Dynamic Scheduling for IoT Analytics at the Edge

Apostolos Galanopoulos*, Víctor Valls†, Douglas J. Leith*, George Iosifidis*
*School of Computer Science and Statistics, Trinity College Dublin
†Department of Electrical Engineering and Institute for Network Science, Yale University

***Abstract*—We propose an online policy that schedules the transmission and processing of data analytic tasks in an Internet of Things (IoT) network. The tasks are executed with different precision at the (possibly heterogeneous) nodes; the network is subject to link bandwidth and node processing capacity changes; and the task requests vary following unknown statistics. For this general IoT scenario, we formulate a resource allocation problem towards maximizing the aggregate tasks precision, and design a dynamic solution policy by combining the Frank-Wolfe and dual subgradient algorithms. Our policy (FWDS) is guaranteed to converge within bounded distance from the optimal solution, while ensuring interference-free transmissions, and being oblivious to network and/or traffic load changes. We use a wireless testbed and a state-of-the-art face recognition application to implement FWDS and compare it with static or dynamic (maxweight-type) competitor policies. Our findings verify that FWDS pushes the envelope of network control algorithms by handling time-varying objectives and possibly non-i.i.d. network/load statistics, with smaller complexity than its competitors.**

***Index Terms*—Frank-Wolfe Algorithm, Internet of Things, Data Analytics, Network Optimization**

I. Introduction

The IoT era has spurred a series of new data analytics services that often need to be executed in real time [1], e.g face/object recognition, traffic monitoring, video stream analytics etc [2]. These services can be executed at, or near the IoT nodes that generate the data to satisfy real-time performance. However, many IoT nodes cannot handle the load they create [3] and need assistance from other network devices to process their data. Indeed, modern networks utilize edge devices, like e.g. Next Unit of Computing (NUC) nodes [4], that can execute analytics faster and with higher precision. This node heterogeneity allows modern networks to keep up to the demanding requirements of these services, by sending the generated analytics tasks to the network nodes that are best fitted to execute them.

Exploiting such devices however, requires making fast scheduling decisions, i.e. which network links will be activated and which nodes should execute each task. Finding an efficient scheduling policy for such a heterogeneous network is not an easy task. The nodes produce a time-varying, often non-iid traffic load, since the latter is often event-driven, eg. a motion-sensor/camera node that starts recording upon detecting movement. Moreover, data analytics services are unique, in the way that their performance metric is also dynamic, e.g. the recognition accuracy of an object recognizer varies over time, as the classifier can be presented with more samples from other external sources.

In light of these new intricacies, we revisit the classic problem of scheduling both task transmissions, i.e. link activation, and task processing at the heterogeneous IoT nodes. Face recognition tasks are generated throughout the network, and the goal is to formulate a dynamic scheduling policy that respects interference constraints and optimizes the service's aggregate performance, considering the network's transmission and processing capacities. Such a solution can be applied to many existing services, e.g. [5], [6], that target computing at the edge for IoT nodes and improve their performance.

Unlike other state-of-the-art solutions based on max-weight scheduling like e.g. the drift-plus-penalty (DPP) algorithm [7], our solution is designed to adapt not only to varying traffic load, but also to a varying objective function, which is ideally suited for analytics. Furthermore, our algorithm (FWDS) is designed towards having robust complexity per schedule evaluation, since it only solves a linear program in each iteration, even if the objective function is convex. This is achieved by using the Frank-Wolfe (FW) algorithm [8], and effectively linearizing the objective function, yielding schedules that are feasible, i.e they do not violate interference constraints, and are efficiently evaluated. To deal with the rest of networking and processing capacity constraints, we employ a dual method that relaxes them, and use the FW algorithm for the update of the primal variables (scheduling decisions). In contrast, max-weight based solutions either need to solve a (usually convex) admission control problem before max-weight scheduling [9], or assume a-priori knowledge of the set of interference-free schedules [10]–[13].

Our contributions are the following:

- We formulate an optimization problem for maximizing the execution accuracy of analytics in an IoT network. Our model has a general structure, allowing different tasks and node types of various capabilities.
- We propose a primal-dual method that elicits the system's unknown parameters and relaxes the problem's elastic constraints. We use the FW update in order to acquire a schedule that can be directly implemented at each iteration.
- We prove the convergence bounds of our algorithm to the optimal solution and use simulations to demonstrate its convergence rate. We compare its robustness in terms of computational complexity and performance, to other standard solutions and highlight its advantageous properties.

Paper Organization. In Sec. II we present the system model. In Sec. III we describe the relation between the static and dynamic problems, while Sec. IV presents our proposed algorithm and our main theorem about its performance. In Sec. V we discuss the related work and Sec. VI presents our testbed experiments and evaluation of our algorithm. Finally Sec. VII concludes the paper, followed by the Appendix, which includes the paper's proofs.

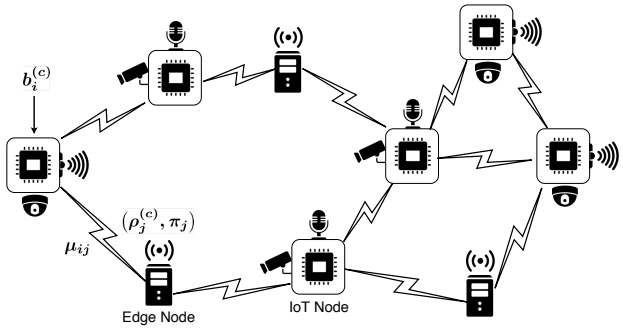


Fig. 1: An IoT network with heterogeneous nodes running various types of analytic tasks with different accuracy.

II. Model and Problem Setup

A. Network Model

We consider a wireless IoT network that connects a set $\mathcal{N}=\{i_1, i_2 \ldots, i_N\}$ of possibly heterogeneous nodes through a set $\mathcal{E} = \{(i,j) \mid i,j \in \mathcal{N}\}$ of E links, see Fig. 1. The system operation is time-slotted and w.l.o.g. we assume that slots have unit duration. The nodes communicate through one or more hops to jointly execute a set $\mathcal{C}=\{c_1, c_2, \ldots, c_C\}$ of data analytic tasks. For example, c_1 may correspond to image classification, c_2 to filtering data collected by sensors, etc. A task can be executed at the nodes which generate the data or elsewhere in the network. This decision depends on the computing and network resources, and the performance, i.e., the accuracy with which each task is performed at each node.

Each node i injects $b_{i,t}^{(c)} \geq 0$ data of task (or, commodity) c into the network in slot t, following a stochastic process $\{b_{i,t}^{(c)}\}_{t=1}^{\infty}$ with average value of $b_i^{(c)}$ bits/sec. The capacity of each link (i,j) may vary over time, $\{\mu_{ij,t}\}_{t=1}^{\infty}$, with averabe value μ_{ij} bits/sec. Similarly, every node i has a possibly time-varying processing capacity $\{\pi_{i,t}\}_{t=1}^{\infty}$ cycles/sec, and π_i is its long-term average. These random variables are uniformly upper-bounded by b_{max}, μ_{max} and π_{max}, respectively. Finally, parameter $\rho_i^{(c)}$ is the processing load of i, in cycles/bit, for tasks of type c. We consider the general case where $\{\rho_i^{(c)}\}_{i,c}$ may vary across nodes since they might have different hardware or analytic algorithms.

