Online Learning for Industrial IoT: The Online Convex Optimization Perspective

Livia Elena Chatzieleftheriou*, Chen-Feng Liu†, Iordanis Koutsopoulos‡, Mehdi Bennis§, Mérouane Debbah†
*IMDEA Networks Institute, Madrid, Spain
†Technology Innovation Institute, Masdar City, Abu Dhabi, United Arab Emirates
‡Department of Informatics, Athens University of Economics and Business, Athens, Greece
§Centre for Wireless Communications, University of Oulu, Oulu, Finland
livia.chatzieleftheriou@imdea.org, {chen-feng.liu, merouane.debbah}@tii.ae, jordan@aueb.gr, mehdi.bennis@oulu.fi

2022 IEEE International Mediterranean Conference on Communications and Networking (MediTCom) | 978-1-6654-9825-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/MEDITCOM55741.2022.9928703

***Abstract*—Industrial Internet of things (IIoT), one enabler for Industry 4.0 Smart Factories, is a mission-critical and latency-sensitive application of 5G networks. Due to the stringent latency requirements in IIoT, coordinating the simultaneous transmissions of massive entities and knowing the interference they create to each other is not feasible. Additionally, due to the mobility feature of mobile robots and automated guided vehicles, the experienced channel fading may differ from the estimated one. Therefore, some uncertainties exist in IIoT networks while we decide the communication and control mechanisms. Within the context of IIoT, this paper discusses some resource allocation solutions from the perspective of Online Convex Optimization (OCO). OCO is a computationally lightweight and memory-efficient mathematical tool which tackles the optimization problems, given that the network environment is arbitrary and unknown. We first introduce the key performance indicators in IIoT networks and highlight the uncertain factors, which we may encounter while allocating the communication resources in IIoT. Then we provide an overview of main principles of OCO and present the comparison benchmarks and related metrics for performance evaluation. Moreover, we discuss the kind of resource allocation problems in IIoT that can be tackled by OCO. Finally, we summarize the advantages of applying OCO to IIoT networks.**

***Index Terms*—5G and Beyond, Industrial Internet of Things (IIoT), Online Learning, Online Convex Optimization (OCO).**

I. Introduction

Through connecting physical objects via the network, i.e., Internet of things (IoT), and by embedding them with intelligence, cyber-physical systems can operate and interact efficiently and proactively to provide smart services. Among the services, smart factories cooperatively and efficiently manufacture products as envisioned in Industry 4.0 [1]. In smart factories, factory equipments such as sensors, mobile robots, and automated guided vehicles (AGVs) are (preferably, due to the flexibility and agility) wirelessly connected through industrial IoT (IIoT). In contrast to the communication in IoT for improving human awareness of the surrounding environment, the machine-centric IIoT has more stringent communication performance requirements [2]. In this regard, some works such as [2], [3] have specified the features, directions, and challenges in IIoT. However, the useful methodologies for solving the communication problems in the generic viewpoint were missed therein, albeit specific mathematical approaches have been used in particular IIoT works [4], [5]. Among the problem-tackling approaches, the online ones, e.g., Reinforcement Learning (RL) [4] and Lyapunov Optimization [5], are powerful to deal with the dynamics in wireless environments, providing adaptive solutions. However, RL requires abundant data to learn the policy. Given that the sensors are enabled with autonomy, RL can deplete the sensor's limited memory. Moreover, as the mobility feature is prominent in IIoT (due to mobile robots and AGVs), the originally-observed channel value may change when the transmitter starts transmission [6]. This variation will degrade the solution performance of RL and Lyapunov Optimization.

Motivated by the aforementioned issues, in this paper, we consider the *Online Convex Optimization* (OCO) technique in online learning as a possible candidate to address the communication problems of IIoT from a learning perspective. Briefly speaking, OCO is an online learning technique which does not need to collect a large amount of data either online or offline. Instead, it operates on the fly and progressively learns the action policy with only the latest-received feedback in response to the action. As stated by the full form of OCO, the communication policy is learned in an online manner. In the online framework, the communication action is decided based on the currently available (partial) information about the network status. As the available information becomes more complete over time, the communication policy evolves and converges.

Among the online approaches, OCO is a powerful one for solving the problems, e.g., resource allocation, computation offloading, and user association, without knowing the current network status [7]–[10]. More specifically, relying only on the most recent historical values of the system parameters, the OCO framework decides the action for the current time instant before the actual parameter values are revealed and incorporated with the action to obtain a cost or utility. Since historical data are no longer required, this procedure makes OCO memory-efficient.

