Cooperative Edge Inferences With Online Learning

Li, Mengyuan; Venkatesha Prasad, R.; Iosifidis, George

DOI
10.1109/JIOT.2025.3610695

Publication date
2025

Document Version
Final published version

Published in
IEEE Internet of Things Journal

Cooperative Edge Inferences With Online Learning

Mengyuan Li, R. Venkatesha Prasad, and George Iosifidis

***Abstract*—The efficient execution of inferences at the edge is becoming increasingly critical for communication systems that are expected to provide users with fast and accurate mobile data analytics. These inference tasks are inherently latency-sensitive and computationally demanding, whereas edge nodes are limited by energy budgets and heterogeneous resources. This article studies how a set of edge nodes can collaborate in executing demanding streaming inference tasks to optimize their aggregate performance. Such collaborative task exchange schemes enable the sharing of scarce computing resources and machine learning (ML) models (which perform the inferences) and constitute a scalable approach to this intricate problem. We formulate this exchange process as an online convex optimization (OCO) problem and design a dynamic task assignment algorithm, which is proven to have optimality guarantees even when the network and service parameters (resources and task properties) are unknown and vary arbitrarily over time. The algorithm aims to maximize inference accuracy while minimizing overall task latency and energy (including for data transfers) and simultaneously ensures that collaborating nodes do not suffer imbalanced energy costs. Through a series of data-driven experiments, we quantify the cooperation benefits under different weight combinations and validate the convergence and adaptability of the proposed learning algorithm across diverse conditions, including variations of system parameters, as well as heterogeneity across nodes and tasks.**

***Index Terms*—Cooperative inferences, edge inferences, online learning, resource management.**

NOMENCLATURE

$\mathcal{I}_i$	Set of one-hop neighbors of node i, including i.
$\mathcal{I}_i^-$	Set of one-hop neighbors of node i, excluding i.
C_{ijt}	Data transfer capacity (bytes/s) of link (i, j) in slot t.
H_{it}	Processing capacity (cycles/s) of node i in slot t.
e^c_{it}	Computing energy (J/cycle) of node i in slot t.
e^x_{ijt}	Transmission energy (J/cycle) of link (i, j) in slot t.
β_{it}	Number of tasks generated by node i in slot t.
p_{it}	Computing load (cycles) of tasks of i in slot t.
s_{it}	Data load (bytes) of tasks of node i in slot t.
a_{it}	Compute accuracy for tasks of i at node j in slot t.
x_{ijt}	Portion for tasks of node i sent to j in slot t.
u_t	Network utility accuracy for tasks processing in t.
v_t	Network energy cost for tasks processing in slot t.
d_t	Network delay for tasks processing in slot t.
γ_1, γ_2, and γ_3	Priority parameters of the EC.
$\boldsymbol{x}_t$	Task assignment applied during slot t.
$\boldsymbol{z}_t$	Prescient decision in t (for internal calculations).
$\boldsymbol{\lambda}_t$	Dual variables for relaxed constraints, $\boldsymbol{g}_t(\boldsymbol{x})$, in t.
$r_t(\boldsymbol{x})$	Primal entropic regularizer in slot t.
$q_t(\boldsymbol{\lambda})$	Dual quadratic regularizer in slot t.
σ_t	Learning rate for primal variables in t.
ϕ_t	Learning rate for dual variables in t.
δ_t	Learning parameter for dual update in slot t.
$\theta \in [0, 1)$	Constraint-utility tuning parameter.

Received 5 May 2025; revised 18 July 2025 and 26 August 2025; accepted 3 September 2025. Date of publication 16 September 2025; date of current version 7 November 2025. This work was supported in part by the National Growth Fund through the Dutch 6G Flagship Project "Future Network Services" and European Commission under Grant 101017109 (DAEMON), Grant 101139270 (ORIGAMI) and Grant 101192462 (FLECON-6G). An earlier version of this paper was presented at the IEEE Globecom 2024 [DOI: 10.1109/GLOBECOM52923.2024.10901439]. *(Corresponding author: George Iosifidis.)*

The authors are with the Department of Software Technology, Delft University of Technology, 2600 AA Delft, The Netherlands (e-mail: G.Iosifidis@tudelft.nl).

Digital Object Identifier 10.1109/JIOT.2025.3610695

I. INTRODUCTION

THE vision for 6G [1] and the Next Generation of the Internet of Things (NGIoT) [2] predict that robots, wearables, and other small devices will be connected with the backbone Internet and with each other to offer new intelligent services such as mobile data analytics. These services require the collection and analysis of data streams toward providing real-time inferences to users based on smaller or larger machine learning (ML) models that the nodes host. Examples of such services include safety and security applications where cameras and other types of sensors monitor an area, or robots that analyze data streams to make real-time decisions for their mission, and wearables that infer their carrier's health condition in real-time and issue advisory notifications (see [3], [4]). Despite their promise, today's networks do not have provisions or the capacity to support such services at scale and in a resource-sustainable fashion.

A. Background and Motivation

Unlike typical communication services that involve data transfers, mobile data analytics and streaming inferences also

require the processing of the collected data with specialized ML libraries. This processing is demanding in terms of computing and energy costs, and requires suitable ML models, which, in turn, presume proper offline training with representative datasets, as well as enough storage capacity at the hosting nodes. Unfortunately, however, the extreme-edge devices that are more common in NGIoT networks have limited hardware capabilities (small form-factor) and operate with energy constraints. This impacts their capability to execute independently these services with the desired accuracy and speed [5].

The main solution that is promoted for such demanding services is, naturally, to offload them to the cloud or cloudlets. There is a vast literature on such *hierarchical architectures* studying offloading strategies such as the selection of the cloud server or the tasks that need to be offloaded, the transmission policies or the data preprocessing, and so on (see Section II). However, such solutions require transmitting voluminous data flows to servers, which is costly in terms of communication bandwidth and energy, often induces delays that can be intolerable for latency-sensitive inferences [6], and inevitably does not scale since the stronger nodes are expensive and hence not abundant.

A significantly less explored yet, we argue, more promising approach is to execute these inference tasks collaboratively at the edge by leveraging the dormant resources and complementary ML libraries of nearby nodes. For instance, nodes that do not have enough computation power can outsource part of their inference tasks to nearby nodes that are idle; nodes that do not have a properly trained ML model for their tasks can seek support from a neighbor with a better-matching library. Given the scale, diversity, and communication capabilities of these devices, such collaborative schemes can support both the accurate and fast execution of these services, without overloading the backbone network and cloud servers. Indeed, there has recently been an increased interest, and shift of focus, from hierarchical to collaborative (or, flat) inference solutions [7].

Cooperative approaches to communication problems are not new (see Section II), yet they raise fresh challenges in the case of inference services. Namely, in order to devise an effective collaborative inference execution plan (which node should execute which task), one has to take into account that: 1) the nodes are heterogeneous in terms of communication and computing resources, which complicates the identification of optimal collaboration plans; 2) the nodes have different energy consumption profiles, which bounds their data exchange and processing capabilities; 3) not all nodes can execute accurately all tasks, as this depends on the ML models that they possess [8]; 4) there are different and possibly conflicting performance criteria, such as the task accuracy (find the best ML library for each task), task execution latency, and energy costs; and 5) most of the above parameters are time-varying and *unknown* when the task assignment plans needs to be devised [9], [10]. For instance, the accuracy of a classification task cannot be predicted as it depends on how well the input data match the training data, and the resource availability of each node is not a priori known, as it depends on volatile network conditions and its usage by other services.

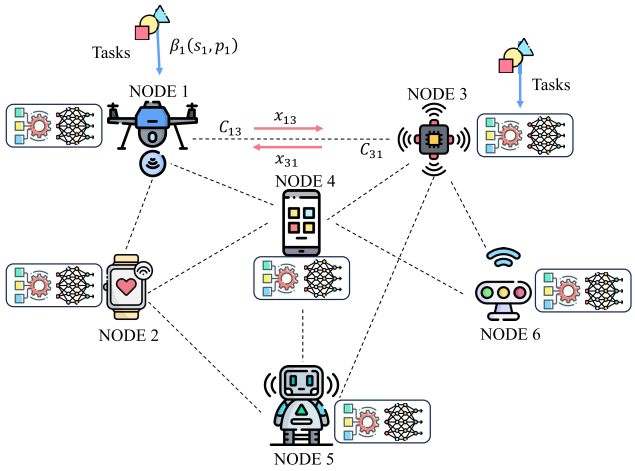


Fig. 1. IoT network model represented by a directed graph $\mathcal{G}(\mathcal{I}, \mathcal{E})$, where different nodes generate different types of tasks and execute them locally or outsource to nearby nodes.

It becomes clear from the above that collaborative edge inferences are a promising idea but require overcoming many obstacles. The goal of this article is to formalize this problem and develop a framework that allows devising dynamic task assignment policies with rigorous online learning techniques.

It is important to note that the proposed framework is applicable in scenarios where privacy concerns are either absent or handled through orthogonal technical solutions—a reasonable assumption in many practical deployments. Cases falling in the former category include systems where the devices are under the same administrative domain (hence, there is an established trusted environment), or they belong to the same user. Cases in the latter category include applications where the users share data but not labels (e.g., such as the parallel split learning paradigm [11]) or some other sensitive information or when there is some safe sandboxing environment in place. That said, privacy remains a critical consideration for broader adoption, and integrating privacy-preserving mechanisms—such as secure multiparty computation or federated learning variants—constitutes an important direction for future work.

B. Methodology and Contributions

We consider a network of nodes and one gateway, or edge controller (EC), which orchestrates the interaction of nodes over a sequence of time slots (see Fig. 1). The heterogeneity of nodes may manifest in their computing capabilities, energy consumption, type and rate of generated tasks, and their inference capabilities. The nodes are connected to each other through links with potentially different capacities over Wi-Fi or some other similar technology (e.g., ZigBee). The EC has a global view of the network and can communicate with all nodes, so as to disseminate the task assignment and collaboration policy. Its goal is to assign the inference tasks to nodes, so as to maximize their aggregate accuracy and minimize the execution delay and consumed energy, while balancing the energy that the different nodes consume due to their participation in this collaboration scheme. This

latter requirement serves as a minimal-intervention fairness criterion to avoid heavily imbalanced assignments in terms of energy—a resource that is very critical at the extreme edge.

The type and volume of the generated tasks, the node resources and task-processing accuracy values, and the link capacities may all vary over time in a nontrivial fashion, possibly following some nonstationary process with unknown statistics (mean values and so on). In fact, we use an adversarial perturbation model where we allow these quantities to be selected even by an adversary that aims to disrupt the system operation [12]. This captures the more general and demanding scenarios (e.g., operating in the presence of a jammer) that one can encounter in practice. In addition, we assume that the parameter values become available only after the tasks are executed by the nodes at each round. Therefore, the EC needs to devise the task assignment policy while being oblivious to the values of these parameters in each decision round.

To solve this dynamic task assignment problem, we leverage the theory of online convex optimization (OCO) [13] to design an algorithm with optimality guarantees (known also as regret bounds) that hold even under adversarial perturbation models. OCO has been particularly successful in tackling many key problems in communication networks, ranging from online caching [14] and IoT link scheduling [15], [16] to large-scale flow routing [17], among many others. Besides their robustness, OCO algorithms provide a full characterization of how their performance depends on the various system parameters and dimensions, thus offering a transparent and interpretable solution. Finally, as we will show, they are scalable and with minimal overheads and, hence, are suitable for the considered problem.

The algorithmic framework that we design observes past values of the system parameters (nodes, networks, and tasks), devises the assignment for the next slot, then observes the new performance and parameter values, and updates its assignment accordingly. In doing so, the algorithm ensures that the aggregate (over time and number of nodes) performance approaches that of a hypothetical desirable (but unknown in practice) benchmark assignment that the EC would have devised if it had access to the future parameter values. Technically, this means that its average regret, denoted R_T, diminishes with the collaboration time period T, i.e., $\lim_{T\to\infty} R_T/T = 0$, and the same holds for the energy balance constraint violation V_T. Hence, we are guaranteeing that for any possible scenario, the dynamic task execution plan approaches the ideal one.

To evaluate the proposed framework, we employ a representative data analytic service, namely, object identification in video frames, and use real-world measurements from our testbed that connects Raspberry Pi nodes over Wi-Fi to create large-scale synthetic experiments. We measure the benefits of collaboration and quantify how the diversity, in terms of hardware profile or task properties, impacts the results; we demonstrate the learning performance (convergence of R_T and V_T) of the proposed dynamic algorithm.

In summary, this article addresses an important gap in the literature, focusing on inferences (instead of training), and providing a rigorous and interpretable task execution mechanism with guaranteed performance. The main contributions of this work can be, therefore, summarized as follows.