B. Variables and Constraints

We begin by describing the static operation of the system. Variables $y_{ij}^{(c)}, z_i^{(c)} \geq 0$ denote the *average* rate (bits/sec) at which commodity $c \in \mathcal{C}$ is transmitted over link (i,j) and processed at node i, respectively. These should satisfy:

$$b_i^{(c)} + \sum_{j \in \mathcal{N}_i} y_{ji}^{(c)} = \sum_{j \in \mathcal{N}_i} y_{ij}^{(c)} + z_i^{(c)}, \quad i \in \mathcal{N}, c \in \mathcal{C}, \quad (1)$$

where $\mathcal{N}_i = \{j \in \mathcal{N} \mid (i,j) \in \mathcal{E}\}$ are the one-hop neighbors of node i. Eq. (1) ensures that the incoming and locally-generated data are equal to the outgoing and locally-processed[1] data. Also, the data routed over each link should not exceed its capacity:

$$\sum_{c \in \mathcal{C}} y_{ij}^{(c)} \leq \mu_{ij}, \quad (i,j) \in \mathcal{E}, \quad (2)$$

and the processing is constrained by the computing capacity:

$$\sum_{c \in \mathcal{C}} \rho_i^{(c)} z_i^{(c)} \leq \pi_i, \quad i \in \mathcal{N}. \quad (3)$$

The transmissions are subject to interference, which is captured using the 2-hop interference model [14]. Hence, if link (i,j) is active, no neighbor of i can receive data, and no neighbor of j can transmit. This requirement leads to the necessary and sufficient conditions for a transmission policy to be implementable [15]:

$$\sum_{c \in \mathcal{C}} \frac{y_{ij}^{(c)}}{\mu_{max}} + \sum_{c \in \mathcal{C}} \sum_{(k,l) \in I(i,j)} \frac{y_{kl}^{(c)}}{\mu_{max}} \leq 1, \ \forall (i,j) \in \mathcal{E}, \quad (4)$$

which captures the fraction of time that link (i,j) can be active with respect to its interfering links:

$$I(i,j) = \big\{(m,k), (l,m), \ k \in \mathcal{N}_i, l \in \mathcal{N}_j\big\}.$$

Note that the interference sets $I(i,j), \forall (i,j) \in \mathcal{E}$ are assumed fixed, but we do allow changes in link capacities over time.

C. Network Control Problem & Challenges

Let $U_i^{(c)}(z_i^{(c)})$ be a utility function that expresses the reward derived from processing tasks $c \in \mathcal{C}$ at node i. For example, this function quantifies the benefits from making a successful image classification, or prediction using sensor measurements. It is a continuous and increasing function of $z_i^{(c)}$, and modulated by parameters $w_i^{(c)}$ which capture the performance offered by each node. In practice, it is unknown and expresses the average reward of process $\{w_{i,t}^{(c)}\}_{t=1}^{\infty}$. We select $U_i^{(c)}(z_i^{(c)}) = w_i^{(c)} z_i^{(c)}$, but our framework can be readily applied to other functions. We define the *Transmission and Computation Rate Allocation* problem:

$$\text{TCRA}: \quad \underset{\{y_{ij}^{(c)}, z_i^{(c)}\} \geq 0}{\text{maximize}} \quad \sum_{i \in \mathcal{N}} \sum_{c \in \mathcal{C}} U_i^{(c)}(z_i^{(c)})$$
$$\text{subject to}: \quad (1) - (4)$$

This is a convex program that, in theory, could be solved with off-the-shelf solvers. However, in most practical systems this is not possible for the following reasons:

- (R1): The task requests $\{b_{i,t}^{(c)}\}$, link capacities $\{\mu_{ij,t}\}$, and rewards $\{w_{i,t}^{(c)}\}$ are time-varying, with unknown mean values.
- (R2): Its solution is not directly implementable, because it would create collisions between interfering nodes, i.e., it is optimal only on average.

Let us elaborate on (R2). While (4) captures the *time-average* interference coupling, in each slot the network can support only one of the eligible *link-processing schedules*, i.e.:

$$S = \Big\{ \big(y_{ij}^{(c)}, z_i^{(c)}\big) \mid \mathbb{I}(\Psi_{ij}) + \sum_{(k,l) \in I(i,j)} \mathbb{I}(\Psi_{kl}) \leq 1 \Big\},$$

where $\Psi_{ij} = \sum_c y_{ij}^{(c)}$, and $\mathbb{I}(x) = 1$ if $x > 0$ and 0 otherwise. The optimal solution of TCRA does not necessarily belong to S, but the elements of S satisfy (4), namely:

[1] Processing is technically equivalent to "extracting" data from the network, as we obtain a certain utility and terminate their routing.

Proposition 1 (Link-Processing schedules): Any average schedule that satisfies (4) belongs to conv(S).

This means that we can select a series of schedules, the convex combination of which approximates the optimal solution of TCRA. The caveat however is that finding this "time-sharing" is a computationally cumbersome problem (see Sec. VI) and presumes knowing the system parameters.

We present next a practical *online* algorithm that makes *implementable* scheduling decisions, i.e., the selected schedules belong to S, and converges to the optimal TCRA solution without knowing the system statistics.

III. Reformulation and Dynamic Problem

We first reformulate the problem to streamline its presentation, explain technically how (R1)-(R2) hinder its solution, and introduce the t-slot problems that we will be using to solve TCRA dynamically. We use superscripts "$\circ$" and "$*$" for the optimal solution of the static and t-slot problems, respectively.

A. Preliminaries

We first define the bounded polytopes:

$$Y=\big\{y_{ij}^{(c)}\in[0,\mu_{max}]\mid (4),\ c\in\mathcal{C},\ (i,j)\in\mathcal{E}\big\},\ \text{and}$$
$$Z=\big\{z_{i}^{(c)}\in[0,\pi_{max}/\rho_{i}^{(c)}],\ c\in\mathcal{C},\ i\in\mathcal{N}\big\}.$$

and express our variables in a single set $X=Y\times Z\subseteq\mathbf{R}_{+}^{e}$, with $e=C^2NE$. Vector $x_t=(y_{ij,t}^{(c)},z_{i,t}^{(c)})$ is the amount of transmitted and processed data in slot t. We assume that TCRA has a finite optimal solution[2] x° (*Assumption 1*) which is a mild assumption as data can be dropped locally at the nodes that generate them, with zero utility.

We write constraints (1)-(3) in form[3] $Ax+\delta\leq 0$, where $A\in\mathbf{R}^{m\times e}$, $\delta\in\mathbf{R}^m$, $m=(CN+E+N)$; and define:

$$f(x)=-\sum_{i\in N}\sum_{c\in C}U_i^{(c)}(z_i^{(c)})=-w^\top x,$$

where we used vector notation to represent the sum for all nodes and commodities. We wish to stress that our framework can handle also general convex functions (as it will be evident from the analysis below), and hence one can use, for instance, $f(x)=-\log(w^\top x)$ to enforce some type of fairness or load-balancing. We can now rewrite TCRA as:

$$\texttt{Primal:}\quad \underset{x\in X}{\text{minimize}}\quad f(x)\quad \text{s.t.}\quad g(x)=Ax+\delta\preceq 0,$$

and the respective dual (concave) problem is:

$$\texttt{Dual:}\quad \underset{\lambda\succeq 0}{\text{maximize}}\quad h(\lambda)\triangleq\min_{x\in X}L(x,\lambda),$$

where $\lambda\in\mathbf{R}_{+}^{m}$ are the dual variables and $L(x,\lambda)=f(x)+\lambda^\top(Ax+\delta)$ is the Lagrangian. This problem can be solved with the subgradient method, which consists of the update:

$$\lambda_{t+1}=\big[\lambda_t+\alpha h'(\lambda_t)\big]^{+},\qquad (5)$$

where $\alpha>0$ is the step size and $h'(\lambda)$ is a subgradient in the subdifferential $\partial h(\lambda_t)$ of h at λ_t, that is given by:

$$x_t\in\arg\min_{x\in X}\big\{f(x)+\lambda_t^\top g(x)\big\}.\qquad (6)$$

One could use (5)-(6) to gradually approach the optimal TCRA solution $f(x^\circ)$; or even apply the obtained $\{x_t\}_t$ as they are generated in each iteration, see [16]. However, due to (R1)-(R2) either approach is not possible here, as we have time-varying parameters and (6) might yield $x_t\notin S$.