Furthermore, OCO provides solutions with performance guarantees. That is, the achieved performance differs by, at most, a provable bound from the performance of the optimal solutions which know the system parameter evolution in hindsight. The performance difference can be measured by

The work of Livia Elena Chatzieleftheriou was done while she was affiliated with the Athens University of Economics and Business, Greece.

regret, a metric quantifying the cost of gradually learning in OCO. In addition, the aggregate constraint violation of the OCO solution is measured by *fit*. We will further explain these two metrics later in the paper. The content of this paper is outlined as follows:

- We explain the key performance indicators (KPIs) of IIoT and highlight some uncertain factors which may be encountered in IIoT.
- We present a primer of OCO and introduce a widely-used algorithm in OCO, which is called *Online Mirror Descent* (OMD). We then explain how the historical information is leveraged for iterative action updates.
- We define the static and dynamic benchmarks against which OCO solutions are commonly compared. Further, we give the definitions of two metrics, regret and fit, which are measured for evaluating the solution performance. Regret captures the aggregate (over the time horizon) difference of the costs accumulated by an online algorithm and an offline benchmark that takes decisions in hindsight. Fit captures the aggregate (over the time horizon) constraint violation of the online decisions.
- We connect the OCO framework and IIoT problems, and summarize the advantages of OCO.

II. Key Performance Indicators and Uncertainties in IIoT

A. Key Performance Indicators

The KPIs in IIoT networks include [2], [11]:

1) Reliability;
2) Latency;
3) Determinism;
4) Availability;
5) Scalability;
6) Battery lifetime;

We now elaborate on them and present them in detail.

1) Reliability and Latency: Since factory environments vary dynamically, ineffectively tracking or controlling plants may incur catastrophic issues. Hence, information and instructions in IIoT need to be timely delivered. Due to the intrinsic randomness in wireless networks resulting in non-negligible information delivery delays and errors, guaranteeing the *reliability* and *latency* of information delivery is of paramount importance. To this end, ultra-reliable low-latency communication (also known as URLLC) has been considered as one enabler for IIoT [12].

2) Determinism and Availability: In industrial automation, wireless sensors sample the status data of the monitored environments and then send them to the controller of control systems. After manipulating the received information, the controller subsequently issues control commands to actuators. In this procedure, sensors and actuators are required to carry out specific actions at precise instants, i.e., *determinism* [3] exists. Since manufacturing and processing operations are uninterrupted in factories, wireless service should be available on demand and instantly. Quantitatively, the *availability* of communication service is measured by the percentage of time during which the system requirements are guaranteed [11]. The determinism and availability performances of IIoT will affect factory operations.

3) Scalability and Battery Lifetime: Cisco envisioned that there will be 14.6 billion machines in the Internet by 2022 [13]. As per the white paper [11] of the 5G Alliance for Connected Industries and Automation (5G-ACIA), wireless service needs to support 10000 devices per km^2 in status monitoring. Therefore, how to scale up the network, i.e., the *scalability* issue, is crucial for IIoT. Finally, most wireless sensors are powered by batteries. It will be beneficial if the sensors can harvest renewable energy while depleting electricity. Otherwise, batteries need to be changed often, which is impractical or infeasible in the hazardous industrial environments. In this situation, prolonging the *battery lifetime* is another critical concern in IIoT.

Furthermore, 5G-ACIA has specified the targeted KPI requirements of various IIoT applications in 5G networks [11], where the stringency of the requirement is application-dependent.

B. Uncertainties

In some IIoT networks with mobile robots or AGVs, the transmitter or receiver is moving. Due to the mobility feature, the channel fading within a coherence time, in practice, does not remain static but varies [6]. In other words, even if the fading channel is perfectly estimated, the experienced fading will not be identical to the estimated one, thus leading to unpredictable communication performance. In addition, a tremendous number of IIoT entities need to share the scarce communication resources, e.g., time and bandwidth, resulting in severe co-channel interference. However, coordinating all transmissions and knowing the interference incur significant overheads, in particular, when the payload size in IIoT is typically small [11].

In control systems, the controller's available information may not correctly reflect the actual status of the concerned time-varying environment at any time instant. This inconsistency degrades the control system performance [14], resulting in the need for effective information updates from the sensors in industrial automation. Measures such as the *Age of Information* (AoI) [15], the *Value of Information* (VoI) [16], and the *Age of Incorrect Information* (AoII) [17] have been considered for proposing the information-updating mechanisms in wirelessly-networked control systems [18].