1) We formulate the problem of collaborative execution of streaming inference tasks toward maximizing their accuracy, reducing latency and energy costs, while adhering to energy fairness constraints. Unlike prior works that rely on hierarchical architectures, here, we promote a flat, or cooperative, inference architecture.
2) We design an online learning algorithm, which is oblivious to tasks, node resources, and network conditions and offers optimality guarantees with respect to an ideal benchmark, even when the parameters of the system change adversarially. To the best of our knowledge, this is the first solution of its kind for this problem.
3) We evaluate the algorithm using a typical inference application and real datasets. We quantify the benefits of collaboration in scenarios of increasing node and task heterogeneity, under different perturbation models, and for different priorities of the metrics of interest (accuracy, energy, and delay). We also measure the regret convergence and constraint violation for networks of different sizes and showcase the robustness of the algorithm.

1) Article Organization: The rest of this article is organized as follows. Section II discusses the related work and background, and explains how our proposal is positioned with respect to them. Section III introduces the system model, and Section IV introduces formally the problem and the learning objectives. Section V designs the learning policy for the collaborative inference mechanism and characterizes its performance guarantees. We provide a data-driven numerical evaluation in Section VI and conclude our study in Section VII. The missing proofs are deferred to the Appendix.

II. Related Work

Our work proposes a collaborative inference solution, which is adaptive and provides performance, delay, and energy consumption guarantees, while ensuring balanced energy costs across the nodes. This proposal departs from the prior works that mainly consider hierarchical offloading architectures.

A. Inference Services at the Edge

Inference tasks in IoT networks involve the collection and analysis of data generated by small devices to support intelligent decision-making. Examples include real-time video analytics, autonomous driving, smart manufacturing, and so on (see [18]). For example, [19] implements a framework for low-latency high-accuracy recognition tasks, [20] introduces containerized intelligence services for wearables and proposes policies to reduce their response time, and [21] explores deep-learning (DL) models for space monitoring. This is a fast-growing area, as more and more communication services utilize inferences; thus, it is imperative to optimize their operation. To date, the vast majority of such inference services rely on hierarchical offloading architectures (e.g., [22] and references therein), which, however, presume the existence of stronger models and, inevitably, do not scale. Here, instead, we focus on collaborative inferences directly among the nodes.

B. Collaborative Inferences at the Edge

Indeed, there is lately increasing interest in collaborative solutions for such inference services [23]. For instance, [21] considers a collaborative system for vision applications that harnesses the computational power of multiple devices, while [24] studies collaborative inferences of LLMs across the devices. Similarly, [25] proposes an edge computing framework to enhance human detection accuracy within the video task's deadline by leveraging the computing capacity of multiple nodes, [26] consideres a general edge computing problem and proposes an algorithm for minimizing task delay and energy, [27] designes a framework where the nodes outsource to each other their tasks in order to optimize the average execution delay and accuracy, and [19] introduces an auction framework to incentivize the participation of the devices. Finally, [28] adopts knowledge distillation for cooperative DNN inferences to maximize the inference accuracy, catering to the delay and memory limitations of the devices.

Similarly, [29] proposes a system that orchestrates energy-efficient cooperative DNN inferences over heterogeneous edge devices by deciding how to partition the DNNs; [30] considers a network of devices with different DNNs and decides how to route their inference tasks to the best models. The work [31] proposes a greedy task allocation algorithm that minimizes the total execution time for multiple tasks across a group of edge devices, effectively reducing the overall execution cost. Xie et al. [32] propose the task exchange among nodes, so as to minimize the energy consumption while respecting latency constraints. The problem is formulated as a Markov decision process and solved using deep reinforcement learning. On the other hand, [33] and [34] consider a cooperative inference scenario at the edge, but the focus is on how inference ensembling can increase the accuracy.

Our work follows these important works and studies a fully dynamic collaborative solution where the tasks and resources can vary dynamically; hence, their collaboration plans need to be adaptive and achieve this goal through a principled solution that offers optimality guarantees.

C. Resource Allocation for Inference Tasks

Recent advancements in compact inference models have facilitated the deployment of inference services at the edge. However, the resource-constrained nature of edge devices introduces new challenges, particularly when these services are subject to tight deadlines. In this context, [35] proposes a genetic algorithm to address the NP-hard problem of assigning inference models to tasks or offloading them to edge servers under time and energy constraints, [25] proposes an edge computing accuracy-maximizing solution for delay-sensitive video tasks, [36] designs a safe learning algorithm for increasing accuracy while respecting delay constraints, and [37] investigates additionally the server selection aspect. Focusing particularly on DL models, [38] studies how to create effective DL model ensembles at edge servers, [39] studies offloading strategies with the aim to improve DL accuracy, and [40] and [41] optimize offloading from devices to servers and collaboration across servers to maximize DL accuracy. These works consider architectures where the task-processing nodes (typically, edge servers) are more powerful and distinct from the task-generating nodes.

A different thread focuses on extreme-edge devices that create and execute tasks alike. Pu et al. [42] and Chen et al. [43] propose device cooperation models for energy-efficient task execution, while [44] and [45] design systems for cooperative stream processing, and [46] studies whether such solutions can support augmented-reality tasks. These studies do not consider inference-specific performance metrics. On the other hand, [19] presents a task assignment algorithm that optimizes average execution accuracy, [47] focuses on hierarchical systems, [48] designs QoS-aware task exchanges to optimize energy and latency using a queuing model, and similarly, [49] studies collaboration among devices in DL-based inferences.

The above works study the offline version of the problem, where the task requirements, network, and node parameters are known and fixed. Our work differs in that we consider the practical *online* version where all these parameters evolve with time, possibly in a non-i.i.d. fashion, and additionally, we observe their values after the tasks have been assigned. Compared to our previous work [50], here, we focus on balancing the energy consumption across the nodes, so as to avoid overloading some of them, which, in turn, would disrupt the network and deter them from collaborating. This additional criterion fundamentally changes the learning problem and requires a new model and algorithm.

D. OCO for Resource Management

Our solution relies on the theory of OCO, which has been recently used as the decision engine for several practical resource allocation problems. OCO-based solutions advance and outperform prior studies, which assume that the various problem parameters exhibit static or stationary behavior. Hence, they fail to account for environments characterized by irregular distributional variations, as one expects to happen at edge computing settings due to a small number of devices, small-range cells, and so on. OCO has already been applied, with remarkable success, to the design of dynamic caching algorithms [51], scheduling of radio access policies [52], deployment of mobile vision services [53], and resource management in IoT [54], among many others. Here, we utilize the *constrained OCO* (COCO) framework where, apart from learning how to optimize a time-varying objective function (as was the case in [50]), the system needs to satisfy a long-term budget constraint [55], [56]. We tailor this approach to the problem at hand and modify it to account for the energy fairness constraints and the multiobjective optimization goal.

III. System Model

This section defines the network and node model, and introduces the decision variables and performance metrics.

A. Notation

We denote with bold typeface vectors, e.g., $\boldsymbol{x}$ and $\boldsymbol{g}_t$ for vector functions. The transpose of a vector $\boldsymbol{x}$ is denoted with

$\boldsymbol{x}^\top$. We use capital calligraphic letters for sets, e.g., $\mathcal{X}$. We use the shorthand notation $r_{1:t} \doteq \sum_{\tau=1}^{t} r_\tau$; we use $\|\cdot\|_\infty$ for the infinity ℓ_∞ norm, and $\|\cdot\|$ for the ℓ_2 norm,..., $\mathbf{I}_{\mathcal{S}}(\boldsymbol{x})$ is the indicator function for the membership of $\boldsymbol{x}$ in set $\mathcal{S}$ ($\mathbf{I}_{\mathcal{S}}(\boldsymbol{x}) = 0$) or not ($\mathbf{I}_{\mathcal{S}}(\boldsymbol{x}) = \infty$).

B. Network and Node Model

We model a wireless IoT network with a directed graph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ of $I = |\mathcal{I}|$ nodes and $E = |\mathcal{E}|$ links, which includes an EC that coordinates the task execution, as depicted in Fig. 1. The network operation is time-slotted, and we study it for a set $\mathcal{T}$ of T slots, where, without loss of generality, we assume unit slot duration. The nodes operate with a standard IoT technology such as Wi-Fi or ZigBee. A link $(i, j) \in \mathcal{E}$ exists if node j is within range of node i, and we define the set $\mathcal{I}_i = \{j \mid (i, j) \in \mathcal{E}\} \cup \{i\}$ of one-hop neighbors of each node $i \in \mathcal{I}$ and the respective set $\mathcal{I}_i^- = \mathcal{I}_i \backslash \{i\}$.

Each link $(i, j) \in \mathcal{E}$ has an average data transfer capacity of C_{ijt} (bytes/s) during slot $t \in \mathcal{T}$, which can vary arbitrarily across different links and slots. The link capacities are affected by obstacles, exogenous interference, and so on and can change with time, even being severed. Furthermore, every node i is equipped with a processor unit (PU) of capacity H_{it} cycles/sec during slot t, which can fluctuate arbitrarily over time and even become zero if, e.g., a node fails. We denote with e_{it}^c the computing energy (Joules/cycle) that i spends in t for each processing cycle that it has to execute. This quantity depends on the type and model of the processing unit of the node, the processing speed, and the coexecution of other workloads (if any) [19]. We also denote with e_{ijt}^x the energy consumption (Joules/byte) when transmitting one byte over link $(i, j) \in \mathcal{E}$. These energy costs depend on the effective link capacities [57] and, hence, may vary across the links and over time. We omit the data reception energy as it is typically smaller, but the model can be extended to capture this quantity as well. To streamline presentation, we define the respective vectors $\boldsymbol{e}_t^c$, $\boldsymbol{e}_t^x$, $\boldsymbol{C}_t$, and $\boldsymbol{H}_t$.

The inference tasks that nodes generate are executed locally or at some adjacent node or dropped if it is deemed necessary (for load admission reasons). We define the vector $\boldsymbol{\beta}_t = (\beta_{it}, i \in \mathcal{I})$ where $\beta_{it} \in [0, \beta_{\max}]$ is the number of tasks generated by node $i \in \mathcal{I}$ during slot t. The tasks induce computing workloads (cycles) of $\boldsymbol{p}_t = (p_{it}, i \in \mathcal{I})$ and data loads (bytes) of $\boldsymbol{s}_t = (s_{it}, i \in \mathcal{I})$. A node i can outsource one or more tasks to its one-hop neighbors $\mathcal{I}_i^-$. This facilitates nodes with limited computing resources or ML libraries that, it turns out, are inefficient (e.g., inaccurate) for their own tasks. We model the ability of node j to execute the tasks of node i using the normalized accuracy parameter $a_{ijt} \in [0, 1]$, which might change over time; e.g., the nodes may update their training datasets, obtain a new model, and so on [58]. We also introduce the vector of accuracy values $\boldsymbol{a}_t = (a_{ijt}, i, j \in \mathcal{I})$.

Our model is general since we do not restrict it to specific transmission or interference schemes, nor do we make assumptions about fixed and known link throughput, node PU capacities, and so on. These parameters can change in every slot in an unpredictable fashion (not necessarily following a stationary process), and they may remain unknown at the beginning of each slot when the EC devices the assignment.

C. Task Assignment and Performance Metrics

We focus on the EC and study its task assignment decisions for each slot. We define the portion of tasks $x_{ijt} \in [0, 1]$ of node i, which are sent to each neighbor $j \in \mathcal{I}_i$ during slot t, and we denote with x_{iit} the portion of tasks i executes locally and with x_{i0t} the rejected tasks (admission control). We collect these variables in the *assignment* vector $\boldsymbol{x}_t = (x_{ijt}, (i, j) \in \mathcal{E})$, which is subject to a multisimplex constraints

$$\mathcal{X} = \left\{ \boldsymbol{x}_t \in [0, 1]^E \;\middle|\; \sum_{j \in \mathcal{I}_i} x_{ijt} = 1, \quad \forall i \in \mathcal{I} \right\}. \tag{1}$$

We note that these continuous variables are, clearly, an approximation for some systems where the tasks are indivisible. Still, since our focus is on large systems where the nodes have to execute a large number of tasks per slot, having a continuous relaxation, which is then rounded, is a practical compromise that does not introduce significant errors.[1] Such continuous approximations have been used extensively in similar edge computing or edge caching problems. Finally, we note that in smaller systems where the scale itself does not compensate for the rounding errors, one can interpret $\boldsymbol{x}$ as a probabilistic policy. This only changes the fact that the performance bounds (derived below) should be interpreted in expectation (probabilistic).