B. The t-slot Problem

Our strategy is to employ the t-slot version of TCRA that uses only information that is available up to t. In detail, we define the functions $f_t(x)=-w_t^\top x,\forall t$, where we assume that the sequence of observed precisions w_t converges to the final precision parameters of TCRA, i.e., $\lim_{t\to\infty}w_{i,t}^{(c)}=w_i^{(c)}\forall i,c$, (*Assumption 2*). This is because, as the nodes collect more data, they eventually achieve their maximum possible analytic performance.[4] Also, the changes across functions are bounded, i.e., $|f_{t+1}(x)-f_t(x)|\leq\sigma_f,\ \forall t,x$. Similarly, we define:

$$g_t(x)=Ax+\bar{\delta}_t,\ \text{ with }\ \bar{\delta}_t=\sum_{\tau=1}^{t}\delta_\tau/t$$

where the perturbations (link/node capacities, and arrivals) are assumed to converge, i.e. $\lim_{t\to\infty}\bar{\delta}_t=\delta$ (*Assumption 3*). Then, we can define the t-slot Lagrangian and dual problem:

$$\underset{\lambda_t\succeq 0}{\text{maximize}}\ h_t(\lambda_t)\triangleq\min_{x\in X}L_t(x,\lambda_t)=\min_{x\in X}\big\{f_t(x)+\lambda_t^\top g_t(x)\big\}.$$

In the sequel, we use these t-slot problems and functions to create a sequence of implementable decisions $\{s_t\}_t\in\mathcal{S}$ that ensure near optimal (average) performance, i.e.,

$$\frac{1}{t}\sum_{\tau=1}^{t}f_\tau(s_\tau)\longrightarrow f(x^\circ)\quad\text{and}\quad\frac{1}{t}\sum_{\tau=1}^{t}g_\tau(s_\tau)\longrightarrow g(x^\circ),$$

under *Assumptions 1-3*, that hold in most practical systems.

IV. Online Optimization Framework

We propose to combine the Frank-Wolfe (FW) algorithm [8] with the approximate dual subgradient algorithm [17]. FW yields implementable actions by solving a linear program (LP), and the dual subgradient algorithm handles the perturbed constraints. We synthesize these methods via the use of "errors" at the time to compute subgradients of the Lagrangian.

A. The Building Algorithmic Blocks

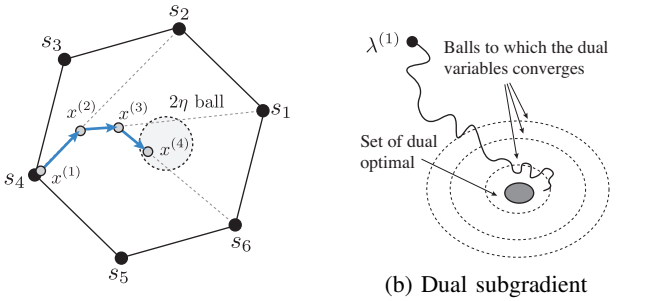
[2]This in practice means that the rates $b_i^{(c)}$ are such that they lie in the capacity region of the system; but one can modify slightly eq. (1) by introducing a variable that dictates how much data are dropped locally at each node, and thus drop this assumption.

[3]Each equality constraint in (1) can be replaced by two inequality constraints that should be satisfied concurrently; alternatively we can use lagrange multipliers that can take negative values.

[4]For classification tasks, for instance, the transmitted images can be used to re-train/update the classifiers, achieving eventually their limiting classification error; see also the discussion in Section VI.

Algorithm 1 FW with constant step and changing objective

1: **Set:** $\beta \in (0,1]$ and $x_1 \in X$.
2: **for** $t = 1, 2, \ldots,$ **do**
3: $\quad s_t \leftarrow u \in \arg\min_{u \in X} \; u^\top \nabla f_t(x_t)$
4: $\quad x_{t+1} \leftarrow (1-\beta)x_t + \beta s_t$
5: **end for**



(a) Frank-Wolfe

(b) Dual subgradient

Fig. 2: **(a)**: In each iteration we select an extreme point and move the average x_t in that direction, until it converges within 2η from the optimum. **(b)**: Convergence of the dual subgradient algorithm.

1) Frank-Wolfe algorithm: FW is a projection-free algorithm for unconstrained convex problems with smooth objective.[5] It was designed to solve static problems (fixed f) by making convex combinations of actions [18]. In our previous work [19], we showed[6] that FW converges also when we have a sequence of functions $\{f_t\}_t$, in the sense that, as t increases, it manages to reduce the gap $|f_t(x_t) - f_t(x_t^*)|$ until it becomes smaller than a set bound 2η. The steps are detailed in Algorithm 1 and presented schematically in Fig. 2(a). In each iteration, we compute the gradient of f_t at x_t and select a vector s_t that minimizes $u^\top \nabla f(x_t)$, w.r.t u (Step 4). Then, we update the running average x_{t+1}, where $\beta \in (0,1]$ is a selected parameter, and repeat the process until convergence. FW is useful for our problem due to the following lemma.

Lemma 1 (Discrete Actions): The FW minimization step 3 in Algorithm 1 yields an eligible schedule in S.

Proof: First, recall that solving a LP over a polytope results to an extreme point [17, Prop. 3.4.2]. By Proposition 1, conv(S) satisfies (4). The same holds for X by definition, and the two sets are identical. Since the FW step yields an extreme point of X [18], we conclude that it is also a point in S, i.e., an implementable action. ∎

2) The Approximate Dual Subgradient Method (ADSM): This algorithm leverages the dual functions $\{h_t\}_t$ to maximize the time-average $\sum_{\tau=1}^t f_\tau(x_\tau)/t$ which, based on *Assumption 2*, is equivalent in limit to the static TCRA. See Algorithm 2 and the example in Fig. 2(b). In each slot, we select x_t as:

$$\mathcal{X}(\gamma_t, \lambda_t) := \big\{ x \in X \mid h_t(\lambda_t) \le L_t(x, \lambda_t) \le h_t(\lambda_t) + \gamma_t \big\}$$

[5]A convex function f is M-smooth if there exists a constant $M \ge 0$ such that $f(v) \le f(u) + \nabla f(u)^\top (v-u) + 2^{-1} M \|v-u\|^2$ for all $u, v \in X$.

[6]Similar results were developed in the context of online convex optimization where FW is applied to sequence of changing functions; albeit the performance criterion is the ergodic convergence and not $|f_t(x_t) - f_t(x_t^*)|$.

TABLE I: Constants and Bounds

Parameter	Definition
σ_f	$\lvert f_{t+1}(x) - f_t(x)\rvert \le \sigma_f, \forall x \in X, t$
σ_g	$\|g_t(x)\| \le \sigma_g, \forall x \in X, t$
σ_L	$\lvert h_{t+1}(\lambda_{t+1}) - h_t(\lambda_t)\rvert \le \sigma_L, \forall \lambda_t$
R	$\|x - y\| \le R, \forall x, y \in X$
Λ_t	Upper bound on $\|\lambda_t\|$; see Lemma 4.
Λ	$\Lambda = \max_t \Lambda_t$
Perturbations	$\epsilon_t = w_t - w, \;\; \phi_t = \bar{\delta}_t - \delta$
Assumption 2	$\lim_{t\to\infty} w_{i,t}^{(c)} = w_i^{(c)}, \;\; \forall i, c$
Assumption 3	$\lim_{t\to\infty} \bar{\delta}_t = \delta$

Algorithm 2 Approximate Dual Subgradient

1: **Set:** $\alpha > 0$ and $\lambda_1 = 0$
2: **for** $t = 1, 2, \ldots$ **do**
3: $\quad x_t \leftarrow u \in \mathcal{X}(\gamma_t, \lambda_t)$
4: $\quad \lambda_{t+1} \leftarrow [\lambda_t + \alpha g_t(x_t)]^+$
5: **end for**

This approximate minimization of the Lagrangian $L_t(\cdot, \lambda_t)$ amounts to using ϵ-subgradients [17, pp. 235] with upper bound of error γ_t in each slot. The algorithm ensures bounded optimality and constraint violation:

Lemma 2 (Approximate dual subgradient method): Algorithm 2 achieves the following bounds:

$$\text{(i)} \;\; \frac{1}{t}\sum_{\tau=1}^t f_\tau(x_\tau) - f(x^\circ) \le \frac{\alpha \sigma_g^2}{2} + \frac{1}{t}\sum_{\tau=1}^t \gamma_\tau + \epsilon_\tau^\top \hat{x}_\tau + \lambda_\tau^\top \phi_\tau$$

$$\text{(ii)} \;\; \Big\| \frac{1}{t}\sum_{\tau=1}^t g_\tau(x_\tau) \Big\| \le \frac{\Lambda_{t+1}}{\alpha t},$$

where $\epsilon_\tau = w_\tau - w$, $\phi_\tau = \bar{\delta}_\tau - \delta$, and see also Table I.

We explain next how we can combine these two algorithms to solve TCRA in an online fashion.