Consider a scheme in which the sensor's data-sampling mechanism is event-triggered. In this situation, an unknown queuing latency, resulting in uncertain information measures (i.e., AoI, VoI, and AoII), will be incurred to the newly-sampled data while the communication resources are allocated for uploading the previously-sampled data to the controller. Therefore, an effective and scalable IIoT transmission or resource allocation mechanism is required, in which the transmitter is not aware of the aforementioned or other uncertainties.

III. Online Convex Optimization

In this section, we firstly introduce the general procedure of OCO and give the examples connecting the OCO framework

and IIoT networks. We then explain a basic OCO algorithm, called Online Mirror Decent (OMD). Finally, we outline the main OCO benchmarks, regret, and fit.

A. Brief Introduction to Online Convex Optimization

In the OCO framework [19], [20], we consider a discrete-time horizon with T slots. Let d be a positive integer capturing the number of decisions or, equivalently, the dimension of the problem. For example, in the IIoT context d can capture the number of robots in a factory. Before each slot $t \in \{1, \cdots, T\}$, we decide the d-tuple action vector $\mathbf{x}(t) = \big(x_1(t), \cdots, x_d(t)\big) \in \mathcal{A} \subseteq \mathbb{R}^d$ for slot t without knowing the random state vector $\boldsymbol{\lambda}(t)$ (experienced in slot t) of the environment, where $\mathcal{A}$ is the convex feasible set of actions. Additionally, $\mathbf{x}(t)$ and $\boldsymbol{\lambda}(t)$ are fixed within the duration of each slot t but change over slots. The random state $\boldsymbol{\lambda}(t)$ can follow any arbitrary distribution to which we are agnostic. In each slot t, we have the cost $f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big)$ which is a function of the action $\mathbf{x}(t)$ and the state $\boldsymbol{\lambda}(t)$. Given a fixed $\boldsymbol{\lambda}(t)$, the cost function is convex with respect to $\mathbf{x}(t)$. We aim to minimize the aggregate (over the time horizon) cost by choosing the action series $\{\mathbf{x}(t) : t = 1, \cdots, T\}$.

For example, in the IIoT context, a problem is to minimize the energy consumed by mobile robots in a smart factory. Robots can be used in the IIoT context to carry boxes from one part of the factory to the other. In this scenario, the unknown vector $\boldsymbol{\lambda}(t)$ can capture channel conditions, the decision vector $\mathbf{x}(t)$ can capture the frequency, i.e., the number of signals per unit of time, which the robots send to an access point within the factory to inform it about their locations, and the cost function $f(\cdot)$ captures the energy that the robots consume.

Without loss of generality, for utility maximization scenarios, we can simply consider the cost function in the OCO framework as the negative utility, i.e., the utility multiplied by -1. Some special cases of OCO consider linear costs (also called Online Linear Optimization, i.e., OLO) or strongly convex cost functions. In these cases, the proposed solutions may exploit the additional properties of the cost function $f(\cdot)$, related to linearity or strong convexity.

The solutions $\{\mathbf{x}(t) : t = 1, \cdots, T\}$ may also need to satisfy some constraints. Constraints can be either *strict* or *budget-based*. The strict constraints need to be ensured in each slot t, whereas the budget-based constraints are ensured in the time-averaged manner, i.e., over the entire time-horizon T. Similar to the cost function $f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big)$, the value of the constraint function $g_c\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big), \forall\, c \in \{1, \cdots, C\}$, depends on both the action $\mathbf{x}(t)$ and the (unknown) random state $\boldsymbol{\lambda}(t)$. Moreover, given a fixed $\boldsymbol{\lambda}(t)$, the constraint function is convex with respect to $\mathbf{x}(t)$. In practice, constraints may capture service constraints. An example of strict constraint in the IIoT context is to ensure URLLC in each time slot, while an example of budget-based constraints is to ensure that the total energy needed for a device to operate for a given time horizon does not exceed an amount in the long term that reflects its energy autonomy.