The EC wishes to maximize the aggregate task execution accuracy while minimizing the total energy consumption and the computation and transmission delays. Starting with the former, the *performance* for each task depends on the achieved accuracy, and we can define it for all tasks at the network level

$$u_t(\boldsymbol{x}) = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}_i} \alpha_{ijt} \beta_{it} \log\left(x_{ijt} + 1\right) \quad \forall t \in \mathcal{T}. \tag{2}$$

This model captures the salient properties of the problem. First, the processing performance for each pair of nodes (i, j) depends on the number of tasks that i generates (parameter β_{it}) and increases logarithmically with the tasks that i sends to j. The logarithm is a concave function with decreasing gradient and, hence, captures the effect of diminishing returns, i.e., the per-task utility diminishes as the nodes send more tasks to each neighbor (saturation). From a different perspective, this logarithmic transformation can be seen as a method for ensuring proportional performance fairness. Finally, the performance depends on the accuracy α_{ijt}, which expresses how well the tasks of node i match the ML model of node j. We stress that we do not make any assumption about this information being available beforehand.

In order to capture the task delay, we consider the following contributing factors: 1) the transmission delay that increases with the amount of data and decreases proportionally to each link's effective data transfer capacity and 2) the computing delay that, for each processing node, increases with the aggregated compute load assigned to it and decreases proportionally

[1]For instance, if the algorithm decides to send over a link 22.7 frames, then, rounding this to 23 or 22 is practically the same.

to its CPU speed. Putting these together, we define the delay during each slot t as a function of $\boldsymbol{x}$

$$d_t(\boldsymbol{x}) = \sum_{i\in\mathcal{I}} \left(\underbrace{\sum_{j\in\mathcal{I}_i^-} \frac{\beta_{it} x_{ijt} s_{it}}{C_{ijt}}}_{\text{transmission delay}} + \underbrace{\sum_{j\in\mathcal{I}_i} \frac{\beta_{it} x_{ijt} p_{it}}{H_{jt}}}_{\text{computing delay}} \right). \tag{3}$$

This model is aligned with prior works that assume linear dependence on (communication or computation) workloads and is valid as long as the system is not oversaturated. Besides, our analysis is valid for nonlinear delay models (e.g., M/M/1 queues) as long as they preserve convexity.

For the energy cost, different nodes induce different energy consumption,[2] and the same holds for the transmission energy cost.[3] Thus, the EC considers the energy cost for all nodes

$$v_t(\boldsymbol{x}) = \sum_{i\in\mathcal{I}} v_{it}(\boldsymbol{x}), \quad \text{where} \tag{4}$$

$$v_{it}(\boldsymbol{x}) = \sum_{j\in\mathcal{I}_i^-} \underbrace{e^x_{ijt} s_{it} \beta_{it} x_{ijt}}_{\text{transm. energy}} + \sum_{j\in\mathcal{I}_i} \underbrace{e^c_{it} p_{jt} \beta_{jt} x_{jit}}_{\text{comp. energy}}. \tag{5}$$

Similar to delay, our energy model has a linear dependence on the transmitted and processed data, which is aligned with prior theoretical and measurement studies (see [19]).

Putting together the above expressions for the performance and cost components, we introduce the network *utility* function $f_t : \mathcal{X} \mapsto \mathbb{R}$, $\forall t \in \mathcal{T}$ defined as

$$f_t(\boldsymbol{x}) = \gamma_1 u_t(\boldsymbol{x}) - \gamma_2 d_t(\boldsymbol{x}) - \gamma_3 v_t(\boldsymbol{x}). \tag{6}$$

Parameters $\gamma_1, \gamma_2, \gamma_3 \in \mathbb{R}_+$ have a twofold role here. First, they tune the relative importance of each utility component, i.e., of the accuracy, latency, and energy, based on the priorities of the EC who might wish to prioritize one over the other. Second, they normalize the measurement units of these component functions. There are many ways one can set these parameters, e.g., by using the maximum attainable values of each function component.

It is clear that one can use different performance and cost functions based on the studied system and application. Our analysis in the following is readily extendable to these cases as long as the convexity of the final utility functions $\{f_t\}_t$ is preserved and can even be applied to nonconvex functions through proper convexification techniques.

IV. Problem Statement and Benchmark

The EC aims to find the task assignment $\{\boldsymbol{x}_t\}_{t=1}^T$ that maximizes the *aggregate utility* of the network over time by balancing the performance, delay, and energy consumption that its assignment decisions induce in each slot and without requiring knowing, in advance, the values of the time-varying parameters of the network and nodes. The sequence of events for this learning model is summarized in Fig. 2. At the same time, as a minimal form of incentive provision to the participating nodes, the EC wishes to ensure that their energy consumption is as balanced as possible. Energy is probably the most scarce resource in such systems, and the nodes would be less willing to agree to serve other tasks if this leads to an excessive (compared to others) consumption of their energy budget. Besides, asymmetric energy consumption is likely to eventually render inoperative some nodes, thus impacting the ability of the entire network to process future tasks.

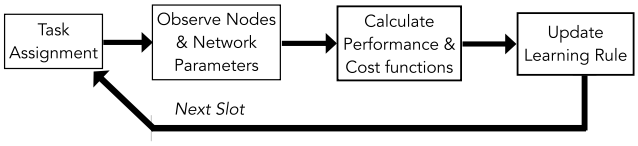


Fig. 2. Sequence of events in the decision process of the EC.

With these considerations in mind, the optimization problem that describes the EC's objectives and constraints when devising the task assignment policy can be expressed as follows:

$$\mathbb{P}: \quad \max_{\boldsymbol{x}\in\mathcal{X}} \quad \sum_{t=1}^{T} f_t(\boldsymbol{x}) \tag{7}$$

$$\text{s.t.} \quad \sum_{t=1}^{T} \boldsymbol{g}_t(\boldsymbol{x}) \preceq \boldsymbol{0}. \tag{8}$$

where (8) expresses the energy balancing requirement; namely, we introduce

$$\boldsymbol{g}_t(\boldsymbol{x}) \triangleq \left(g_t^{(1)}(\boldsymbol{x}), g_t^{(2)}(\boldsymbol{x}), \ldots, g_t^{(E)}(\boldsymbol{x}) \right), \quad \text{with} \tag{9}$$

$$g_t^{(n)}(\boldsymbol{x}) = v_{it}(\boldsymbol{x}) - v_{jt}(\boldsymbol{x}) - \varepsilon_{ij}^{th}, \quad \forall n = (i,j) \in \mathcal{E} \tag{10}$$

and note that we define two constraints for every pair of nodes i and j, i.e., one for (i, j) and one for (j, i). In other words, this is a bidirectional constraint, which enables enforcing a simple and practical fairness with respect to the energy consumption of the nodes. Parameter $\boldsymbol{\varepsilon}_{th} = (\varepsilon_{ij}^{th} \geq 0, \forall (i,j) \in \mathcal{E})$ allows to decide how strict this fairness criterion should be and also defines this slack differently for each pair of nodes (if there is a need to do so). As it will become clear from our analysis below, the model and the algorithm can also handle additional energy consumption constraints, e.g., by enforcing an energy budget per node, and this can be even set differently based on each node's energy priorities.

Proceeding with the solution of the problem, there are three challenges that render classical optimization methods insufficient for $\mathbb{P}$. First, the EC has to decide $\boldsymbol{x}_t$ at the beginning of each slot t, whereas the values of parameters such as the number of tasks, compute and link capacities, and task execution accuracy by each node are revealed only at the end of the slot.[4] Second, these parameters might change arbitrarily across slots in the general case; thus, we cannot benefit from typical statistical models that use, e.g., running average or from stochastic optimization tools that presume stationarity [59]. Finally, these decisions $\{\boldsymbol{x}_t\}_t$ need to be devised in a near real-time scale, even for large networks. For these reasons, we will resort to the COCO framework [56].

[2]For instance, some CPUs are more energy-efficient than others, or the energy consumption might be more costly for some nodes, and so on

[3]Due to different Tx/Rx capabilities of nodes or channel conditions.

[4]The link throughput is affected by the volatile channel quality, interference, and so on, and in practice, it can be accurately known only a posteriori.

The COCO framework has applications in various fields as it provides a powerful framework for decision-making in dynamic and uncertain environments where the objective function or constraints may change over time. Yet, such algorithms are known to be particularly challenging to design. To make this more specific, the key question is how we measure the performance of the algorithms, namely, what is the benchmark solution that the algorithm aims to approach online. For problems without time-varying constraints (standard OCO), the performance is quantified with the metric of regret

$$\mathcal{R}_T = \sum_{t=1}^{T} \left(f_t\left(\boldsymbol{x}^\star\right) - f_t\left(\boldsymbol{x}_t\right) \right) \tag{11}$$

where $\boldsymbol{x}^\star = \arg\max_{x\in\mathcal{X}} \sum_{t=1}^{T} f_t(\boldsymbol{x})$ is the single-best task assignment that the EC would have selected if it had a priori knowledge of the entire future, i.e., knew all utility functions $\forall t = 1, \ldots, T$. Designing an algorithm that achieves asymptotically zero regret, i.e., $\lim_{T\to\infty} \mathcal{R}_T/T = 0$, is equivalent to saying that the algorithm learns to perform optimally with respect to the benchmark $\boldsymbol{x}^\star$ (no-regret algorithm).

In the case of COCO that we study here, we are additionally interested in ensuring asymptotically-zero *constraint violation*, $\lim_{T\to\infty} \mathcal{V}_T/T = 0$, where

$$\mathcal{V}_T = \left\| \left[\sum_{t=1}^{T} \boldsymbol{g}_t\left(\boldsymbol{x}_t\right) \right]_+ \right\|$$

and note that we take the norm since we have multiple constraints (one for each pair of nodes). Also, in this case, the benchmark naturally changes, as it has to satisfy the constraints, as well. Ideally, we would like our algorithm to learn to perform as well as a benchmark policy $\boldsymbol{x}^\star$ that maximizes the performance and belongs to the set

$$\mathcal{X}_T^{\max} = \left\{ \boldsymbol{x} \in \mathcal{X} \;\middle|\; \sum_{t=1}^{T} \boldsymbol{g}_t\left(\boldsymbol{x}\right) \preceq \boldsymbol{0} \right\}. \tag{12}$$

Unfortunately, [60] proved that designing COCO algorithms, which can learn with respect to such competitive benchmarks, is impossible (i.e., not achievable). Hence, follow-up works considered less demanding benchmarks that are selected from more restricted sets, such as

$$\mathcal{X}_T = \left\{ \boldsymbol{x} \in \mathcal{X} \;\middle|\; \boldsymbol{g}_t\left(\boldsymbol{x}\right) \preceq \boldsymbol{0}, \quad \forall t \in \mathcal{T} \right\}. \tag{13}$$

In this case, $\boldsymbol{x}^\star$ maximizes again the performance but has to satisfy the constraints in each slot, and it is clear that it is $\mathcal{X}_T \subseteq \mathcal{X}_T^{\max}$. For a detailed discussion about COCO benchmarks and impossibility results, we kindly refer the reader to [56].

Our goal in this article is to develop a dynamic algorithm that allows the EC to devise the assignments $\boldsymbol{x}_t$ at the beginning of each slot t, aiming to achieve satisfactory performance with respect to $\boldsymbol{x}^\star \in \mathcal{X}_T$; specifically, we aim to achieve asymptotically zero regret and constraint violation. Moreover, we wish to prove that these bounds are guaranteed for any sequence of functions $\{f_t\}_{t=1}^T$ and $\{\boldsymbol{g}_t\}_{t=1}^T$. In practice, this means that the algorithm learns to perform as well as the ideal policy $\boldsymbol{x}^\star$ without, obviously, requiring to know the latter, for every possible scenario/setting the EC might encounter.

V. OTA Policy

We start with some key properties of the system and problem $\mathbb{P}$. These will be used in the design of the algorithm and in analyzing its performance. First, observe that the set $\mathcal{X}$ is convex, compact, and bounded by definition, with the Euclidean diameter $\|\boldsymbol{x}\| \le 2I, \forall \boldsymbol{x} \in \mathcal{X}$. Moreover, functions $f_t, g_t^{(n)} : \mathcal{X} \mapsto \mathbb{R}, \forall t \in \mathcal{T}, n$, are convex and Lipschitz continuous with constants $L_{f_t} \le L_f$ and $L_{g_t} \le L_g$, respectively, for some fixed parameters L_f and L_g. Also, since $\mathcal{X}$ is compact, it holds $|f_t(\boldsymbol{x})| \le F$ and $\|\boldsymbol{g}_t(\boldsymbol{x})\| \le G, \forall t \in \mathcal{T}, \boldsymbol{x} \in \mathcal{X}$, for some values of F and G that depend on the model parameters and will be specified in the sequel.