B. Online Approximate Scheduling Algorithm

The key idea is to apply Algorithm 1 in the t-slot Lagrangian (instead of the f_t) and use the dual subgradient Algorithm 2 with the t-slot constraints. The steps are shown in Algorithm 3. We first set the design parameters β and α, and initialize the variables (Step 1). In each slot t, we observe the current f_t and g_t, construct the t-slot Lagrangian $L_t(v_t, \lambda_t)$ and solve an LP to find the schedule s_t (Step 3). As explained above, this ensures that $s_t \in \mathcal{S}$ since changing from f_t to L_t does not affect Lemma 1. We can therefore implement directly this schedule (Step 4). Next we update the average value of the primal variables (Step 5) and perform a subgradient update for the dual variables (Step 6). In each slot t we only use information that has been made available up to that slot.

Algorithm 3 generates a sequence of vectors $\{s_\tau\}_{\tau=1}^t$ which ensure that the system operation approaches the performance prescribed by the solution of TCRA. The following theorem formalizes the performance of FWDS.

Algorithm 3 Frank-Wolfe Dual Subgradient (FWDS)

1: **Set:** $\beta \in (0,1]$, $\alpha > 0, v_1 \in X$ and $\lambda_1 = 0$.
2: **for** $t = 1, 2, \dots$ **do**
3: $\quad s_t \leftarrow \arg\min_{u \in X} \; u^\top \nabla_v L_t(v_t, \lambda_t)$
4: $\quad$ Implement schedule $s_t = (y_{ij}^{(c)}, z_i^{(c)}) \in S$
5: $\quad v_{t+1} \leftarrow (1-\beta) v_t + \beta s_t$
6: $\quad \lambda_{t+1} \leftarrow [\lambda_t + \alpha g_t(v_t)]^+$
7: **end for**

Theorem 1: Consider the updates in Algorithm 3 with α and β selected as per Lemma 5 (see Sec.VIII). It holds:

$$\text{(i)} \; \frac{1}{t}\sum_{\tau=1}^{t} f_\tau(s_\tau) - f(x^\circ) \leq \frac{\alpha \sigma_g^2}{2} + 2\Lambda\sigma_g + \frac{1}{t}\sum_{\tau=1}^{t}\Big(\epsilon_\tau^\top \hat{x}_\tau + \lambda_\tau^\top \phi_\tau + \max\{\zeta_\tau, 2\eta\}\Big)$$

$$\text{(ii)} \; \|\frac{1}{t}\sum_{\tau=1}^{t} g_\tau(s_\tau)\| \leq \frac{\Lambda_{t+1}}{\alpha t}(1 + \frac{1}{\beta})$$

where $\zeta_t = L_{t+1}(v_t, \lambda_{t+1}) - h_{t+1}(\lambda_{t+1})$.

Discussion. The algorithm's performance depends on parameters α, β and η, which need to be carefully selected, namely:

$$0 < \beta \leq 1, \quad 0 < \alpha < \frac{\beta\eta - (\beta^2 M_L R^2)/2 - \sigma_f - \sigma_L - 2\Lambda\sigma_g}{\sigma_g^2};$$

see Lemma 5 for the proof and Table I for the parameters.[7] It is evident that α negatively affects the optimality gap, while both α and β do not impact the constraint violation as they are amortized by t. Parameter β however, affects η in the selection of α (which must be positive), and can thus indirectly decrease the optimality gap, see Lemma 5. One needs to select these parameters based on the desirable system performance, e.g., whether the priority lies in execution accuracy, or in reducing the backlogs (i.e., delay).

Let us now elaborate on the bounds of the Theorem. Starting with (i), the first term can be made arbitrarily small by selecting α. The first term in the summation is upper bounded by $|\epsilon_\tau^\top \hat{x}_\tau| \leq \|\epsilon_\tau\| \|\hat{x}_\tau\|$ where $\|\epsilon_\tau\|$ diminishes since $w_t \to w$. And in the same way we can show that the second term diminishes as long as the dual variables $\|\lambda_t\|$ are bounded, which is prove in Lemma 4 of the Appendix. The last term in the summation depends on parameter η and is affected by α and β, as explained above, while ζ_t is a decreasing sequence as shown in Lemma 5; hence the term $\max\{\zeta_\tau, 2\eta\}$ is replaced by 2η for τ sufficiently large. Regarding the feasibility bound, we showed that it diminishes with α, t, but we have used the running average of perturbations on g_t, i.e. $\bar{\delta}_t$, and not δ_t. However, it is easy to see that we can write

$$\|\frac{1}{t}\sum_{\tau=1}^{t} A s_\tau + \delta_\tau\| \leq \|\frac{1}{t}\sum_{\tau=1}^{t} A s_\tau + \bar{\delta}_\tau\| + \|\frac{1}{t}\sum_{\tau=1}^{t} \delta_\tau - \bar{\delta}_\tau\|.$$

This adds the residual $\|\frac{1}{t}\sum_{\tau=1}^{t} \delta_\tau - \bar{\delta}_\tau\|$, which is a decreasing with t sequence, since as more input samples are added, the running average will converge to the true average $\bar{\delta}_t$.

Finally, in terms of overheads and comlexity, the bad news is that FWDS requires the solution of an NP-hard problem, but the good news is it has still smaller complexity than the state-of-the-art network control algorithms. In particular, while Steps 5-7 involve simple calculations that can be executed in polynomial time and in a decentralized fashion; Step 4 requires the centralized solution of a problem that is at least as hard as the *weighted maximum independent set* problem. This issue, however, arises in all wireless scheduling algorithms, cf. [9], – we elaborate in Section V – and the various proposals for reducing the complexity or enabling its distributed solution, at the expense of achievable throughput or increased backlogs, apply also to FWDS. Moreover, in many IoT networks that include small nodes, there is already provision for central gateways or cluster-head nodes, which can undertake the role of executing these computations.

V. Related Work

The main advantages of FWDS over other wireless network control algorithms are that it: *(i)* offers deterministic non-asymptotic bounds, i.e., hold per sample path and we characterize its convergence rate; *(ii)* supports serving non-i.i.d. or non-Markovian task arrivals; and *(iii)* can handle changing objective functions.

A. Wireless Scheduling & Optimization

Network control techniques were revolutionized by the seminal backpressure and maxweight stability algorithms of Tassiulas and Ephremides, [20], [21]. These works spurred a flurry of related efforts [22] and were eventually extended by Neely et al with the Drift-Plus-Penalty (DPP) algorithm that optimizes any convex objective, thus enabling the design of cross-layer algorithms, cf. [9]. DPP algorithms *run in two stages*. First, they select a continuous action, e.g., data admission, by solving the convex program $x_t \in \arg\max_{x \in X} Vf(x) - J_t^\top x$, where J is a vector of virtual queues (modeling constraints) and $V > 0$ a penalty parameter. And second, they find a link activation schedule s_t that minimizes $\Delta Q + \Delta J$, i.e., the drift of packet and virtual queues. They are proven to converge asymptotically, and on expectation, to the optimal point while ensuring bounded constraint violation; and by tuning V we can trade-off optimality for feasibility.

Unlike the above approaches, FWDS solves only a linear program for finding s_t. This operation (Step 4) is almost identitcal with the schedule selection in maxweight and DPP algorithms, the only difference being that we use the Lagrangian. Hence, FWDS has the same complexity with the respective step in those algorithms, namely it is NP-hard as it involves an exponential number of constraints[8]; but overall FWDS is simpler as it does not require solving additionally a convex program (first stage of DPP). Moreover, FWDS can readily include a range of techniques developed for: *reducing*

[7] Similarly to the definition of M, $L_t(\cdot, \lambda)$ is a M_L-smooth function where $L_t(v, \lambda) \leq L_t(u, \lambda) + \nabla L_t(u, \lambda)^\top (v - u) + 2^{-1} M_L \|v - u\|^2$ for all $u, v \in X, t$.

[8] In fact the complexity of this program depends on the interference model. For 1-hop interference model it has polynomial complexity, but for other models it is as hard as the Weighted Maximum Independent Set problem.

the computation time of maxweight/DPP algorithms, e.g., using pick-and-compare matchings, see discussion in [9, Ch. 7] and [23]; or for *enabling their distributed execution* through message passing or randomization, cf. [24]. Many of these suggestions can be directly applied to FWDS, and this is a promising direction for future work.