Algorithm 1 General Online Convex Optimization Procedure

Input: Step size η and cost function $f(\cdot)$.
Output: $\{\mathbf{x}(t) : t = 1, \cdots, T\}$.
1: Initialize $\mathbf{x}(1)$.
2: **for** $t = 1, \cdots, T$ **do**
3: Observe the random state realization $\boldsymbol{\lambda}(t)$ and obtain the cost value $f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big)$.
4: Calculate the gradient $\nabla_{\mathbf{x}} f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big)$ with respect to the action decision $\mathbf{x}(t)$.
5: Update the next action, i.e., decide $\mathbf{x}(t+1)$, based on the step size η, obtained cost $f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big)$, and/or gradient $\nabla_{\mathbf{x}} f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big)$.
6: **end for**

The general procedure of OCO is shown in Algorithm 1. In OCO, we start with an initial solution, $\mathbf{x}(1)$. Then, at the end of each time-slot t, we:

1) Observe the value of the unknown parameter $\boldsymbol{\lambda}(t)$,
2) Evaluate our current solution $\mathbf{x}(t)$, i.e., find the value of $f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big)$, and
3) Update our solution to $\mathbf{x}(t+1)$.

We next discuss each of the above stages.

Observation of the environment vector $\boldsymbol{\lambda}(t)$: Many network parameters, such as the channel state information of wireless systems, are highly dynamic. Although techniques for the statistical characterization of these parameters can be used, in practice, the stringent latency requirements in IIoT make this characterization impossible, because of the increased delay and resource requirements of these techniques. To overcome this lack of prior information over the values of the system's parameters, some OCO approaches consider that, when decision $\mathbf{x}(t)$ is taken, the environment vector $\boldsymbol{\lambda}(t)$ is arbitrary and unknown, or it can even be adversarial. This implies that the values of the objective function, $f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big)$, and the constraint function, $g_c\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big), \forall\, c \in \{1, \cdots, C\}$, for slot t are chosen by an adversary too. Although in reality oftentimes there is no adversary, this is an elegant framework for optimizing the systems in the worst case, obtaining solutions with performance guarantees. The actual value of vector $\boldsymbol{\lambda}(t)$ can be observed only when the time slot t is completed. For example, in the energy minimization for autonomous robots problem, the vector parameter $\boldsymbol{\lambda}(t)$ for slot t captures an average channel state during slot t. Thus, only after slot t finishes, it is possible to compute it and, thus, reveal $\boldsymbol{\lambda}(t)$.

Evaluation of the solution $\mathbf{x}(t)$: When the actual value of the environment vector $\boldsymbol{\lambda}(t)$ is revealed, the value $f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big)$ and the gradient $\nabla_{\mathbf{x}} f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big)$ of the cost function are revealed too. The gradient of the cost with respect to (w.r.t.) the decision $\mathbf{x}(t)$ informs the decision-maker about how the existing solution should be updated, in order to get a more cost-effective solution. Given the costs, the computation of the gradient function is a computationally light step, as it suffices to substitute the observed values for $\boldsymbol{\lambda}(t)$ and the decisions $\mathbf{x}(t)$ in the formula of $\nabla_{\mathbf{x}} f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big)$.

Algorithm 2 Online Mirror Descent

Input: Mirror function $h : \mathbb{R}^d \rightarrow \mathcal{A}$, step size η, and objective function $f(\cdot)$.
Output: $\{\mathbf{x}(t) : t = 1, \cdots, T\}$.
1: Initialize the auxiliary vector $\mathbf{y}(1) = \mathbf{0}$
2: **for** $t = 1, \cdots, T$ **do**
3: Decide the action $\mathbf{x}(t) = h\big(\mathbf{y}(t)\big)$.
4: Update the auxiliary vector as $\mathbf{y}(t+1) = \mathbf{y}(t) - \nabla_{\mathbf{x}} f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big)$.
5: **end for**

Update of decision to $\mathbf{x}(t+1)$: The solution updates are based on the information obtained during the solution evaluation. When the cost gradient w.r.t. the decision $\mathbf{x}(t)$ is available, the solution $\mathbf{x}(t+1)$ is obtained by making a step of size η towards a direction opposite to that of the gradient. For example, in the autonomous robot scenario, the cost may capture energy consumption. In this case, the gradient captures the change in energy consumption, when an infinitesimal change in frequency occurs. The robot communication frequency policy is updated by taking a step against the gradient, i.e., deciding a higher or lower (depending on the gradient) frequency for each robot's communication with the access point, based on the channel state that was observed during the previous slot, and the current frequency policy.