A. Learning Algorithm

The assignment algorithm is based on a saddle point reformulation of $\mathbb{P}$. We rely on the seminal Follow-The-Regularized-Leader (FTRL) algorithm [61] and its recent extension for COCO problems [56] to design a learning algorithm that is fast and effective with respect to both $\mathcal{R}_T$ and $\mathcal{V}_T$. First, we define the regularized Lagrangian

$$\mathcal{L}_t\left(\boldsymbol{x}, \boldsymbol{\lambda}\right) = r_t\left(\boldsymbol{x}\right) + \boldsymbol{c}_t^\top \boldsymbol{x} + \boldsymbol{\lambda}^\top \boldsymbol{g}_t\left(\boldsymbol{x}\right) - q_t\left(\boldsymbol{\lambda}\right) \tag{14}$$

where $\boldsymbol{c}_t \doteq \nabla_x f_t(\boldsymbol{x}_t)$ is the gradient of the utility at slot t and $\boldsymbol{\lambda} = (\lambda_n \ge 0,\ n = 1, \ldots, E)$ is the vector of dual variables, one for each constraint function $g_t^{(n)}$. Interestingly, using $\boldsymbol{c}_t$ instead of the function (known as *linearization* in OCO; see [61]) reduces the computation and memory requirements of the algorithm. Function $r_t : \mathcal{X} \mapsto \mathbb{R}$ is an entropic regularizer for the primal variables $\boldsymbol{x}$, while $q_t : \mathbb{R}_+^E \mapsto \mathbb{R}$ is a quadratic regularizer for the dual variables. These are selected as such due to the geometry of the respective constraint spaces, i.e., the multisimplex for the primal variables and a hyperplane for the dual variables.

In particular, the primal regularizer $\forall t \in \mathcal{T}$ is defined as

$$r_{1:t}\left(\boldsymbol{x}\right) = \frac{\sigma_{1:t}}{2} \left(I \log I + \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{I}_i} x_{ij} \log x_{ij} \right), \quad \text{with}$$
$$\sigma_{1:t} = \sigma \sqrt{h_{1:t}}, \ \text{and} \ h_t = \left\| \boldsymbol{c}_t + \boldsymbol{\lambda}_t^\top \nabla \boldsymbol{g}_t\left(\boldsymbol{x}_t\right) \right\|_\infty^2. \tag{15}$$

where recall that $r_{1:t} \doteq \sum_{\tau=1}^{t} r_\tau$ is just a shorthand notation and $\|\cdot\|_\infty$ is the infinity norm. Parameter σ_t shapes the regularization, i.e., the learning rate, and indeed, we can see from the above formula that it adapts to the encountered scenario since it depends on the observed gradients and constraint violations. The quadratic regularizer, on the other hand, is defined as

$$q_0\left(\boldsymbol{\lambda}\right) = \mathbf{I}_{\mathbb{R}_+^E}\left(\boldsymbol{\lambda}\right) \ \text{and} \ q_t\left(\boldsymbol{\lambda}\right) = \frac{\phi_t}{2} \|\boldsymbol{\lambda}\|^2, \quad t \ge 1,$$
$$\text{with}: \quad \phi_t = \frac{1}{\delta_t} - \frac{1}{\delta_{t-1}}, \quad \delta_{t-1} \ge \delta_t > 0. \tag{16}$$

where δ_t is the dual learning rate at each slot $t \in \mathcal{T}$, and following the rationale in [56], we define it as follows:

$$\delta_t = \frac{\delta}{\max\left\{ \sqrt{\sum_{\tau=1}^{t} \left\| \boldsymbol{g}_t\left(z_t\right) \right\|^2},\ t^\theta \right\}}. \tag{17}$$

Parameter $\theta \in [0, 1)$ has a special role; it can be selected by the EC to prioritize the convergence speed of the constraint violation and regret based on whether performance or fairness is of higher interest. Nomenclature summarizes the key notations for the algorithm.

Algorithm 1 OTA

1: **Initialize:** $\boldsymbol{x}_1 \in \mathcal{X},\ \boldsymbol{\lambda}_1 \geq 0$
2: **for** $t = 2, 3, \cdots, T$ **do**
3: Calculate $r_{1:t-1}$ and apply the task assignment:

$$\boldsymbol{x}_t = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \sum_{\tau=1}^{t-1} \mathcal{L}_\tau(\boldsymbol{x}, \boldsymbol{\lambda}_\tau) \tag{18}$$

4: Observe $\nabla f_t(\boldsymbol{x}_t)$ and $\boldsymbol{g}_t(\boldsymbol{x})$ and calculate:

$$\boldsymbol{z}_t = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \sum_{\tau=1}^{t} \mathcal{L}_\tau(\boldsymbol{x}, \boldsymbol{\lambda}_\tau) \tag{19}$$

5: Calculate $q_{0:t}$ and update the dual variables:

$$\boldsymbol{\lambda}_{t+1} = \arg\max_{\boldsymbol{\lambda}\in\mathbb{R}_+^E} \sum_{\tau=1}^{t} \mathcal{L}_\tau(\boldsymbol{z}_\tau, \boldsymbol{\lambda}) \tag{20}$$

6: **end for**

The dynamic policy that sets the assignment decisions at each slot implements the steps in Algorithm 1, which we call online task assignment (`OTA`). First, we select randomly a feasible primal and dual variable vector (step 1). Then, at each slot t, we update the regularization parameters and calculate the new task assignment $\boldsymbol{x}_t$ by solving the respective primal update of the saddle point problem (step 3). The EC applies this policy $\boldsymbol{x}_t$ immediately, and at the end of the slot, it observes the performance and cost functions (step 4) and calculates the *prescient* assignment $\boldsymbol{z}_t$. This is the assignment that the EC would have proposed if it had known these (unknown) functions. Using this updated information ($\boldsymbol{z}_t$ instead of $\boldsymbol{x}_t$), the EC calculates the dual regularizer parameters and updates the dual variables in step 5. The above steps are repeated as long as the system is in operation.

B. Performance Analysis and Implementation Aspects

Algorithm 1 comes with guarantees for its performance, which, additionally, can be fully characterized and linked to the problem parameters. This latter aspect favors interpretability. In detail, we have the following result.

Lemma 1: Algorithm 1 ensures the following regret and constraint violation convergence rates based on the tuning parameter $\theta \in [0, 1)$:

$$\mathcal{R}_T = \mathcal{O}\left(T^{5/8}\right) \quad \mathcal{V}_T = \mathcal{O}\left(T^{3/4}\right), \qquad \text{when} \quad \theta < 1/2$$
$$\mathcal{R}_T = \mathcal{O}\left(T^{3-\theta/4}\right) \quad \mathcal{V}_T = \mathcal{O}\left(T^{1+\theta/2}\right), \quad \text{when} \quad \theta \geq 1/2.$$

Using θ, the EC can improve the learning bounds for the regret or the constraint violation, hence prioritizing one metric over the other. We note, however, that this prioritization concerns the worst case scenarios, which typically arise in adversarial conditions, while, as our experiments show, in more typical cases, both metrics perform equally well. Besides, the constraint violation can also be tuned by adjusting the threshold parameters in the energy consumption. Hence, algorithm `OTA` provides practical tuning versatility for adapting to the nodes' energy preferences and applications' requirements.

Regarding the implementation of the algorithm, it is crucial to stress that it is lightweight in terms of computation and communication overheads. This is of utmost importance for the problem at hand since the nodes have scarce resources, which should not be spent on communication overheads, while the EC is also a resource-constrained device that needs to economize computations. Starting with the computing overheads, the optimization problems for devising $\boldsymbol{x}_t$ and $\boldsymbol{z}_t$ can be solved in closed form and, hence, in constant $\mathcal{O}(1)$ time. This is formalized with the following proposition.

Proposition 1: The primal and prescient updates (18) and (19) can be executed using the closed-form expressions

$$x_{ijt} = \frac{\exp\frac{2w_{(t-1)ij}}{\sigma_{1:t-1}}}{\sum_{j\in\mathcal{I}_i}\exp\frac{2w_{(t-1)ij}}{\sigma_{1:t-1}}}, \quad z_{ijt} = \frac{\exp\frac{2w_{(t)ij}}{\sigma_{1:t}}}{\sum_{j\in\mathcal{I}_i}\exp\frac{2w_{(t)ij}}{\sigma_{1:t}}}$$

where $w_{ij(t-1)} = -\eta_{ij(t-1)} - \epsilon_{ij(t-1)}$ and $\eta_{ij(t-1)} = c_{ij1:t-1}$, and we defined $\boldsymbol{\epsilon}_{t-1}^\top \boldsymbol{x} \doteq \sum_{\tau=1}^{t-1} \lambda_\tau^\top g_\tau(\boldsymbol{x})$.

This particularly attractive property stems from the usage of the entropic regularizer, which allows us to solve the decision update problems in full, obtaining these convenient closed-form expressions and avoiding the typically demanding projection operations involved in OCO iterations. Fortunately, the dual update is also easy to perform as the projection there needs to be done over the entire half-space (the dual variables are unbounded). Furthermore, the EC needs only to communicate the assignment decisions to the respective nodes (and not the entire vector to all nodes) or can instead broadcast the plan at once, while the nodes need only to communicate to EC their parameters (numerical values) pertaining to tasks and resources at the end of each slot. In summary, we verify that the algorithm is computationally lightweight, and the communication overheads are minimal, i.e., two-way communication of numerical parameters among the nodes and EC.

Last, it is important to discuss the few technical assumptions of the algorithm. First, the EC should be able to observe the information $\nabla f_t(\boldsymbol{x}_t)$ and $\boldsymbol{g}_t(\boldsymbol{x}_t)$. Revisiting (2)–(4), we can see that these functions involve information about the communication and computing capacities, the energy consumption parameters, and the inference accuracy parameters. All this information, apart from the accuracy, can be easily measured at the end of each slot by the devices or the EC. The accuracy parameters, on the other hand, require either the periodic usage of labeled data (when available) or proxy metrics such as the confidence intervals of a classifier, as in [19] and [62]. Second, the model assumes a fixed set of nodes that interact throughout the time horizon $\mathcal{T}$. Yet, this assumption is not as restrictive as it appears. For instance, if a node is disconnected from the network, the algorithm will start observing zero utilities for the tasks sent to it and will gradually learn not to consider it as a possible task recipient. Also, if there is some dramatic network change, e.g., several nodes depart, or new nodes join the system, then the EC can simply restart the algorithm. This

TABLE I
SUMMARY OF EXPERIMENT SETUPS

Cases (Perturbations)	Type
1: Stationary	Uniform Static Distribution
2: Dynamic	Adversarial Ping-Pong
3: Static	Uniform Zipf-type Distribution
4: Dynamic	Time-varying
Scenarios (Diversity)	**Type**
1: Homogeneous	Nodes & tasks are similar
2: Device-Heterogeneous	Diverse node resources
3: Device/Task-Heterog.	Diverse nodes and tasks
Priority Settings	**Notes**
1: $\gamma_1=10$ $\gamma_2=\gamma_3=0.3$	Priority to accuracy, delay and energy equal
2: $\gamma_1=10$ $\gamma_2=\gamma_3=1$	Increase priority of delay, energy
3: $\gamma_1=10$ $\gamma_2=\gamma_3=3$	Increase priority of delay, energy
4: $\gamma_1=10$ $\gamma_2=1, \gamma_3=3$	Prioritize energy more
5: $\gamma_1=10$ $\gamma_2=3, \gamma_3=1$	Prioritize delay more

TABLE II
RECOGNITION ACCURACY VERSUS ML CONFIGURATIONS

Accuracy	Haar		LBP	
	Frontal Face	Side Face	Frontal Face	Side Face
LBPH	0.82	0.80	0.62	0.56
Eigen Faces	0.74	0.70	0.50	0.48
Fisher Faces	0.42	0.30	0.24	0.18

will only have a small performance effect as the algorithm will need to learn the new cooperation policies.

VI. PERFORMANCE EVALUATION

The evaluation setup comprises a set of IoT nodes connected over Wi-Fi, performing image processing tasks. Our goal is to study: 1) the collaboration benefits when the nodes exhibit diversity in tasks and resources and 2) the convergence properties, in terms of regret and constraint violation, of the learning algorithm under a variety of scenarios.

A. Experiment Setups and Evaluation Scenarios

1) Task Properties: The inference tasks are to recognize faces in images. We use the open-access dataset [19] obtained with OpenCV, which performs this task in two stages by combining a face detector (Haar or LBP) with one of the face recognizers: local binary pattern histogram (LBPH), eigen faces, or Fisher faces. This setting yields six ML pipelines with different inference accuracy values, as summarized in Table II, when applied to the face database [63]. In this experimental setup, we have two types of tasks: face detection in frontal- and side-facing images.