In a different line of work, Stolyar proposed the Greedy-Primal Dual (GPD) algorithm in [11] which selects control actions *greedily* by solving $s_t \in \arg\min_{s\in S} (\nabla f(x_t) + \beta Q_t \Delta Q)^\top s$. The continuous variables are updated with $x_{t+1} = (1-\beta)x_t + \beta s_t$. GPD is shown to converge asymptotically, but we note that it uses exponential averaging $\{s_\tau\}_{\tau=1}^t$ and hence does not ensure ergodic convergence. Moreover, the objective is fixed and known, and the data arrivals are constant or follow i.i.d. processes. Our approach is inspired from [16] which we extend to include perturbations and to obtain discrete decisions through the Frank-Wolfe operation. Finally, we note the similarities of our work with the recent works in [19], [25]. In [19] the authors propose a greedy/FW algorithm to solve deterministic convex programmes; while [25] uses DPP with a FW updates, yet the convergence bounds are not deterministic and do not handle time-varying objective functions.

B. IoT Analytics and Edge Computing

The works in [26] and [27] propose max-weight-based scheduling algorithms for processing networks. They perform utility maximization scheduling by controlling the amount of data that is admitted to the network towards providing queue stability. [28] studies the joint task and network flow allocation towards overall task completion time minimization, for an edge computing network. [29] presents a task scheduling framework for a fog network, where the IoT devices generate tasks with deadlines. The authors formulate a knapsack problem and propose an algorithm to maximize the service provider's revenues. Recent works emphasize the importance of executing analytics in edge/cloud infrastructures, e.g. [30], [31]. Most of them perform task offloading to optimize execution delay [3], [28], [32] or power consumption [33].

Contrary to [26] and [27], our solution optimizes a possibly time-varying task utility for the nodes, on top of satisfying stability, while [28] does not embed routing, and a separate algorithm is required. Many related works [29], [33], do not consider interference constraints that complicate scheduling. On the other hand, our solution deals with interference without heavily increasing the complexity of the proposed algorithm by utilizing fast FW primal updates.

Finally, in-network processing gains increasing attention today because it can effectively support latency-sensitive data-heavy applications, and there are several proposals for joint routing and computing policies [34]–[37]. For example, [35] was among the first works to apply a maxweight policy for routing and processing, focusing on big data applications; similarly [36] designs *stability* policies for wired networks with i.i.d. arrivals; while our previous work [37] optimizes the network lifetime assuming, however, interference-free transmissions. The current work builds on and expands these efforts to account for interference, varying network state and objectives, and non-i.i.d. task requests.

TABLE II: Application parameters.

Algorithm/Node	Accuracy	Cycles/bit
LBPH/NUC	82 %	670
Eigen/NUC	74 %	520
Fisher/IoT	42 %	360

VI. Performance Evaluation

We have utilized a wireless testbed comprised of small IoT and edge nodes to evaluate the performance of an image classification application. The operation of the proposed FWDS algorithm is simulated and compared against several benchmarks, with respect to complexity, utility performance and network delay; and for random networks generating non-i.i.d. traffic loads. We first explain the setup of the experiments and then present the results.

A. Experimental Setup

We simulate the operation of an IoT network of wireless cameras where the nodes run face recognition tasks. We used random geometric graphs to create the networks, i.e., we place randomly the nodes in a 100×100 meters area and connect two nodes if their euclidean distance is less than 30 meters. There are two type of nodes, namely Raspberry Pis and NUC nodes[9], that differ on their computation and memory capacity. The nodes create images to be classified using a typical face recognition application developed in Python with OpenCV. We applied 3 different algorithms on the Georgia Tech face database [38] that has 750 images.

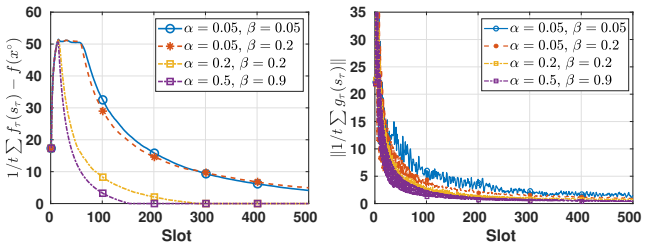
We performed initial measurements in order to calculate the various system parameters. First, we measured the average image size of the dataset and the average execution time for each of the algorithms and type of node. Based on the CPU frequency of the nodes we measured the average $\rho_i^{(c)}$. We report the results, along with the achieved average accuracy $w_i^{(c)}$ of each algorithm in Table II. Observe that the more accurate an algorithm, the more computationally expensive it is. Thus we assign the algorithms so that low computation power nodes (RPis) run the least demanding algorithm (Fisher), and the NUCs run Eigen and LBPH, respectively. We fully trained the algorithms on our dataset to obtain the reported accuracy values. Hence, in the following, we have $w_{i,t}^{(c)} = w_i^{(c)}$, $\forall t$, and also $f_t(x) = f(x)$, $\forall t$. This means that the bounds of Algorithm 3 still hold, but in Lemma 5 we have $\sigma_f = 0$.

B. Parameter Sensitivity Analysis

We start by exploring numerically the impact of algorithm parameters α and β, on its performance. We simulate FWDS for 500 slots and study how the optimality and feasibility bounds of Theorem 1 evolve. Fig. 3 presents the results for different α and β values. Note that the selected values for α and β not affect the convergence bounds of FWDS both directly (see Theorem 1) and indirectly through η.

We observe that the optimality gap in Fig. 3a decreases with t for all cases. Notice that from the bound of Theorem 1, the gap decreases with α. However, increasing α results

[9]The RPis are Model 3B with 1.2 GHz CPU, 1GB RAM; and the NUCs have 2.3 GHz CPU and 16 GB RAM memory.



(a) Optimality gap (b) Constraint violation

Fig. 3: Convergence of FWDS with $N = 40, C = 20$.

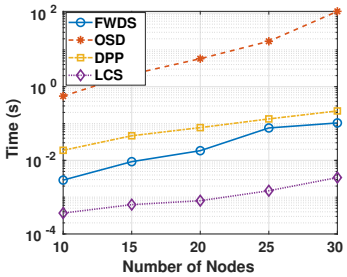


Fig. 4: Time complexity of computing a single schedule.

in slightly better convergence performance. This counter-intuitive behavior is due to the intricate relations between α, β, η. Reducing α requires decreasing β, which in turn might increase η and undermine the optimality gap. Regarding the feasibility gap in Fig. 3b, note that it decreases with t. Hence the differences between the cases of selected α, β are smoothed as t increases, being noticeable only in the early stages of execution. These experiments *demonstrate numerically the convergence of our algorithm and its dependence on the system parameters*, verifying our theoretical analysis.

C. Comparison with Benchmarks

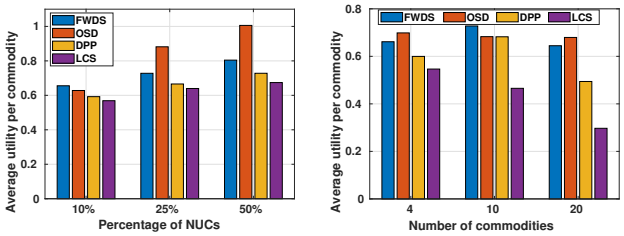
We evaluate the performance of Algorithm 3 w.r.t. computation complexity for acquiring a schedule in each slot, and congestion created as a result of the scheduling decisions. We compare FWDS with the following benchmarks:

1) Optimal Solution Decomposition (OSD). We minimize $L_t(x, \lambda)$ to obtain the optimal x_t^*, and we decompose it to a set of implementable schedules (time-sharing). This can be accomplished by, e.g., using the FW algorithm on $\|x - x_t^*\|^2$. Hence, we obtain a set of schedules and their associated weights, such that their convex combination approaches $-w^\top x_t^*$, and choose a schedule randomly based on its weight.

2) Drift Plus Penalty (DPP). Based on the max-weight algorithm [9], the DPP [7] minimizes the queue backlogs while also maximizing some other performance metric for the network, which for TCRA is the aggregate nodes utilities, multiplied by the control parameter V.