B. Online Mirror Descent

1) Online Mirror Descent Algorithm: The OMD algorithm [20] is widely used in online learning and OCO. The general steps of OMD are outlined in Algorithm 2. OMD is considered as a convenient choice for many optimization problems since it gives the opportunity to exploit the geometry of the problem's feasibility set. It leads to decision updates that lie in the feasible set without the need for projections. This will be formally defined later. Let us now give a brief discussion.

OMD decides the current action based on the previous one, following a simple gradient update rule. We define an auxiliary vector $\mathbf{y}(t)$ with the same dimension as the one of the action $\mathbf{x}(t)$. It tracks the point at which we will end up if starting from $\mathbf{x}(t-1)$ and performing a step against the gradient. The auxiliary vector $\mathbf{y}(t)$ is initialized as $\mathbf{y}(1) = \mathbf{0}$ and updated as

$$\mathbf{y}(t+1) = \mathbf{y}(t) - \nabla_{\mathbf{x}} f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big), \ \forall\, t \in \{1, \cdots, T\}. \quad (1)$$

Note that $\mathbf{y}(t)$ may be out of the feasible space. To obtain a feasible solution $\mathbf{x}(t)$ for slot t, the auxiliary vector $\mathbf{y}(t)$ is given as an input to a mirror function $h(\cdot)$. Essentially, the role of the "mirror" function is to project the current value of the auxiliary vector $\mathbf{y}(t)$ into the feasible space, in order to obtain a feasible solution $\mathbf{x}(t)$. More specifically, the action is decided as

$$\mathbf{x}(t) = h(\mathbf{y}(t)\big) := \underset{\mathbf{x}(t) \in \mathcal{A}}{\arg\min} \big\{ R\big(\mathbf{x}(t)\big) - \eta \cdot \langle \mathbf{y}(t), \mathbf{x}(t) \rangle \big\} \quad (2)$$

in which η and $R(\cdot)$ are the step size and a regularization function, respectively. Moreover, $\langle \cdot, \cdot \rangle$ denotes the inner product between two vectors. Intuitively, the inner product in (2) represents a projection of the auxiliary onto the feasible set $\mathcal{A}$ of actions. This projection essentially will find the element of the set $\mathcal{A}$ that is the closest to the point it is projecting. In our case, it is $R(\mathbf{x}) - \langle \eta \mathbf{y}, \mathbf{x} \rangle$.

2) Regularization Function: The OCO algorithm decides the action for slot t based on the realization of $\boldsymbol{\lambda}(t-1)$, i.e., without knowing the actual realization of $\boldsymbol{\lambda}(t)$ over which the action will be applied. This can have as result the oscillation of solutions between consecutive slots, due to the possibly very different values of $\boldsymbol{\lambda}(t-1)$ and $\boldsymbol{\lambda}(t)$ in slots $t-1$ and t. The regularization function aids towards stability of the decision across time and, if chosen appropriately, it leads to solutions that are asymptotically optimal in a sense that we will present in detail. Moreover, in order to avoid the projection step of (2), the regularization function can be used by exploiting the geometry of the feasible space. For example, the work [10] aimed to find solutions that lie in sets that are the combinations of unit simplices. A possible regularization function $R(\cdot)$ for each simplex in the feasible space is the Gibbs-Shannon entropy. For d the problem dimension and for action vector $\mathbf{x}(t) = \big(x_1(t), \cdots, x_d(t)\big)$, the Gibbs-Shannon entropy is defined as

$$R(\mathbf{x}(t)\big) = \sum_{n=1}^{d} x_n(t) \log x_n(t). \quad (3)$$

Using the Gibbs-Shannon entropy as the regularization function leads to the well-known *exponentiated gradient descent*.

C. Metrics and Benchmarks

Online algorithms are usually compared against *offline benchmarks*, i.e., algorithms that know the entire system evolution in hindsight. The main metrics that characterize the learning performance of online algorithms are *regret* and *fit*. Regret captures the aggregate (over the time horizon) difference of costs between the online decision and the decision taken by an offline benchmark. Fit quantifies the aggregate constraint violations over the entire time horizon. We now define the regret metric and the no-regret property. Then the two main OCO benchmarks and the respective regrets against them are specified. Finally we define the fit.