We create a rich combination of experimental scenarios, as summarized in Table I, where we consider four *cases* regarding how the system parameters vary across time (from stationary to time-varying to adversarial) and three *Scenarios* regarding how these parameters vary across nodes (from homogeneous to fully diverse). We also test these cases and scenarios under five different priority settings (combinations of weights γ_1, γ_2, and γ_3). In detail, in *stationary Case 1*, the number of generated tasks β_{it} for each node i at each slot t is sampled uniformly from $\{2,3,4,5,6\}$, the data load s_{it} is uniformly sampled from $[144,216]$ kB, the computing workload p_{it} is sampled from $[180,912]$ Mcycles, the energy consumption (per task/cycle) e^c_{it} is uniformly distributed in $[0.16,0.24]$ nJ, and the energy consumption e^x_{ijt} for transmitting one bit over each link $(i,j)\in\mathcal{E}$ ranges from 1 to 21 nJ. This energy consumption is inversely proportional to the channel quality. These parameters are derived from the measurements reported in [19]. In the *adversarial Case 2*, these parameters fluctuate between the extreme points of the respective intervals, over successive time slots. This *ping-pong* trajectory is the most challenging one for the learner, as it is likely to drastically change the induced decisions, and in fact, it is used to prove lower bounds of learning algorithms (see [13]). Hence, *Case 2* is of special interest because an algorithm performing well in this case is guaranteed to also perform well in other cases.

Case 3 is a static case where the parameters are drawn from an asymmetric, Zipf-like distribution. To generate these values, we use the values of parameters in *Case 1*, which we shift to create the asymmetry. In particular, let us denote with $[\kappa_{\min},\kappa_{\max}]$ the range of values in *Case 1* for some system parameter and with (a,b) the sampling range of the Zipf distribution. Using the inverse cdf of Zipf given by $x=(1/u)^{1/s}$, where $u\sim Uniform(0,1)$ and s is the Zipf exponent, i.e., as s increases, the distribution becomes more concentrated toward the lower (left) end of the range. The resulting random x_{mapped} (of the initial value x) belongs to the target interval and can be calculated as follows:

$$x_{\text{mapped}}=\kappa_{\min}+(\kappa_{\max}-\kappa_{\min})\,\frac{x-a}{b-a}.$$

Finally, we also consider *Case 4*, which is a dynamic random setting where the parameters are drawn from a time-shifting distribution such as in the case of computing capacity

$$H_{it}=A\sin(\omega\pi t)+B+n_{it}.$$

To enable a more consistent comparison with *Case 1* and *Case 2*, we define A as the amplitude, set to $(\kappa_{\max}-\kappa_{\min})/2$, and B as the mean, chosen as $(\kappa_{\max}+\kappa_{\min})/2$ to match the mean of the original randomly generated data. Perturbations are introduced via n_{it}, which is uniformly distributed over $(-\zeta,\zeta)$, for some values of ζ based on the setting.

2) Node Profiles: IoT nodes, such as Raspberry Pi, Jetson, or Arduino devices, exhibit variations in CPU. We categorize the nodes into three groups: high-CPU nodes with speed $H_{it}\in[1.4,1.6]$ GHz, medium-CPU nodes with speed $H_{it}\in[1.10,1.15]$ GHz, and low-CPU nodes $H_{it}\in[0.7,0.9]$ GHz.

Based on these properties, we use three *evaluation scenarios*.

1) *Sc1*: All nodes are medium-CPU, all ML configurations are LBP and eigen faces, and all tasks are side faces.
2) *Sc2*: All nodes are medium-CPU, ML libraries are randomly configured, and all tasks are side faces.
3) *Sc3*: Half nodes are high-CPU, and the rest are low-CPU; ML libraries are randomly configured; and half nodes create frontal (side) tasks with probability 0.8 (0.2), and the other half create tasks with the opposite probabilities.

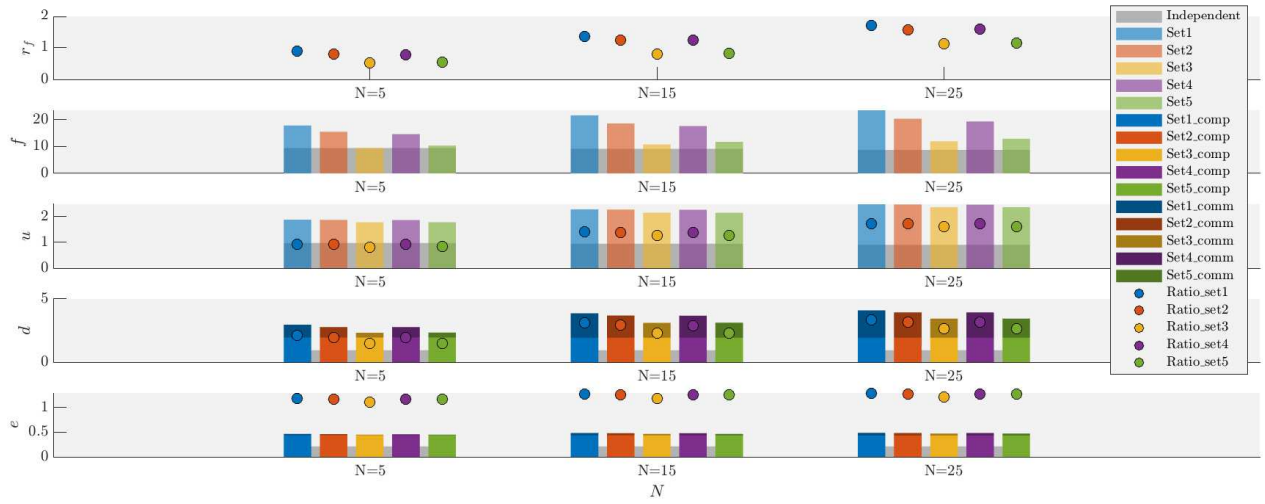


Fig. 3. Improvement ratios r_f, r_u, r_d, and r_v for each node's utility, delay, and energy, respectively, and the average total utility f, inference u, delay d, and energy v under cooperative and independent operation for benchmarks $\boldsymbol{x}^\star$ and $T = 1000$. Results averaged over 50 runs. Evaluations are repeated for five settings: Set1: $\gamma_1 = 10$ and $\gamma_2 = \gamma_3 = 0.3$; Set2: $\gamma_1 = 10$ and $\gamma_2 = \gamma_3 = 1$; Set3: $\gamma_1 = 10$ and $\gamma_2 = \gamma_3 = 3$; Set4: $\gamma_1 = 10, \gamma_2 = 1$, and $\gamma_3 = 3$, and Set5: $\gamma_1 = 10, \gamma_2 = 3$, and $\gamma_3 = 1$, where γ_1 (accuracy), γ_2 (delay), and γ_3 (energy).

Sc1 is the references homogeneous scenario, *Sc2* captures the device heterogeneity, and *Sc3* device and task heterogeneity.

3) Wireless Network: The network is generated with a random geometric graph model. The nodes are randomly placed within a 100×100 m area, and we assume that nodes within a distance of 50 m are in communication range and can exchange tasks. These links constitute the set $\mathcal{E}$. To streamline the simulation scenarios, we introduce a new parameter, denoted as ρ_{ijt} to represent channel quality, which is uniformly distributed within the normalized range $[0, 1]$. The transmission capacity C_{ijt} of each link is naturally associated with its quality ρ_{ijt} and distance d_{ijt}, i.e., $C_{ijt} = \delta_{ij}\rho_{ijt}/d_{ijt}$ where δ_{ij} is normalization parameter.[5] Recall, finally, that the duration of the time slots is considered unitary.

B. Evaluation Results

We start by demonstrating the gains of collaboration in the studied inference streaming problem. We compare the achievable performance using the following metrics.

1) r_f, r_u, r_d, and r_e are the improvement ratios of each node's utility, accuracy, delay, and energy in cooperative mode compared to independent mode (executing its own tasks), i.e.,

$$r_f = \frac{f_{1:T}\left(\boldsymbol{x}^\star\right) - f_{1:T}\left(\boldsymbol{x}_{\text{ind}}\right)}{N f_{1:T}\left(\boldsymbol{x}_{\text{ind}}\right)}, \quad r_u = \frac{u_{1:T}\left(\boldsymbol{x}^\star\right) - u_{1:T}\left(\boldsymbol{x}_{\text{ind}}\right)}{N u_{1:T}\left(\boldsymbol{x}_{\text{ind}}\right)}$$
$$r_d = \frac{d_{1:T}\left(\boldsymbol{x}^\star\right) - d_{1:T}\left(\boldsymbol{x}_{\text{ind}}\right)}{N d_{1:T}\left(\boldsymbol{x}_{\text{ind}}\right)}, \quad r_e = \frac{v_{1:T}\left(\boldsymbol{x}^\star\right) - v_{1:T}\left(\boldsymbol{x}_{\text{ind}}\right)}{N v_{1:T}\left(\boldsymbol{x}_{\text{ind}}\right)}$$

where recall the notation $f_{1:T}(\boldsymbol{x}) = \sum_{t=1}^{T} f_t(\boldsymbol{x})$, $\boldsymbol{x}_{\text{ind}}$ is the assignment when nodes execute tasks independently, and $\boldsymbol{x}^\star = \arg\max_{\boldsymbol{x}\in\mathcal{X}_T} \sum_{t=1}^{T} f_t(\boldsymbol{x})$ is the benchmark. The network size varies from $I = 5$ to 45 nodes, the time horizon is $T = 1000$, and we average the results over 50 runs. We discretize the continuous output of the algorithm by a typical rounding scheme while respecting the capacity constraints.

The collaboration results for *Case 1*, $Sc = 3$, varying numbers of network nodes and five priority settings are summarized in Fig. 3. We observe that, under Set1 with $N = 5$ nodes, collaborative operation improves inference u by 93.03% and utility f by 89.23%. In all other configurations, collaborative task execution consistently outperforms the baseline of independent operation (shown in gray bars). As the number of nodes increases, collaboration leads to a 172.98% improvement in u and a 168.06% improvement in f when $N = 25$. Finally, we see that the benefits increase with the size of the network, as this creates more collaboration opportunities.

We also observe again the impact of the priority parameters across the five configurations summarized in Table I. When we adjust the performance priorities by setting $\gamma_2 = \gamma_3 = 1$ (Set2), we observe that, in the scenario with $Sc = 3$ and $N = 25$, the improvement ratio of delay decreases from 334.7% to 318.13%, and the improvement ratio of energy drops from 129.1% to 127.3% compared to Set1. This indicates that increasing either γ_2 or γ_3—that is, assigning more weight to delay and energy—leads the network to reduce inter-node transmissions, thereby lowering the delay and energy consumption associated with such communication. This behavior ultimately minimizes these costs and improves the overall system performance. Moreover, compared to energy, the difference between transmission and computation costs is smaller in the case of delay. As shown in Table III, which summarizes the comparison of cooperative versus independent operation for five different priority settings, when we further adjust the priority parameters to $\gamma_2 = 1$, $\gamma_3 = 3$ (Set4) and $\gamma_2 = 3$, $\gamma_3 = 1$ (Set5), we find that increasing γ_2 has a more pronounced impact on the overall performance compared to Set2.

[5]These parameters account for aspects beyond the control of the EC, e.g., transmission power, which the underlying communication scheme might use. Our framework is orthogonal and oblivious to these.

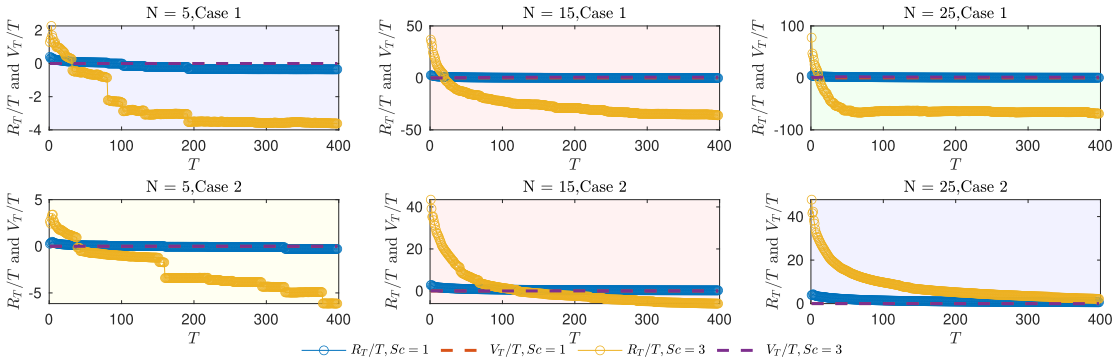


Fig. 4. Average regret $\mathcal{R}_T/T$ and constraint violation $\mathcal{V}_T/T(\varepsilon_{ij}^{th} = 0.3)$ of Algorithm 1 for *Case 1* and *2* (Sc = 1 and 3).

