}^t$ is chosen similarly). The parameters that capture interference and noise ($\alpha_{l,b}^t$ and $\alpha_{b,n}^t$) are chosen randomly between 1 and 10 at each time slot. We fix $b_w = 2$. The optimal decisions are obtained using CVXPY [17]. We compare Algorithm 1 with three algorithms from the literature: the algorithm proposed by Yuan and Lamperski [9] extended for time varying constraints, the algorithm proposed by Yu et al. [18] and the saddle-point method proposed by Chen and Giannakis [19] modified to use full gradient knowledge instead of bandit feedback.

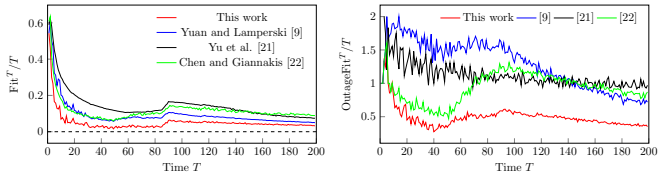
Figs. 2(a) and 2(b) show the dynamic and the static regret divided by T. The static regret is sub-linear since $\lim_{T\to\infty} \frac{\mathrm{Reg}_s^T}{T} = 0$ as shown in Fig. 2(b). For this setting, the dynamic regret is linear as we have $\lim_{T\to\infty} \frac{\mathrm{Reg}_d^T}{T} \approx 1.1$, this is because the dynamic regret is proportional to the accumulated variation $\mathbf{V}(\boldsymbol{x}_{1:T}^*)$ presented in Fig. 4(b), and since $\mathbf{V}(\boldsymbol{x}_{1:T}^*)$ is linear then the dynamic regret is also linear. The difference between the static and dynamic regret is shown in Fig. 2(c). That difference is equal to the cost function of the static benchmark minus the cost function of the dynamic benchmark. After $T = 85$, the difference becomes more important as we have a peak in demand at $T = 85$ so the static benchmark will generate a very high cost function after that point. Figure 3 shows the fit and the outage fit. We can observe that Algorithm 1 outperforms the other algorithms in terms of fit and outage fit, as well as that both metrics are sub-linear except for the time interval $[85 - 90]$ for the fit and $[40 - 90]$ for the outage fit. This can be explained as follows: when taking the action $\boldsymbol{x}^t$ at time slot t, the algorithm does not know neither d_l^t nor the cost and constrains functions; and has to estimate the optimal decision $\boldsymbol{x}_*^t$ based on the history $d^1..d^{t-1}$, $f^1..f^{t-1}$ and $\mathbf{g}^1..\mathbf{g}^{t-1}$. This may not be helpful when the demands and those functions deviate from the past at constant speed at least. Therefore an online algorithm cannot track well the benchmark sequence.

Compared to the baselines, Algorithm 1 may slightly increase the regret but it greatly ameliorates constraint violations. Specifically, we can see from Fig. 3(b) that the outage fit is almost half while the regret remains very close. The numerical results illustrate that the proposed



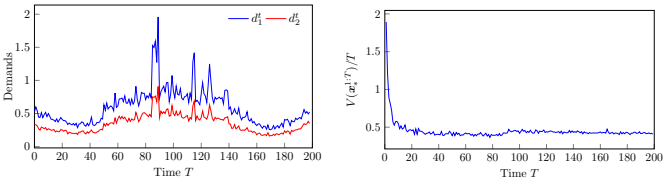
(a) Dynamic regret (b) Static regret (c) Comparison between static and dynamic regret of Algorithm 1

Figure 2: Simulation results for the regret of different algorithms.



(a) Cumulative constraint fit. (b) Cumulative number of outages.

Figure 3: Simulation results on constraint violations:



(a) Demands for each location. (b) Variation of the benchmark used for the dynamic regret.

Figure 4: Data traces and optimal solution.

algorithm can indeed significantly reduce the number of constraint violations and, consequently, service outages while having only a small impact on the cost.

VI. Conclusion

In this paper we have formulated the edge resource reservation problem as a constrained convex optimization problem with time-varying cost and constraint functions. We have introduced a new metric, called outage fit, in order to prevent service outages due to allocating insufficient resources and proposed an online algorithm for which we analytically show that it guarantees to learn a to take resource allocation decisions that cause no service outages and also have sublinear regret. Numerical results based on real data traces illustrate the improvement that results from our algorithm in terms of service outages and constraint satisfaction. As a future work, we will focus on bandit convex optimization where the information available are only the values of the cost and constraints functions for the resource allocation that was selected.

References

[1] N. Zhang *et al.*, "Dynamic spectrum access in multi-channel cognitive radio networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 11, pp. 2053–2064, 2014.

[2] H. Ji *et al.*, "Ultra-reliable and low-latency communications in 5g downlink: Physical layer aspects," *IEEE Wireless Communications*, vol. 25, 2018.

[3] Y. C. Hu *et al.*, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[4] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and trends in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2011.

[5] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *(ICML-03)*, 2003, pp. 928–936.

[6] A. Jadbabaie *et al.*, "Online optimization: Competing with dynamic comparators," in *Artificial Intelligence and Statistics*. PMLR, 2015, pp. 398–406.

[7] E. C. Hall and R. M. Willett, "Online convex optimization in dynamic environments," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 647–662, 2015.

[8] M. J. Neely and H. Yu, "Online convex optimization with time-varying constraints," *arXiv preprint arXiv:1702.04783*, 2017.

[9] J. Yuan and A. Lamperski, "Online convex optimization for cumulative constraints," in *NIPS*, 2018.

[10] T. Chen, Q. Ling, and G. B. Giannakis, "An online convex optimization approach to proactive network resource allocation," *IEEE Transactions on Signal Processing*, 2017.

[11] T. Chen *et al.*, "Heterogeneous online learning for "thing-adaptive" fog computing in iot," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4328–4341, 2018.

[12] N. Liakopoulos, G. Paschos, and T. Spyropoulos, "No regret in cloud resources reservation with violation guarantees," in *IEEE INFOCOM*, 2019.

[13] B. Donassolo *et al.*, "Online reconfiguration of iot applications in the fog: The information-coordination trade-off," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 5, pp. 1156–1172, 2022.

[14] R. Deng *et al.*, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, 2016.

[15] G. Barlacchi *et al.*, "A multi-source dataset of urban life in the city of milan and the province of trentino," *Scientific data*, vol. 2, no. 1, pp. 1–15, 2015.

[16] T. Italia, "Telecommunications - SMS, Call, Internet - MI," 2015. [Online]. Available: https://doi.org/10.7910/DVN/EGZHFV

[17] 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.

[18] H. Yu, M. J. Neely, and X. Wei, "Online convex optimization with stochastic constraints," in *NIPS*, 2017.

[19] T. Chen and G. B. Giannakis, "Bandit convex optimization for scalable and dynamic iot management," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1276–1286, Feb. 2019.