1) Regret Metric and No-Regret Property: As $\mathbf{x}(t)$ is the action taken by the online algorithm in slot t, let $\mathbf{x}^*(t)$ denote the action taken by an offline benchmark which knows the evolution of $\boldsymbol{\lambda}(t), \forall\, t \in \{1, \cdots, T\}$, in advance. Regret is defined as

$$Reg(T) := \sum_{t=1}^{T} \big[f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big) - f\big(\boldsymbol{\lambda}(t), \mathbf{x}^*(t)\big) \big], \quad (4)$$

capturing the difference between of aggregate costs between the online policy and the offline benchmark over the time horizon. A desirable property for online algorithms is to have a regret that is sublinear to the time horizon T, which is equivalent to

$$\lim_{T \rightarrow \infty} \frac{Reg(T)}{T} = 0. \quad (5)$$

This implies that $\forall\, c > 0$, $\exists\, t_0$ such that $Reg(t) < c \cdot t, \forall\, t \geq t_0$. A regret that is sublinear to the time horizon T indicates that the algorithm behaves asymptotically as the benchmark.

Equivalently, a regret that is sublinear to the time horizon implies that as time evolves, the online algorithm manages to learn policies that are on average (over the time horizon) as good as those produced by the benchmark in terms of total cost over the time horizon.

2) Dynamic Benchmark and Dynamic Regret: The dynamic benchmark *takes the optimal action* $\mathbf{x}^*(t)$ *in each slot* t *separately* [21]. Thus, the action $\mathbf{x}^*(t)$ that is taken by the dynamic benchmark at slot t is defined as

$$\mathbf{x}^*(t) = \underset{\mathbf{x}\in\mathcal{A}}{\arg\min}\, f\big(\boldsymbol{\lambda}(t), \mathbf{x}\big),\ \forall\, t = 1, \cdots, T. \tag{6}$$

That is, the dynamic benchmark takes a sequence of T actions, i.e., one for each slot t. The regret against the dynamic benchmark is called *dynamic regret*. The dynamic regret is obtained by substituting the solution $\mathbf{x}^*(t)$ of (6) into (4). Knowing the entire system evolution in hindsight, the dynamic benchmark can adapt to the changes in $\boldsymbol{\lambda}(t)$ and take a different decision in each slot t. Hence, the dynamic benchmark is the most powerful policy that can be obtained In practice, a comparison against the dynamic benchmark offers a lower bound on the performance guarantees of the online algorithms.

3) Static Benchmark and Static Regret: The static benchmark [20] *takes only one fixed action* $\mathbf{x}^*$ *over the entire time-horizon* which minimizes the aggregate cost over T. That is, $\mathbf{x}^*(t) = \mathbf{x}^*, \forall\, t = 1, \cdots, T$. Thus, the fixed action $\mathbf{x}^*$ that is taken by the static benchmark over the entire time horizon is defined as

$$\mathbf{x}^* := \underset{\mathbf{x}\in\mathcal{A}}{\arg\min} \sum_{t=1}^{T} f\big(\boldsymbol{\lambda}(t), \mathbf{x}\big). \tag{7}$$

The regret against the static benchmark is called *static regret* and is obtained by substituting the solutions $\mathbf{x}^*(t) = \mathbf{x}^*, \forall\, t = 1, \cdots, T$, of (7) into (4). The static benchmark is the most common benchmark against which the OCO solutions are compared. Despite knowing the system evolution in hindsight, the static benchmark is allowed to take only one decision over the entire time-horizon.

The work [10] introduced a novel periodic benchmark, which is particularly useful for performance comparison in the case of conjectured periodicity in $\boldsymbol{\lambda}(t)$. The benchmark is called Optimal Periodic Static (OPS). OPS partitions the time horizon T into multiple time windows and takes one fixed (i.e., static) action over each time window. Thus, OPS is more strict than the dynamic but more lenient than the static. Furthermore, since OPS generalizes the state-of-the-art static and dynamic regrets, both the dynamic and static benchmarks arise as two special cases of the OPS benchmark.

4) Fit: In OCO, solutions are required to satisfy the constraints. The learning fit captures the aggregate constraint violation over the time-horizon T. The formal definition is given by

$$Fit(T) := \max\left\{\sum_{t=1}^{T} g_c\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big), 0\right\}, \tag{8}$$

where $g_c(\cdot)$ is the constraint function. It is especially important for problems with budget-based constraints.

IV. Potential Roadmap of OCO in IIoT

In this section, we start by mapping the variables of the OCO framework to the resource allocation problems in IIoT. Then we connect the aforementioned IIoT KPIs with the main advantages of OCO techniques. Finally, we discuss the challenge, which may be encountered while tackling problems that arise within the IIoT contexts, with tools that have their roots in the OCO techniques.