TABLE III

AVERAGE UTILITY, DELAY, AND ENERGY WITH COOPERATIVE VERSUS INDEPENDENT OPERATIONS FOR $N = 15$, $Sc = 3$, CASE 1, AND THE FIVE PRIORITY PROFILES FROM FIG. 3

Set.	Accur.	Delay (C)	Delay (I)	Energy (C)	Energy (I)
1:	2.28	3.87	0.94	0.48	0.21
2:	2.27	3.69	0.94	0.48	0.21
3:	2.15	3.11	0.94	0.46	0.21
4:	2.27	3.68	0.94	0.48	0.21
5:	2.15	3.13	0.94	0.47	0.21

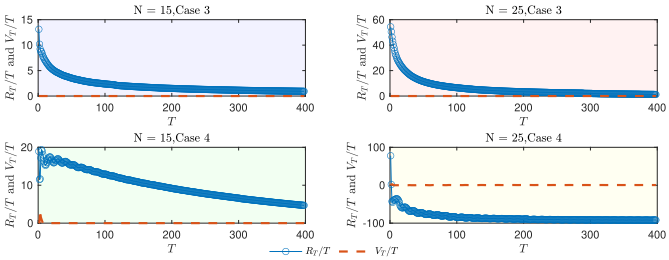


Fig. 5. Average regret $\mathcal{R}_T/T$ and constraint violation $\mathcal{V}_T/T$ of algorithm for *Case 3* and *4*. The parameters for the Zipf-type distribution (*Case 3*) are $a = 1$, $b = 10$, $s = 2$, and $Sc = 3$ of $N = 15$ and $N = 25$. The parameters of the time-varying distribution (*Case 4*) are Set $\omega = 1/6$, $\zeta = 0.1$, and $Sc = 3$ of $N = 15$ and $N = 25$.

Next, we proceed to demonstrate that Algorithm 1 (OTA) can achieve the desirable convergence of the regret and constraint violation in a variety of conditions. Here, we expand our study using three more cases that include a new nonuniform static case, a time-varying case, and an adversarial case. The aim of this latter, admittedly corner, case is to demonstrate that OTA succeeds in learning even when one explicitly tries to degrade the operation of the network (by altering every parameter to its extreme values). It is also important to stress that the comparison uses as a reference the benchmark solution $\boldsymbol{x}^\star$, which is the policy that one could design had they known the entire future. This benchmark is essentially the solution that some prior works propose, e.g., our own work [19], [27], yet they require having static and known conditions across all nodes and time. The convergence of OTA to this benchmark shows that, through learning, we can achieve the same result without having to rely on such strong assumptions.

In detail, we present the regret and constraint fit plots for *Sc1* and 3 in Fig. 4. It is clear that as the number of nodes increases, the convergence speed of the regret slows down, and the constraint fit shows no significant difference across scenarios. What is more, the parameters in *Case 2* are selected in an adversarial ping-pong fashion, which has no significant impact on the convergence of the regret and the constraint fit. Finally, for $N = 15$ and $N = 25$, we consider a Zipf-type distribution (*Case 3*) with parameters $a = 1$, $b = 10$, $s = 2$, and $Sc = 3$, as well as a time-varying distribution (*Case 4*) with parameters $\omega = 1/6$, $\zeta = 0.1$, and $Sc = 3$. In both cases, convergence of regret and constraint fit is maintained, as shown in Fig. 5. This verifies the robustness of the learning algorithm, which eventually achieves the above-studied cooperation gains without having access to the network, node, and task time-evolving parameters.

VII. CONCLUSION

The growing need to execute inference tasks using resource-constrained devices renders imperative the design of resource-prudent and performance-efficient solutions. Whereas the idea of offloading such tasks to the cloud(lets) has been heavily explored, a less studied idea is the collaborative execution of such tasks by the same devices that generate them. Multiplexing and combining dormant resources have been traditionally a scalable solution for systems with resource limitations. This article follows this path and proposes a scheme where a controller distributes the tasks generated by the devices among them, so as to maximize the aggregate inference accuracy while minimizing the induced delays and energy costs. Furthermore, the controller ensures that no device suffers energy consumption that is excessively higher than its neighbors—a minimal intervention toward fairness. To achieve this goal, we put forward a rigorous framework that relies on COCO, and we design a learning algorithm that offers optimality guarantees for a range of perturbation scenarios while being oblivious to information about the network state (e.g., channel conditions), node resources (e.g., computing capacity), and the task requirements. Hence, it has robust performance guarantees and practical value.

We believe that this study paves the road for interesting follow-up works. For instance, in some systems, the nodes operate with intermittent energy sources, and this energy (un)availability needs to be explicitly taken into account. In

other scenarios, the nodes might have small battery capacities that are replenished periodically (with unknown rates), and the learning framework needs to be extended to account for such cross-slot dependencies of the decisions. Finally, it is clear that in some types of applications (e.g., when the task rates are very small), the continuous approximation model of this work might induce nonnegligible rounding errors; this requires a different treatment of the problem, along the lines of works with discrete decision models, such as [62].

APPENDIX

A. Proof of Lemma 1

The proof is based on [56, Theorem 1], which is adapted with updated regularizers and constraint types for the studied problem. First, note that we replace the proximal variant of the quadratic regularizer in [56] with an entropic regularizer for the multisimplex constraint of $\mathbb{P}$. We bound the regret as

$$\begin{aligned}\mathcal{R}_T \leq B_T &\triangleq \frac{\sigma_{1:T}}{2} I \log I + \sum_{t=1}^{T} \frac{\delta_{t-1}\|\boldsymbol{g}_t(\boldsymbol{z}_t)\|^2}{2} \\ &+ \frac{2L_f}{\sigma}\sqrt{\sum_{t=1}^{T}\|\boldsymbol{c}_t + \boldsymbol{\lambda}_t^\top \nabla \boldsymbol{g}_t(\boldsymbol{x}_t)\|_\infty^2} \\ &\leq \left(\frac{\sigma I \log I}{2} + \frac{2L_f}{\sigma}\right)\sqrt{\sum_{t=1}^{T}\|\boldsymbol{c}_t + \boldsymbol{\lambda}_t^\top \nabla \boldsymbol{g}_t(\boldsymbol{x}_t)\|_\infty^2} \\ &+ \sum_{t=1}^{T}\frac{\delta_{t-1}\|\boldsymbol{g}_t(\boldsymbol{z}_t)\|^2}{2} \\ &\leq \left(\frac{\sigma I \log I}{2} + \frac{2L_f}{\sigma}\right)\left(\sum_{t=1}^{T}\|\boldsymbol{c}_t + \boldsymbol{\lambda}_t^\top \nabla \boldsymbol{g}_t(\boldsymbol{x}_t)\|\right) \\ &+ \sum_{t=1}^{T}\frac{\delta_{t-1}\|\boldsymbol{g}_t(\boldsymbol{z}_t)\|^2}{2} \\ &\leq \left(\frac{\sigma I \log I}{2} + \frac{2L_f}{\sigma}\right)\left(\sum_{t=1}^{T}\left(\|\boldsymbol{c}_t\| + \|\boldsymbol{\lambda}_t^\top\|\|\nabla \boldsymbol{g}_t(\boldsymbol{x}_t)\|\right)\right) \\ &+ \sum_{t=1}^{T}\frac{\delta_{t-1}\|\boldsymbol{g}_t(\boldsymbol{z}_t)\|^2}{2}. \end{aligned} \tag{21}$$

Then, we define $\boldsymbol{m}_t = \nabla \boldsymbol{g}_t(\boldsymbol{x}_t)^\top(\boldsymbol{z}_t - \boldsymbol{x}_t)$ and write

$$\begin{aligned}\mathcal{V}_T &= \|\sum_{t=1}^{T}\boldsymbol{g}_t(\boldsymbol{x}_t) + \boldsymbol{m}_t - \boldsymbol{m}_t\| \\ &\leq \|\sum_{t=1}^{T}\boldsymbol{g}_t(\boldsymbol{x}_t) + \boldsymbol{m}_t\| + \|-\boldsymbol{m}_{1:T}\| \\ &\leq \|\sum_{t=1}^{T}\boldsymbol{g}_t(\boldsymbol{z}_t)\| + \|-\boldsymbol{m}_{1:T}\| \\ &\leq \sqrt{\frac{2(B_T - \mathcal{R}_T)}{\delta_{T-1}}} + \sum_{t=1}^{T}\|\nabla \boldsymbol{g}_t(\boldsymbol{x}_t)\|_{(t)t}\|\boldsymbol{z}_t - \boldsymbol{x}_t\|_{(t^\star)} \\ &\leq \sqrt{\frac{2(B_T - \mathcal{R}_T)}{\delta_{T-1}}} + \frac{2L_g}{\sigma}\sqrt{h_{1:T}}. \end{aligned} \tag{22}$$

Now that we have the detailed expressions for $\mathcal{R}_T$ and $\mathcal{V}_T$, we proceed to show that their convergence rates are sublinear. We start with the dual update to obtain a useful bound

$$\begin{aligned}\boldsymbol{\lambda}_{t+1} &= \arg\max_{\boldsymbol{\lambda}\in\mathbb{R}_+^E}\sum_{\tau=1}^{T}\mathcal{L}_\tau(\boldsymbol{z}_\tau, \boldsymbol{\lambda}) \\ &= \arg\max_{\boldsymbol{\lambda}}\sum_{\tau=1}^{t}\left(r_t(\boldsymbol{z}_\tau) + \boldsymbol{c}_\tau^\top \boldsymbol{z}_\tau + \boldsymbol{\lambda}^\top \boldsymbol{g}_\tau(\boldsymbol{z}_\tau) - q_\tau(\boldsymbol{\lambda})\right) \\ &= \arg\max_{\boldsymbol{\lambda}}\sum_{\tau=1}^{t}\left(\boldsymbol{\lambda}^\top \boldsymbol{g}_\tau(\boldsymbol{z}_\tau) - \frac{\phi_\tau\|\boldsymbol{\lambda}\|^2}{2}\right) \Rightarrow \\ \|\boldsymbol{\lambda}_{t+1}\| &\overset{(i)}{\leq} \frac{\|\sum_{\tau=1}^{t}\boldsymbol{g}_\tau(\boldsymbol{z}_\tau)\|}{\phi_{1:t}} \end{aligned} \tag{23}$$

where, in (i), we simply used the KKT conditions to solve the RHS problem. Let us denote with k the growth rate of the dual vector, i.e., $\boldsymbol{\lambda}_t \leq \boldsymbol{\Lambda}_T = \mathcal{O}(T^k)$. Similarly, using (17), the following inequalities satisfy:

$$\begin{aligned}\sum_{t=1}^{T}\delta_{t-1}\|g_t(z_t)\|^2 &\leq \sum_{t=1}^{T}\frac{\delta\|g_t(z_t)\|^2}{\sqrt{\sum_{i=1}^{t}\|g_i(z_i)\|^2}} \\ &\leq 2\delta\sqrt{\sum_{t=1}^{T}\|g_t(z_t)\|^2} = \mathcal{O}\left(\sqrt{T}\right) \\ \sum_{t=1}^{T}\delta_{t-1}\|g_t(z_t)\|^2 &\leq \sum_{t=1}^{T}\frac{\delta\|g_t(z_t)\|^2}{t^\theta} \leq \frac{\delta G^2 T^{1-\theta}}{1-\theta} \\ &= \mathcal{O}\left(T^{1-\theta}\right). \end{aligned}$$

Hence, we obtain

$$\sum_{t=1}^{T}\delta_{t-1}\|g_t(z_t)\|^2 = \mathcal{O}(T^\eta), \quad \eta = \min\{1/2, 1-\theta\} \leq 1.$$

Then, we can proceed to write

$$\begin{aligned}\mathcal{R}_T \leq \left(\frac{\sigma I \log I}{2} + \frac{2L_f}{\sigma}\right)(T(F + \Lambda_T G)) \\ + \sum_{t=1}^{T}\frac{\delta_{t-1}\|\boldsymbol{g}_t(\boldsymbol{z}_t)\|^2}{2}. \end{aligned}$$

We can further tailor this analysis to the properties of $\mathbb{P}$ by replacing the gradient and constraint bounds, namely,

$$\begin{aligned}F = I\alpha_{\max}\beta_{\max} + E\alpha_{\max}\beta_{\max} + I\frac{\beta_{\max}s_{\max}}{C_{\min}} \\ + I\frac{\beta_{\max}p_{\max}}{H_{\min}} + Ie^x_{\max}s_{\max}\beta_{\max} + Ie^c_{\max}p_{\max}\beta_{\max}\end{aligned}$$

where the max-indexed symbols represent the maximum values of the respective system parameters. Similarly, for the constraint-related parameters, we can write

$$\begin{aligned}G = \sqrt{E}\left|e^x_{\max}s_{\max}\beta_{\max} - e^x_{\min}s_{\min}\beta_{\min}\right. \\ \left. + e^c_{\max}p_{\max}\beta_{\max} - e^c_{\max}p_{\min}\beta_{\min} - \varepsilon^{th}_{\min}\right|.\end{aligned}$$