3) Linear Complexity Scheduling (LCS). Inspired by [10], we implement a simple policy where, having knowledge of the possible schedules, we pick a schedule $s \in S$ randomly. At each iteration, we pick another schedule $\hat{s}$ at random and compare $s^\top \nabla_x L_t(x_t, \lambda_t)$ to $\hat{s}^\top \nabla_x L_t(x_t, \lambda_t)$. We then apply the schedule that yields the minimum value.

Fig. 4 depicts the required time for each algorithm to compute the schedule in each time slot, for different network sizes. Note that this delay will impact the system's slot duration. It is evident that OSD is by orders of magnitude slower than the rest, and thus its practical application in a large IoT network is challenging if not impossible. FWDS is faster than DPP, since it does not have to solve a convex resource allocation problem (DPP's first stage). However, it is slower than the LCS algorithm but the difference is much more tractable, making it suitable for fast scheduling. In essence, *FWDS can be utilized in systems where resource allocation decisions need to be taken every few seconds.*



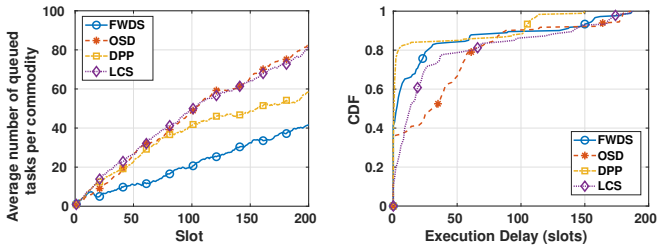
(a) Utility for $C = 20$. (b) Utility for 10 NUCs.

Fig. 5: Average network utility for $N = 40$ and varying number of commodities/NUCs.

Next we evaluate the average network utility obtained by each algorithm (Fig. 5a). We consider different scenarios where we modify the number of NUC nodes, in a 40-node network. The results indicate that, as expected, the average utility increases with the percentage of NUCs. Moreover, we observe that OSD manages to clearly outperform the competition when there is more than 25% of NUCs, while our solution has the edge over the max-weight based algorithms. Fig. 5b shows again the network's average utility, but this time for a fixed number of NUCs and a varying load, as we increase the number of nodes that generate traffic, i.e. the commodities. The results, once again demonstrate that FWDS roughly maintains the same performance, despite the load increase, compared to DPP and especially LCS, while is also competitive to OSD in all load cases.

Fig. 6a presents the network congestion for the operation of a 40-node hybrid network, where 20 of the nodes create tasks and the rest of them act as relays and computing nodes. We see that having high utility in OSD comes at the cost of high network congestion. This is because decomposing the optimal solution to a set of feasible schedules takes many slots; hence OSD keeps implementing the same set of (outdated) schedules until the next primal update iteration. However, the dynamics of the system require better response to the implemented schedules and congestion builds up. LCS although very fast, makes random, "naive" scheduling selections in each slot and thus performs quite poorly, in both utility and congestion. On the other hand, DPP is better for both metrics, and allow us to trade-off one for the other by tuning the design parameter V. Still, *FWDS performs better than DPP in both accuracy and network congestion.*

The performance of data analytics services is also affected by their execution delay. Fig. 6b displays the CDF of total execution delay for the same setup as with Fig. 6a. Observe



(a) Congestion (b) Task execution delay

Fig. 6: Network congestion in tasks per commodity and task execution delay CDF with $N = 40,\ C = 20$.

that the CDF of OSD and LCS lay below FWDS and DPP for most delay values. We also observe that FWDS performs slightly worse than DPP, which can be attributed to having superior utility performance. However, *both FWDS and DPP achieve a delay of about 30 slots for more than 80% of tasks.*

VII. Conclusion

In this paper we combined the ADSM and FW algorithms to provide a robust scheduling policy that optimizes the system's performance and keeps congestion in low levels compared to other standard approaches. Our algorithm makes fast schedule evaluations, since it solves a LP in each iteration that is also a feasible schedule. Moreover, it is a solution tailored to analytic services where the parameters like traffic load, network capacities and even task execution accuracy vary over time in a possibly non-i.i.d way. We provide theoretical convergence bounds for the proposed solution and finally demonstrate the high performance level of our algorithm, comparing it to standard scheduling methods.

VIII. Appendix

As f is M-smooth, it is $f(y)\leq f(x)+\nabla_x f(x)^\top(y-x)+\frac{M}{2}\|y-x\|^2$. Let $y=x_{t+1}=(1-\beta)x_t+\beta s_t$, $x=x_t$, to obtain:

$$f(x_{t+1}) \leq f(x_t) + \beta\nabla_x f(x_t)^\top(s_t - x_t) + \beta^2\frac{MR^2}{2}. \quad (7)$$

Proof of Proposition 1. The proposition will be true if any schedule that satisfies the constraints can be written as a convex combination of schedules in S. Consider schedule s for which (2) - (4) hold $\forall(i,j)$. Further, assume that for link (i,j) we have $\sum_c y_{ij}^{(c)} > 0$, and $\sum_c y_{kl}^{(c)} > 0, \forall(k,l) \in I(i,j)$, so $s \notin S$. Consider the following schedules that belong in S:

- s_{ij}: Same as s but holds $\sum_c y_{ij}^{(c)} = \mu_{ij}$, and $\sum_c y_{kl}^{(c)} = 0, \forall(k,l) \in I(i,j)$.
- s_{kl}: Same as s but $\sum_c y_{ij}^{(c)} = 0$, $\sum_c y_{kl}^{(c)} = \mu_{kl},\ (k,l) \in I(i,j)$, and $\sum_c y_{mn}^{(c)} = 0,\ (m,n) \in I(i,j) - \{(k,l)\}$.

The above schedules mean that either (i,j) or one of its interfering links (k,l) is active. W.l.o.g. we assume that s satisfies (4) strictly, i.e $\xi_{ij} + \sum_{(k,l)\in I(i,j)} \xi_{kl} = 1$, where $\xi_{ij} = \sum_c y_{ij}^{(c)}/\mu_{ij} \in (0,1)$, and $\xi_{kl} = \sum_c y_{kl}^{(c)}/\mu_{kl} \in (0,1),\ (k,l) \in I(i,j)$. Observe that we can express s as a convex combination of schedules that belong to S, i.e.,

$$s := \xi_{ij}s_{ij} + \sum_{(k,l)\in I(i,j)} \xi_{kl}s_{kl}.$$

This holds also when s has many active interfering links that respect (4), since we can express them as convex combinations of extreme point schedules from S. ■

The next two lemmas are used in the proof of Lemma 2.

Lemma 3 (Bounded level set): Let the Slater condition hold, i.e. there exists a vector x_s such that $g_t(x_s) < 0, \forall t$. The superlevel set $\mathcal{Q}_{\hat{\lambda}} = \{\lambda \succeq 0 \mid h_t(\lambda) \geq h_t(\hat{\lambda})\}$ is bounded. That is, for any $\lambda \in \mathcal{Q}_{\hat{\lambda}}$ we have

$$\|\lambda\| \leq \big(f_t(x_s) - h_t(\hat{\lambda})\big)/q_t, \text{where } q_t = \min_i\{-g_{t,(i)}(x_s)\}.$$

Proof: $\forall\lambda \in \mathcal{Q}_{\hat{\lambda}}$ it is $h_t(\hat{\lambda}) \leq h_t(\lambda) = \min_x\{f_t(x) + \lambda^\top g_t(x)\} \leq f_t(x_s)+\lambda^\top g_t(x_s) = f_t(x_s)+\sum_{i=1}^m \lambda(i)g_{t,(i)}(x_s)$. Since $g_{t,(i)}(x_s) < 0$, and $\lambda(i) \geq 0$ we have:

$$\min_i\{-g_{t,(i)}(x_s)\}\sum_{i=1}^m \lambda(i) \leq f_t(x_s) - h_t(\hat{\lambda}) \Rightarrow$$
$$\|\lambda\| \leq \frac{1}{q_t}(f_t(x_s) - h_t(\hat{\lambda})). \quad ■$$