A. Adoption of OCO for Resource Allocation in IIoT

Depending on the considered resource allocation problems, the action $\mathbf{x}(t)$ may represent the transmit power, bandwidth, transmission time interval, and more. Additionally, for the information updating problems regarding AoI, VoI, and AoII, the action $\mathbf{x}(t)$ can denote the time interval between successive transmissions which reflects the information update frequency. As mentioned previously in Section II-B, the unknown random state $\boldsymbol{\lambda}(t)$ can be the experienced channel fading, co-channel interference, queuing latency, or other factors in IIoT. In these cases, the actual state can be observed only after the action in slot t is executed or finished.

The cost and budget-based constraint functions include the net throughput, end-to-end latency, transmit energy, and information age/value. Moreover, the total power and bandwidth budget as well as the transmission time threshold are considered as the strict constraints. The closed-form expressions and gradients of these cost and constraints function are available for system designs.

In IIoT, we mainly aim to minimize the energy consumption, end-to-end latency, impacts of information updates, and so on. Furthermore, deriving the gradient function is computation-light, and having the values of $\boldsymbol{\lambda}(t)$, $\mathbf{x}(t)$, and $\nabla_{\mathbf{x}} f\big(\boldsymbol{\lambda}(t), \mathbf{x}(t)\big)$ suffices for action updates. Thus, the OCO framework is beneficial for several applications in Industry 4.0 and IIoT, since it provides a framework for enabling the memory-limited sensors with autonomy, without the need to store any datasets onboard.

B. Connection of IIoT KPIs and OCO Characteristics

As discussed in Section II, the key KPIs in IIoT include reliability, latency, determinism, availability, and scalability while ensuring that the limited energy autonomy of the IIoT devices is not exceeded. We now summarize the main advantages of OCO techniques and give a connection to these IIoT KPIs.

- **Blind and adaptable to unpredictable changes of the network status:** OCO is a good choice when the network status is unpredictable and/or difficult to characterize statistically, since it takes decisions relying on the information of the most recent past. This contributes towards the KPIs which are related to determinism, availability, and reliability in IIoT, since OCO manages to produce provably good solutions even under the worst-case circumstances.
- **Fast, computation-light, and memory-efficient:** OCO performs a simple update of the existing solution, which requires only a substitution of the revealed values to a

precomputed formula. This renders it simple, fast, and memory-efficient, contributing towards the requirements for ultra-low latency, scalability, and low energy consumption in IIoT.

- **Reliable:** OCO techniques usually manage to achieve the regrets which are sublinear to the time-horizon because they exploit the convexity and the geometry of the formulated optimization problems. This important property of OCO techniques implies that they manage to learn provably optimal solutions, despite not knowing the network status over which their decisions will be applied. This implies that OCO can ensure the reliability and availability requirements of IIoT.

C. Challenge

Leveraging OCO techniques enables IIoT entities with autonomy to distributedly decide on the communication and other control actions. However, the distributed mechanism entangles different IIoT entities' actions. For example, in order to further reduce the transmission latency or AoI, the sensor tries to use more resources such as power, bandwidth, or time. On the other hand, consuming more resources increases the other sensors' delays or AoI values due to the raised co-channel interference or postponed transmissions. To compensate the performance degradation, other sensors will also occupy more resources. This competition entangles the sensors' actions, resulting in the negative impacts. However, the action entanglement is not considered in OCO.

To coordinate the competition, incorporating game theory in OCO provides a promising approach, in which the game-theoretic equilibria provide an understanding of the outcome of the learning interactions of different sensors, and sensors are incentivized not to use all resources (i.e., less negative externalities on other sensors) while jointly achieving the satisfactory performance.

V. Conclusions

In this paper, we have focused on how the OCO techniques can be incorporated and used to tackle problems within the IIoT context. As argued above, OCO provides an ideal framework for addressing IIoT challenges. More prominently, OCO promises provably optimal learning performance guarantees without the need to reserve and use network resources such as computation, bandwidth, and storage. Indeed, OCO algorithms are lightweight in terms of computation and communication burden, and they do not require the storage of data onboard the edge devices. We firstly introduced the KPIs and discussed the factors that create uncertainties in IIoT. We then explained OCO as a mathematical tool, along with its benchmarks and metrics for performance comparison. We then connected OCO with IIoT problems, considering that the OCO framework can provide promising solutions for IIoT. We believe that this work opens the way for a faster practical realization of IIoT. In our future work, we will address the action entanglement issues of distributed resource allocation in IIoT networks by incorporating OCO and game theory.