Next, by its definition, the dual learning rate satisfies

$$\delta_T \leq \frac{\delta}{\sqrt{\sum_{t=1}^{T}\|\boldsymbol{g}_t(\boldsymbol{z}_t)\|^2}} = \mathcal{O}\left(T^{-1/2}\right), \quad \text{and}$$

$$\delta_T \le \frac{\delta}{T^\theta} = \mathcal{O}\left(T^{-\theta}\right).$$

Thus, it follows that

$$\delta_T = \mathcal{O}\left(T^\epsilon\right), \epsilon = \min\{-\theta, -1/2\} \le 0. \tag{24}$$

Similarly, using (17), the following inequalities satisfies:

$$\sum_{t=1}^{T} \delta_{t-1} \|g_t(z_t)\|^2 \le \sum_{t=1}^{T} \frac{\delta \|g_t(z_t)\|^2}{\sqrt{\sum_{i=1}^{t} \|g_i(z_i)\|^2}} \le 2\delta \sqrt{\sum_{t=1}^{T} \|g_t(z_t)\|^2} = \mathcal{O}\left(\sqrt{T}\right)$$

$$\sum_{t=1}^{T} \delta_{t-1} \|g_t(z_t)\|^2 \le \sum_{t=1}^{T} \frac{\delta \|g_t(z_t)\|^2}{t^\theta} \le \frac{\delta G^2 T^{1-\theta}}{1-\theta} = \mathcal{O}\left(T^{1-\theta}\right).$$

Hence, we obtain

$$\sum_{t=1}^{T} \delta_{t-1} \|g_t(z_t)\|^2 = \mathcal{O}\left(T^\eta\right), \quad \eta = \min\{1/2, 1-\theta\} \le 1$$

$$h_{1:T} = \sum_{t=1}^{T} \|c_t + \boldsymbol{\lambda}_t^\top \nabla \boldsymbol{g}_t(\boldsymbol{x}_t)\| \le T(F + \Lambda_T G) = \mathcal{O}\left(T^{1+k}\right).$$

Returning to the expression for the dual vector, we can write

$$\|\boldsymbol{\lambda}_{T+1}\| \le \delta_T \sqrt{2(B_T - \mathcal{R}_T)/\delta_{T-1}} \le \sqrt{2\delta_T (B_T - \mathcal{R}_T)}. \tag{25}$$

Replacing B_T in (25) with its definition from (21), we get

$$\|\boldsymbol{\lambda}_{T+1}\| = \mathcal{O}\left(\max\left\{T^{\frac{2\epsilon+k+1}{4}}, T^{\frac{\epsilon+\eta}{2}}, T^{\frac{\epsilon+1}{2}}\right\}\right)$$

where we uesd the worst case bound $-\mathcal{R}_T = \mathcal{O}(T)$. From (23), we observe that $k \le 1 + \epsilon \le 1$ and $n \le 1$ and obtain

$$\|\boldsymbol{\lambda}_{T+1}\| = \mathcal{O}\left(T^{\frac{\epsilon+1}{2}}\right) = \mathcal{O}\left(T^k\right).$$

This means that $k = (\epsilon+1)/2$, and we refine the bounds

$$h_{1:T} = \mathcal{O}\left(T^{\frac{\epsilon+3}{2}}\right)$$

$$\mathcal{R}_T \triangleq B_T = \mathcal{O}\left(\max\left\{T^{\frac{\epsilon+3}{4}}, T^\eta\right\}\right) = \mathcal{O}\left(T^{\frac{\epsilon+3}{4}}\right). \tag{26}$$

Using (24) and (26), and $-\mathcal{R}_T = \mathcal{O}(T)$, we obtain

$$\mathcal{V}_T = \mathcal{O}\left(\max\left\{T^{\frac{3(1-\epsilon)}{8}}, T^{\frac{\eta-\epsilon}{2}}, T^{\frac{1-\epsilon}{2}}, T^{\frac{3+\epsilon}{4}}\right\}\right) = \mathcal{O}\left(T^{\frac{1-\epsilon}{2}}\right).$$

Finally, we get the bounds

$$\mathcal{R}_T = \mathcal{O}\left(T^{\frac{5}{8}}\right), \mathcal{V}_T = \mathcal{O}\left(T^{\frac{3}{4}}\right), \quad \text{when } \theta < 1/2$$

$$\mathcal{R}_T = \mathcal{O}\left(T^{\frac{3-\theta}{4}}\right), \mathcal{V}_T = \mathcal{O}\left(T^{\frac{1+\theta}{2}}\right), \quad \text{when } \theta \ge 1/2.$$

B. Proof of Proposition 1

We formulate the problem of (18) in standard form

$$\min_{\boldsymbol{x}} \quad \sum_{\tau=1}^{t-1} \mathcal{L}_\tau(\boldsymbol{x}, \boldsymbol{\lambda}_\tau) \tag{27}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{I}_i} x_{ij} = 1 \quad \forall i \in \mathcal{I} \tag{28}$$

$$x_{ij} \ge 0, \quad \forall i \in \mathcal{I} \quad \forall j \in \mathcal{I}_i. \tag{29}$$

We can write $\sum_{\tau=1}^{t-1} \mathcal{L}_\tau(\boldsymbol{x}, \boldsymbol{\lambda}_\tau)$

$$= r_{1:t-1}(\boldsymbol{x}) + \boldsymbol{c}_{1:t-1}^\top \boldsymbol{x} + \sum_{\tau=1}^{t-1} \lambda_\tau^\top g_\tau(\boldsymbol{x}) - \sum_{\tau=1}^{t-1} q_\tau(\boldsymbol{\lambda}_\tau)$$

$$= r_{1:t-1}(\boldsymbol{x}) + \boldsymbol{\eta}_{(t-1)}^\top \boldsymbol{x} + \boldsymbol{\epsilon}_{t-1}^\top \boldsymbol{x} - \sum_{\tau=1}^{t-1} q_\tau(\boldsymbol{\lambda}_\tau)$$

$$= r_{1:t-1}(\boldsymbol{x}) - \boldsymbol{w}_{(t-1)}^\top \boldsymbol{x} - \sum_{\tau=1}^{t-1} q_\tau(\boldsymbol{\lambda}_\tau)$$

where we have defined $w_{ij(t-1)} = -\eta_{ij(t-1)} - \epsilon_{ij(t-1)}$, $\eta_{ij(t-1)} = \sum_{\tau=1}^{t-1} c_{ij\tau}$, and $\boldsymbol{\epsilon}_{t-1}^\top \boldsymbol{x} = \sum_{\tau=1}^{t-1} \lambda_\tau^\top g_\tau(\boldsymbol{x})$. To sum up, the problem that we aim to solve is

$$\mathbb{P}': \quad \min_{\boldsymbol{x}} \quad r_{1:t-1}(\boldsymbol{x}) - \boldsymbol{w}_{(t-1)}^\top \boldsymbol{x} \tag{30}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{I}_i} x_{ij} = 1 \quad \forall i \in \mathcal{I}, \tag{31}$$

$$x_{ij} \ge 0 \quad \forall i \in \mathcal{I} \quad \forall j \in \mathcal{I}_i. \tag{32}$$

We define the associated Lagrangian of $\mathbb{P}'$ as

$$L(\boldsymbol{x}, \boldsymbol{\psi}, \boldsymbol{\mu}) = r_{1:t-1}(\boldsymbol{x}) - \boldsymbol{w}_{(t-1)}^\top \boldsymbol{x} - \boldsymbol{\psi}^T \boldsymbol{x} + \sum_{i \in \mathcal{I}} \mu_i \left(\sum_{j \in \mathcal{I}_i} x_{ij} - 1\right),$$

where $\boldsymbol{\psi} \in \mathbb{R}^{|\mathcal{E}|}$ and $\mu \in \mathbb{R}^{\mathcal{I}}$ are the Lagrange multipliers. The KKT conditions are given as follows.

1) Complementary slackness

$$\psi_{ij} x_{ij} = 0, \quad \forall i \in \mathcal{I}, \quad \forall j \in \mathcal{I}_i. \tag{33}$$

2) Stationarity

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\psi}, \boldsymbol{\mu}) = 0. \tag{34}$$

Specifically,

$$\text{if} \quad x_{ij} > 0 : \frac{\sigma_{1:t-1}}{2}\left(\log x_{ij} + 1\right) - w_{ij(t-1)} - \psi_{ij} + \mu_i = 0 \tag{35}$$

$$\text{if} \quad x_{ij} = 0 : -w_{ij(t-1)} - \psi_{ij} + \mu_i = 0. \tag{36}$$

3) Primal feasibility

$$x_{ij} \ge 0, \quad \sum_{j \in \mathcal{I}_i} x_{ij} = 1 \quad \forall i \in \mathcal{I} \quad \forall j \in \mathcal{I}_i. \tag{37}$$

4) Dual feasibility

$$\psi_{ij} \ge 0 \quad \forall i \in \mathcal{I} \quad \forall j \in \mathcal{I}_i. \tag{38}$$

In the first case, set $\boldsymbol{\psi} = 0$, and then, x_{ij} be nonzero. If x_{ij} satisfies the equation in Proposition 1, then it holds

$$\sum_{j \in \mathcal{I}_i} \frac{\exp\left(2w_{ij(t-1)}/\sigma_{1:t-1}\right)}{\sum_{j \in \mathcal{I}_i} \exp\left(2w_{ij(t-1)}/\sigma_{1:t-1}\right)} = 1 \quad \forall i \in \mathcal{I} \quad \forall j \in \mathcal{I}_i. \tag{39}$$

At the same time, we get $\forall i \in \mathcal{I}$

$$\mu_i = \frac{\sigma_{1:t-1}}{2} \log\left(\sum_{j \in \mathcal{I}_i} \exp\left(\frac{2w_{ij(t-1)}}{\sigma_{1:t-1}}\right)\right) - \frac{\sigma_{1:t-1}}{2}. \tag{40}$$

Therefore, this solution satisfies all KKT conditions. In the second case, set $\boldsymbol{\psi} = 0$, and then, x_{ij} can be zero or nonzero. We can know that for a certain i, if there exists $x_{ik} = 0$, $x_{ij} = \left(\exp(2w_{ij(t-1)}/\sigma_{1:t-1})/\sum_{j\in\mathcal{I}_i^{--}}\exp(2w_{ij(t-1)}/\sigma_{1:t-1})\right)$, $k \in \mathcal{I}_i$, and $j \in \mathcal{I}_i$, then μ_i will have two values, which is impossible ($\mathcal{I}_i^{--}$ contains the one-hop neighbors, in which node i transmits data). In the third case, set $\boldsymbol{\psi} > \mathbf{0}$, and then, $\boldsymbol{x} = \mathbf{0}$. Thus, this solution does not satisfy the primal feasibility condition. In the fourth case, set $\psi_{ij} = 0$ and $\psi_{ik} > 0$, and then, it should hold $x_{ij} = \left(\exp(2w_{ij(t-1)}/\sigma_{1:t-1})/\sum_{j\in\mathcal{I}_i^{--}}\exp(2w_{ij(t-1)}/\sigma_{1:t-1})\right)$ and $x_{ik} = 0$. Like Case 2, two values of μ_i will be obtained, so it is impossible.

REFERENCES

[1] *Research Outlook Toward 2030: 6G–Connecting a Cyber-Physical World*, Ericsson, Stockholm, Sweden, 2022.

[2] European Commission. (2025). *The Next Generation Internet of Things*. Accessed: Aug. 26, 2025. [Online]. Available: https://digital-strategy.ec.europa.eu/en/policies/next-generation-internet-things

[3] O. Vermesan and J. Bacquet, *Cognitive Hyperconnected Digital Transformation: Internet of Things Intelligence Evolution*, 1st ed., New York, NY, USA: River Publishers, 2017.

[4] E. Siow, T. Tiropanis, and W. Hall, "Analytics for the Internet of Things: A survey," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 1–36, 2018.

[5] S. Nayak, R. Patgiri, L. Waikhom, and A. Ahmed, "A review on edge analytics: Issues, challenges, opportunities, promises, future directions, and applications," *Digit. Commun. Netw.*, vol. 10, no. 3, pp. 783–804, Jun. 2024.

[6] S. Shukla, M. F. Hassan, M. K. Khan, L. T. Jung, and A. Awang, "An analytical model to minimize the latency in healthcare Internet-of-Things in fog computing environment," *PLoS ONE*, vol. 14, no. 11, Nov. 2019, Art. no. e0224934.

[7] N. Ng et al., "Collaborative inference in resource-constrained edge networks: Challenges and opportunities," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2024, pp. 1–6.