Lemma 4 (Bounded Multipliers): The multiplier sequence generated by Algorithm 2 is bounded, i.e., $\|\lambda_t\| \leq \Lambda_t \triangleq$

$$2\frac{f_t(x_s)-h_t(\lambda^\circ)}{q_t} + \max\left\{\|\lambda_1\|, \frac{f_t(x_s)-h_t(\lambda^\circ)}{q_t} + \frac{\alpha\sigma_g^2}{2q_t} + \alpha\sigma_g\right\}$$

Proof: We define the superlevel set $\mathcal{Q}_a = \{\lambda \succeq 0 \mid h_t(\lambda) \geq h_t(\lambda^\circ) - \frac{\alpha\sigma_g^2}{2}\}$ and firstly prove that $\|\lambda_t - \lambda^\circ\| \leq$

$$\max\left\{\|\lambda_1 - \lambda^\circ\|, \frac{f_t(x_s)-h_t(\lambda^\circ)}{q_t} + \frac{\alpha\sigma_g^2}{2q_t} + \alpha\sigma_g + \|\lambda^\circ\|\right\} \quad (8)$$

Observe that the above holds for $t = 1$. We are going to assume it holds for any t, and prove it holds for $t+1$. **Case** *(i)*: $h_t(\lambda_t) \geq h_t(\lambda^\circ) - \frac{\alpha\sigma_g^2}{2}$. By the dual update rule:

$$\begin{aligned}\|\lambda_{t+1} - \lambda^\circ\| &= \|[\lambda_t + \alpha g_t(x_{t+1})]^+ - \lambda^\circ\| \\ &\leq \|\lambda_t\| + \|\lambda^\circ\| + \alpha\sigma_g \\ &\leq \frac{1}{q_t}(f_t(x_s) - h_t(\lambda^\circ) + \frac{\alpha\sigma_g^2}{2}) + \|\lambda^\circ\| + \alpha\sigma_g\end{aligned}$$

where the last inequality follows from Lemma 3 for $\mathcal{Q}_a$, hence (8) holds. **Case** *(ii)*: $h_t(\lambda_t) < h_t(\lambda^\circ) - \frac{\alpha\sigma_g^2}{2}$. We have:

$$\begin{aligned}\|\lambda_{t+1} - \lambda^\circ\|^2 &\leq \|\lambda_t + \alpha g_t(x_{t+1}) - \lambda^\circ\|^2 \\ &\leq \|\lambda_t - \lambda^\circ\|^2 + 2\alpha g_t(x_{t+1})^\top(\lambda_t - \lambda^\circ) + \alpha^2\sigma_g^2 \\ &\leq \|\lambda_t - \lambda^\circ\|^2 - 2\alpha(h_t(\lambda^\circ) - h_t(\lambda_t)) + \alpha^2\sigma_g^2 \\ &\leq \|\lambda_t - \lambda^\circ\|^2,\end{aligned}$$

where the last inequality holds by the assumption of this case, making (8) true. Now we can rewrite (8) as $\|\lambda_t - \lambda^\circ\| \leq$

$$\max\left\{\|\lambda_1 - \lambda^\circ\|, \frac{f_t(x_s) - h_t(\lambda^\circ)}{q_t} + \frac{\alpha\sigma_g^2}{2q_t} + \alpha\sigma_g + \|\lambda^\circ\|\right\}$$
$$\leq \|\lambda^\circ\| + \max\left\{\|\lambda_1\|, \frac{f_t(x_s) - h_t(\lambda^\circ)}{q_t} + \frac{\alpha\sigma_g^2}{2q_t} + \alpha\sigma_g\right\}.$$

Since $\|\lambda_t\| \leq \|\lambda_t - \lambda^\circ\| + \|\lambda^\circ\|$ we obtain $\|\lambda_t\| \leq 2\|\lambda^\circ\| + \max\left\{\|\lambda_1\|, \frac{f_t(x_s)-h_t(\lambda^\circ)}{q_t} + \frac{\alpha\sigma_g^2}{2q_t} + \alpha\sigma_g\right\}$, and by applying Lemma 3 for $\hat{\lambda} = \lambda^\circ$ we obtain the claimed bound. ■

Proof of Lemma 2. We first prove the upper bound on the objective function by linking the static and t-slot problems.

$$\begin{aligned} h_t(\lambda) &= \min_{x\in X} L_t(x,\lambda) \\ &= \min_{x\in X} f(x) + \epsilon_t^\top x + \lambda^\top(Ax + \delta + \bar{\delta}_t - \delta) \\ &= \min_{x\in X} f(x) + (\epsilon_t^\top B + \lambda^\top)(Ax+\delta) + \lambda^\top\phi_t - \epsilon_t^\top B\delta \\ &= h(B^\top\epsilon_t + \lambda) + \lambda^\top\phi_t - \epsilon_t^\top B\delta, \end{aligned} \tag{9}$$

where $B \in \mathbf{R}^{e\times m}$ such that $BA = I_e$, and I_e is the unit matrix of dimension e. By the concavity of h we can write:

$$h(\mu_t) \leq h(\lambda_t) + (\mu_t - \lambda_t)^\top h'(\lambda_t),$$

and by setting $\mu_t = B^\top\epsilon_t + \lambda_t$, (9) becomes:

$$\begin{aligned} h_t(\lambda_t) &\leq h(\lambda_t) + (B^\top\epsilon_t)^\top(A\hat{x}_t + \delta) + \lambda_t^\top\phi_t - \epsilon_t^\top B\delta \\ &\leq h(\lambda_t) + \epsilon_t^\top\hat{x}_t + \lambda_t^\top\phi_t, \end{aligned}$$

where $A\hat{x}_t + \delta = h'(\lambda_t)$ is a subdifferential of h at λ_t. Hence:

$$\begin{aligned} f(x^\circ) = h(\lambda^\circ) &\geq \frac{1}{t}\sum_{\tau=1}^t h(\lambda_\tau) \geq \frac{1}{t}\sum_{\tau=1}^t h_\tau(\lambda_\tau) - \epsilon_\tau^\top\hat{x}_\tau - \lambda_\tau^\top\phi_\tau \\ &\overset{(a)}{\geq} \frac{1}{t}\sum_{\tau=1}^t L_\tau(x_\tau,\lambda_\tau) - \gamma_\tau - \epsilon_\tau^\top\hat{x}_\tau - \lambda_\tau^\top\phi_\tau \\ &\geq \frac{1}{t}\sum_{\tau=1}^t f_\tau(x_\tau) + \lambda_\tau^\top g_\tau(x_\tau) - \gamma_\tau - \epsilon_\tau^\top\hat{x}_\tau - \lambda_\tau^\top\phi_\tau. \end{aligned}$$

where (a) holds from the approximate minimization of the Lagrangian. Rearranging terms we obtain:

$$\begin{aligned} &\frac{1}{t}\sum_{\tau=1}^t f_\tau(x_\tau) - f(x^\circ) \leq \\ &-\frac{1}{t}\sum_{\tau=1}^t \lambda_\tau^\top g_\tau(x_\tau) - \gamma_\tau - \epsilon_\tau^\top\hat{x}_\tau - \lambda_\tau^\top\phi_\tau. \end{aligned} \tag{10}$$

We proceed to upper bound the first term in the RHS of the last equation. Observe that for any $\theta \in \mathbf{R}_+^m$ we have $\|\lambda_{t+1} - \theta\|^2 = \|[\lambda_t + \alpha g_t(x_t)]^+ - \theta\|^2 \leq \|\lambda_t + \alpha g_t(x_t) - \theta\|^2 = \|\lambda_t - \theta\|^2 + \alpha^2\|g_t(x_t)\|^2 + 2\alpha(\lambda_t - \theta)^\top g_t(x_t)$. Rearranging:

$$\|\lambda_{t+1} - \theta\|^2 - \|\lambda_t - \theta\|^2 \leq \alpha^2\|g_t(x_t)\|^2 + 2\alpha(\lambda_t - \theta)^\top g_t(x_t).$$

Setting $\theta = 0$, and applying the telescopic summation:

$$-2\alpha\sum_{\tau=1}^t \lambda_\tau^\top g_\tau(x_\tau) \leq \alpha^2\sum_{\tau=1}^t \|g_\tau(x_\tau)\|^2,$$

where $-\|\lambda_{t+1}\|^2$ and $\|\lambda_1\|^2$ were dropped in the above since the former is non-positive, and the latter can be zeroed out by setting $\lambda_1 = 0$. Using the fact that $\|g_\tau(x)\| \leq \sigma_g, \forall\tau$, for all $x \in X_\tau$, and diving across by $2\alpha t$ we obtain $-\frac{1}{t}\sum_{\tau=1}^t \lambda_\tau^\top g_\tau(x_\tau) \leq \frac{\alpha\sigma_g^2}{2}$. By using the above bound on (10) we prove the first claim.