Acknowledgments

This work was supported by the CHIST-ERA grant CHIST-ERA-18-SDCDN-004 (grant number T11EPA4-00056) through the General Secretariat for Research and Innovation (GSRI).

References

[1] R. Drath and A. Horch, "Industrie 4.0: Hit or hype? [industry forum]," *IEEE Ind. Electron. Mag.*, vol. 8, no. 2, pp. 56–58, Jun. 2014.

[2] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[3] S. Vitturi, C. Zunino, and T. Sauter, "Industrial communication systems and their future challenges: Next-generation Ethernet, IIoT, and 5G," *Proc. IEEE*, vol. 107, no. 6, pp. 944–961, Jun. 2019.

[4] H. Xu, J. Wu, J. Li, and X. Lin, "Deep-reinforcement-learning-based cybertwin architecture for 6G IIoT: An integrated design of control, communication, and computing," *IEEE Internet Things J.*, vol. 8, no. 22, pp. 16 337–16 348, Jul. 2021.

[5] C.-F. Liu and M. Bennis, "Taming the tail of maximal information age in wireless industrial networks," *IEEE Commun. Lett.*, vol. 23, no. 12, pp. 2442–2446, Dec. 2019.

[6] V. N. Swamy, P. Rigge, G. Ranade, B. Nikolić, and A. Sahai, "Wireless channel dynamics and robustness for ultra-reliable low-latency communications," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 4, pp. 705–720, Apr. 2019.

[7] T. Chen, Q. Ling, and G. B. Giannakis, "An online convex optimization approach to proactive network resource allocation," *IEEE Trans. Signal Process.*, vol. 65, no. 24, pp. 6350–6364, Dec. 2017.

[8] T. Chen, Q. Ling, Y. Shen, and G. B. Giannakis, "Heterogeneous online learning for "thing-adaptive" fog computing in IoT," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4328–4341, Dec. 2018.

[9] L. E. Chatzielefhteriou and I. Koutsopoulos, "Jointly learning optimal task offloading and scheduling policies for mobile edge computing," in *Proc. 20th Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt), Wrkshp on Resource Allocation and Cooperation in Wireless Networks (RAWNET)*, Sep. 2022, pp. 1–8.

[10] L. E. Chatzieleftheriou, A. Destounis, G. Paschos, and I. Koutsopoulos, "Blind optimal user association in small-cell networks," in *Proc. IEEE Int. Conf. Comput. Commun (INFOCOM)*, May 2021, pp. 1–10.

[11] 5G Alliance for Connected Industries and Automation, "White paper: 5G for connected industries and automation," 5G-ACIA, Tech. Rep., Feb. 2019, 2nd ed.

[12] B. Holfeld, D. Wieruch, T. Wirth, L. Thiele, S. A. Ashraf, J. Huschke, I. Aktas, and J. Ansari, "Wireless communication for factory automation: An opportunity for LTE and 5G systems," *IEEE Commun. Mag.*, vol. 54, no. 6, pp. 36–43, Jun. 2016.

[13] Y. Ren, R. Xie, F. R. Yu, T. Huang, and Y. Liu, "Potential identity resolution systems for the industrial Internet of things: A survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 391–430, 1st Quart. 2021.

[14] B. Chang, L. Zhang, L. Li, G. Zhao, and Z. Chen, "Optimizing resource allocation in URLLC for real-time wireless control systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8916–8927, Sep. 2019.

[15] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in vehicular networks," in *Proc. 8th Annu. IEEE Commun. Soc. Conf. Sensor, Mesh Ad Hoc Commun. Netw.*, Jun. 2011, pp. 350–358.

[16] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "Age and value of information: Non-linear age case," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2017, pp. 326–330.

[17] A. Maatouk, S. Kriouile, M. Assaad, and A. Ephremides, "The age of incorrect information: A new performance metric for status updates," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2215–2228, Oct. 2020.

[18] O. Ayan, M. Vilgelm, M. Klügel, S. Hirche, and W. Kellerer, "Age-of-information vs. value-of-information scheduling for cellular networked control systems," in *Proc. 10th ACM/IEEE Int. Conf. Cyber-Physical Syst.*, Apr. 2019, pp. 109–117.

[19] E. Hazan, "Introduction to online convex optimization," *Found. Trends Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[20] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2012.

[21] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. 20th Int. Conf. Mach. Learn.*, Aug. 2003, pp. 928–935.