[8] C. Campolo, A. Iera, and A. Molinaro, "Network for distributed intelligence: A survey and future perspectives," *IEEE Access*, vol. 11, pp. 52840–52861, 2023.

[9] Z. Cheng, X. Zhu, and S. Gong, "Surveillance face recognition challenge," 2018, *arXiv:1804.09691*.

[10] U. Gawande, K. Hajari, and Y. Golhar, "Novel person detection and suspicious activity recognition using enhanced YOLOv5 and motion feature map," *Artif. Intell. Rev.*, vol. 57, no. 2, p. 16, Jan. 2024.

[11] J. Tirana, D. Tsigkari, G. Iosifidis, and D. Chatzopoulos, "Workflow optimization for parallel split learning," in *Proc. IEEE Conf. Comput. Commun.*, May 2024, pp. 1331–1340.

[12] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. 20th Int. Conf. Mach. Learn.*, 2003, pp. 928–936.

[13] E. Hazan, "Introduction to online convex optimization," *Found. Trends Optim.*, vol. 2, nos. 3–4, pp. 157–325, 2016.

[14] G. S. Paschos, A. Destounis, L. Vigneri, and G. Iosifidis, "Learning to cache with no regrets," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2019, pp. 235–243.

[15] H. Yu and M. J. Neely, "Learning-aided optimization for energy-harvesting devices with outdated state information," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1501–1514, Aug. 2019.

[16] A. Marcastel, E. V. Belmega, P. Mertikopoulos, and I. Fijalkow, "Online power optimization in feedback-limited, dynamic and unpredictable IoT networks," *IEEE Trans. Signal Process.*, vol. 67, no. 11, pp. 2987–3000, Jun. 2019.

[17] D. Q. Vu, K. Antonakopoulos, and P. Mertikopoulos, "Fast routing under uncertainty: Adaptive learning in congestion games via exponential weights," in *Proc. NeurIPS*, vol. 34, 2021, pp. 14708–14720.

[18] Q. Zhang et al., "Intelligence inference on IoT devices," in *Learning Techniques for the Internet of Things*. Cham, Switzerland: Springer, 2023, pp. 171–195.

[19] A. Galanopoulos, T. Salonidis, and G. Iosifidis, "Cooperative edge computing of data analytics for the Internet of Things," *IEEE Trans. Cognit. Commun. Netw.*, vol. 6, no. 4, pp. 1166–1179, Dec. 2020.

[20] L. Nkenyereye, K.-J. Baeg, and W.-Y. Chung, "Deep reinforcement learning for containerized edge intelligence inference request processing in IoT edge computing," *IEEE Trans. Services Comput.*, vol. 16, no. 6, pp. 4328–4344, Nov. 2023.

[21] R. Hadidi, J. Cao, M. S. Ryoo, and H. Kim, "Toward collaborative inferencing of deep neural networks on Internet-of-Things devices," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4950–4960, Jun. 2020.

[22] A. P. Behera et al., "Exploring the boundaries of on-device inference: When tiny falls short, go hierarchical," *IEEE Internet Things J.*, vol. 12, no. 18, pp. 37456–37470, Sep. 2025.

[23] N. Shlezinger and I. V. Bajić, "Collaborative inference for AI-empowered IoT devices," *IEEE Internet Things Mag.*, vol. 5, no. 4, pp. 92–98, Dec. 2022.

[24] M. Zhang, X. Shen, J. Cao, Z. Cui, and S. Jiang, "EdgeShard: Efficient LLM inference via collaborative edge computing," *IEEE Internet Things J.*, vol. 12, no. 10, pp. 13119–13131, May 2025.

[25] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge computing framework for cooperative video processing in multimedia IoT systems," *IEEE Trans. Multimedia*, vol. 20, no. 5, pp. 1126–1139, May 2018.

[26] X. Li, L. Zhao, K. Yu, M. Aloqaily, and Y. Jararweh, "A cooperative resource allocation model for IoT applications in mobile edge computing," *Comput. Commun.*, vol. 173, pp. 183–191, May 2021.

[27] A. Galanopoulos, G. Iosifidis, and T. Salonidis, "Optimizing data analytics in energy constrained IoT networks," in *Proc. 16th Int. Symp. Model. Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, May 2018, pp. 1–8.

[28] S. Yun, W. Choi, and I.-M. Kim, "Cooperative inference of DNNs for delay- and memory-constrained wireless IoT systems," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 16113–16127, Sep. 2022.

[29] Z. Jiang et al., "CoEdge: A cooperative edge system for distributed real-time deep learning tasks," in *Proc. ACM/IEEE IPSN*, May 2023, pp. 53–66.

[30] R. Li, T. Ouyang, L. Zeng, G. Liao, Z. Zhou, and X. Chen, "Online optimization of DNN inference network utility in collaborative edge computing," *IEEE/ACM Trans. Netw.*, vol. 32, no. 5, pp. 4414–4426, Oct. 2024.

[31] M. Kotys, Y. Zhang, C. Q. Wu, and A. Hou, "Optimizing task allocation for DNN inference on edge devices," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2025, pp. 458–462.

[32] J. Xie, Q. Jia, and Y. Chen, "Energy-efficient intelligence sharing in intelligence networking-empowered edge computing: A deep reinforcement learning approach," *IEEE Access*, vol. 12, pp. 141639–141652, 2024.

[33] M. Malka, E. Farhan, H. Morgenstern, and N. Shlezinger, "Decentralized low-latency collaborative inference via ensembles on the edge," *IEEE Trans. Wireless Commun.*, vol. 24, no. 1, pp. 598–614, Jan. 2025.

[34] Z. Zhou, J. Xie, M. Huang, T. Ouyang, F. Liu, and X. Chen, "Towards federated inference: An online model ensemble framework for cooperative edge AI," in *Proc. IEEE INFOCOM*, May 2025, pp. 1–10.

[35] A. Ben Sada, A. Khelloufi, A. Naouri, H. Ning, and S. Dhelim, "Selective task offloading for maximum inference accuracy and energy efficient real-time IoT sensing systems," 2024, *arXiv:2402.16904*.

[36] A. Galanopoulos, J. A. Ayala-Romero, D. J. Leith, and G. Iosifidis, "AutoML for video analytics with edge computing," in *Proc. IEEE Conf. Comput. Commun.*, May 2021, pp. 1–10.

[37] Y. Liu, Y. Mao, Z. Liu, F. Ye, and Y. Yang, "Joint task offloading and resource allocation in heterogeneous edge environments," *IEEE Trans. Mobile Comput.*, vol. 23, no. 6, pp. 1–10, Jun. 2024.

[38] Y. Bai, L. Chen, M. Abdel-Mottaleb, and J. Xu, "Automated ensemble for deep learning inference on edge computing platforms," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4202–4213, Mar. 2022.

[39] K. Qu et al., "Stochastic cumulative DNN inference with RL-aided adaptive IoT device-edge collaboration," *IEEE Internet Things J.*, vol. 10, no. 20, pp. 18000–18015, Oct. 2023.

[40] Y. Xu, T. Mohammed, M. Di Francesco, and C. Fischione, "Distributed assignment with load balancing for DNN inference at the edge," *IEEE Internet Things J.*, vol. 10, no. 2, pp. 1053–1065, Jan. 2023.

[41] X. Xu, K. Yan, S. Han, B. Wang, X. Tao, and P. Zhang, "Learning-based edge-device collaborative DNN inference in IoVT networks," *IEEE Internet Things J.*, vol. 11, no. 5, pp. 7989–8004, Mar. 2024.

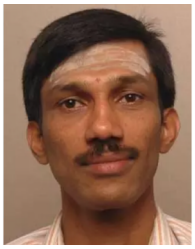
[42] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3887–3901, Dec. 2016.

[43] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive D2D collaboration for energy-efficient mobile edge computing," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 64–71, Aug. 2017.

[44] D. O'Keeffe, T. Salonidis, and P. Pietzuch, "Frontier: Resilient edge processing for the Internet of Things," *Proc. VLDB Endowment*, vol. 11, no. 3, pp. 1179–1191, 2018.
[45] S. Fan, T. Salonidis, and B. Lee, "Swing: Swarm computing for mobile sensing," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 1107–1117.
[46] A. B. Ameur, A. Araldo, and F. Bronzino, "On the deployability of augmented reality using embedded edge devices," in *Proc. IEEE 18th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2021, pp. 1–6.
[47] A. Fresa and J. P. Champati, "Offloading algorithms for maximizing inference accuracy on edge device in an edge intelligence system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 7, pp. 2025–2039, Jul. 2023.
[48] R. W. L. Coutinho and A. Boukerche, "Modeling and performance evaluation of collaborative IoT cross-camera video analytics," in *Proc. IEEE Int. Conf. Commun.*, May 2023, pp. 1804–1809.
[49] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "CoEdge: Cooperative DNN inference with adaptive workload partitioning over heterogeneous edge devices," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 595–608, Apr. 2021.
[50] M. Li, G. Iosifidis, and R. V. Prasad, "Cooperative streaming inferences in IoT networks," in *Proc. GLOBECOM - IEEE Global Commun. Conf.*, Dec. 2024, pp. 4824–4829.
[51] N. Mhaisen, A. Sinha, G. Paschos, and G. Iosifidis, "Optimistic no-regret algorithms for discrete caching," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 6, no. 3, pp. 1–28, 2022.
[52] F. Aslan, G. Iosifidis, J. A. Romero, A. Garcia-Saavedra, and X. Costa-Perez, "Fair resource allocation in virtualized O-RAN platforms," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 52, no. 1, pp. 59–60, 2024.
[53] P. Choi, D. Ham, Y. Kim, and J. Kwak, "VisionScaling: Dynamic deep learning model and resource scaling in mobile vision applications," *IEEE Internet Things J.*, vol. 11, no. 9, pp. 15523–15539, May 2024.
[54] T. Chen, S. Barbarossa, X. Wang, G. B. Giannakis, and Z.-L. Zhang, "Learning and management for Internet of Things: Accounting for adaptivity and scalability," *Proc. IEEE*, vol. 107, no. 4, pp. 778–796, Apr. 2019.
[55] 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.
[56] D. Anderson, G. Iosifidis, and D. J. Leith, "Lazy Lagrangians for optimistic learning with budget constraints," *IEEE/ACM Trans. Netw.*, vol. 31, no. 5, pp. 1935–1949, Oct. 2023.
[57] S. K. Saha, P. Deshpande, P. P. Inamdar, R. K. Sheshadri, and D. Koutsonikolas, "Power-throughput tradeoffs of 802.11n/AC in smartphones," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2015, pp. 100–108.
[58] Y. Zhong et al., "Unequal-training for deep face recognition with long-tailed noisy data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7804–7813.
[59] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–144, 2006.
[60] S. Mannor, J. N. Tsitsiklis, and J. Y. Yu, "Online learning with sample path constraints," *J. Mach. Learn. Res.*, vol. 10, no. 3, pp. 1467–1494, 2009.
[61] H. B. McMahan, "A survey of algorithms and analysis for adaptive online learning," *J. Mach. Learn. Res.*, vol. 18, no. 90, pp. 1–50, 2017.
[62] A. Galanopoulos, G. Iosifidis, T. Salonidis, and D. J. Leith, "Selective edge computing for mobile analytics," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 3, pp. 3090–3104, Sep. 2022.
[63] A. Nefian. *Georgia Tech Face Database*. Accessed: Mar. 28, 2025. [Online]. Available: https://https://www.anefian.com/research/face_reco.htm

Mengyuan Li received the B.S. and M.E. degrees from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2019 and 2022, respectively. She is currently pursuing the Ph.D. degree with Delft University of Technology, Delft, The Netherlands.

Her research interests include edge computing, network optimization, and resource management.



R. Venkatesha Prasad is an Associate Professor at the Networked Systems Group, TU Delft, Delft, The Netherlands. He has supervised 21 Ph.D. students and 63 M.Sc. students. He has over 300 publications in peer-reviewed international journals and conferences, standards, and book chapters. His research interests are in the Tactile Internet, IoT, and 60-GHz mmWave networks.

Dr. Prasad is an Associate Editor on the Editorial Board of IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING, and IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING. He was the Vice-Chair of the IEEE Tactile Internet standardization workgroup and is now a Mentor. For more information, please refer to https://st.ewi.tudelft.nl/rvprasad/

George Iosifidis received the Diploma degree in electronics and telecommunications engineering from Greek Air Force Academy, Acharnes, Greece, in 2000, and the Ph.D. degree from the University of Thessaly, Volos, Greece, in 2012.

He was an Assistant Professor with Trinity College Dublin, Dublin, Ireland, from 2016 to 2020. He is an Associate Professor with Delft University of Technology, Delft, The Netherlands. His research interests lie in the broad area of network optimization and economics.