From the dual update rule recursive expansion we have $\lambda_{t+1} \succeq \lambda_1 + \alpha\sum_{\tau=1}^t g_\tau(x_\tau)$. By dropping λ_1, dividing by αt, and taking the norms:

$$\|\frac{1}{t}\sum_{\tau=1}^t g_\tau(x_\tau)\| \leq \frac{\|\lambda_{t+1}\|}{\alpha t}, \tag{11}$$

which yields the claimed bound after applying Lemma 4.

We need the following Lemma for proving Theorem 1. Note also, that in order to differentiate between Algorithms 2 and 3 we use v_t for the primal variable of the latter.

Lemma 5 (FW for the Lagrangian): Set $\eta > 0$ and select $0 < \alpha < \frac{\beta\eta - (\beta^2 M_L R^2/2) - \sigma_f - \sigma_L - 2\Lambda\sigma_g}{\sigma_g^2}$ and $\beta \in (0,1]$, M_L is a constant independent of α, β. Algorithm 3 guarantees that if $L_t(v_t,\lambda_t) - h_t(\lambda_t) \geq \eta$:

$$\begin{aligned} &L_{t+1}(v_{t+1},\lambda_{t+1}) < L_{t+1}(v_t,\lambda_{t+1}), \quad \text{or} \\ &L_{t+1}(v_{t+1},\lambda_{t+1}) - h_{t+1}(\lambda_{t+1}) \leq 2\eta, \ \text{otherwise.} \end{aligned}$$

Proof: Based on Lemma 4, the sequence of vectors $\{\lambda_t\}_t$ is bounded. Hence, $L_t(\cdot,\lambda_t)$ is an M_L-smooth convex function of v for any parameter λ. We consider two cases. **Case** *(i)*: $L_t(v_t,\lambda_t) - h_t(\lambda_t) > \eta$, where recall that $h_t(\lambda_t) \leq L_t(v,\lambda_t), \forall x$. Due to its smoothness and the FW rule applied on (7), we have $L_t(v_{t+1},\lambda_t) - L_t(v_t,\lambda_t) \leq$

$$\leq \beta\nabla_v L_t(v_t,\lambda_t)^\top(s_t - v_t) + \frac{\beta^2 M_L R^2}{2}, \tag{12}$$

and since s_t is the minimizer of $u^\top\nabla_v L_t(v_t,\lambda_t)$ we can replace the second term in the RHS with $\beta\nabla_v L_t(v_t,\lambda_t)^\top(v_t^* - v_t)$, where $v_t^* = \arg\min_v L_t(v,\lambda_t)$, i.e., $h_t(\lambda_t) = L_t(v_t^*,\lambda_t)$. Due to convexity of $L_t(\cdot,\lambda_t)$, it is $\nabla_v L_t(v_t,\lambda_t)^\top(v_t^* - v_t) \leq h_t(\lambda_t) - L_t(v_t,\lambda_t) < -\eta$, hence

$$L_t(v_{t+1},\lambda_t) - L_t(v_t,\lambda_t) \leq -\beta\eta + \beta^2\frac{M_L R^2}{2}. \tag{13}$$

Now, it holds that $L_{t+1}(v_{t+1},\lambda_{t+1}) - L_t(v_{t+1},\lambda_t) =$

$$\begin{aligned} &f_{t+1}(v_{t+1}) - f_t(v_{t+1}) + \lambda_{t+1}^\top g_{t+1}(v_{t+1}) - \lambda_t^\top g_t(v_{t+1}) \\ &= f_{t+1}(v_{t+1}) - f_t(v_{t+1}) + \lambda_t^\top\left(g_{t+1}(v_{t+1}) - g_t(v_{t+1})\right) \\ &+ \alpha g_t(v_t)^\top g_{t+1}(v_{t+1}) \leq \sigma_f + 2\|\lambda_t\|\sigma_g + \alpha\sigma_g^2, \end{aligned} \tag{14}$$

where we used Cauchy-Schwarz and triangle inequalities. Adding (13) and (14), we get: $L_{t+1}(v_{t+1},\lambda_{t+1}) \leq$

$$L_t(v_t,\lambda_t) - \beta\eta + \beta^2\frac{M_L R^2}{2} + \sigma_f + 2\Lambda\sigma_g + \alpha\sigma_g^2,$$

where we have defined $\Lambda = \max_t\|\lambda_t\|$. Subtracting $L_{t+1}(v_t,\lambda_{t+1})$in both sides, using $|L_{t+1}(v_t,\lambda_{t+1}) - L_t(v_t,\lambda_t)| \leq \sigma_L$, and by the stated choices of α,β we obtain that $L_{t+1}(v_{t+1},\lambda_{t+1}) < L_{t+1}(v_t,\lambda_{t+1})$.

Case *(ii)*: $L_t(v_t,\lambda_t) - h_t(\lambda_t) \leq \eta$. Since s_t is the minimizer of $u^\top\nabla_v L_t(v_t,\lambda_t)$, the term $\nabla_v L_t(v_t,\lambda_t)^\top(s_t - v_t)$ is non-positive and hence can be dropped from (12):

$$L_t(v_{t+1},\lambda_t) - L_t(v_t,\lambda_t) \leq (\beta^2 M_L R^2)/2. \tag{15}$$

Adding (14) to (15) and using $L_t(v_t,\lambda_t)-h_t(\lambda_t)\leq\eta$ we end up with $L_{t+1}(v_{t+1},\lambda_{t+1})-h_t(\lambda_t)\leq$

$$\leq L_t(v_t,\lambda_t)-h_t(\lambda_t)+\frac{\beta^2M_LR^2}{2}+\sigma_f+2\Lambda\sigma_g+\alpha\sigma_g^2$$
$$\leq \eta+\frac{\beta^2M_LR^2}{2}+\sigma_f+2\Lambda\sigma_g+\alpha\sigma_g^2,$$

where we used the assumption of this case for the last inequality. Subtracting $h_{t+1}(\lambda_{t+1})$ in both sides and rearranging terms yields $L_{t+1}(v_{t+1},\lambda_{t+1})-h_{t+1}(\lambda_{t+1})\leq$

$$\leq\eta+\frac{\beta^2M_LR^2}{2}+\sigma_f+\sigma_L+2\Lambda\sigma_g+\alpha\sigma_g^2\leq\eta+\beta\eta+\leq 2\eta,$$

where we used $|h_{t+1}(\lambda_{t+1})-h_t(\lambda_t)|\leq\sigma_L$, and the stated α,β, completing the second case. ∎

Proof of Theorem 1. By the stated selection of α and β, Lemma 5 applies and yields

$$L_{t+1}(v_{t+1},\lambda_{t+1})-h_{t+1}(\lambda_{t+1})\leq\max\{\zeta_t,2\eta\},$$

where $\zeta_t = L_{t+1}(v_t,\lambda_{t+1})-h_{t+1}(\lambda_{t+1})$. By applying Lemma 2 with $\gamma_t=\max\{\zeta_t,2\eta\}$ we obtain $1/t\sum_{\tau=1}^t f_\tau(v_\tau)-f(x^\circ)\leq$

$$\leq\frac{\alpha\sigma_g^2}{2}+\frac{1}{t}\sum_{\tau=1}^t\max\{\zeta_\tau,2\eta\}+\epsilon_\tau^\top\hat{x}_\tau+\lambda_\tau^\top\phi_\tau \qquad (16)$$

We proceed by bounding the average performance of then selected schedules s_t. By convexity of $L_t(\cdot,\lambda_t)$, we have

$$L_t(v_t,\lambda_t)\geq L_t(s_t,\lambda_t)+\nabla L_t^\top(s_t,\lambda_t)(v_t-s_t)\geq L_t(s_t,\lambda_t).$$

So it is $f_t(s_t)\leq f_t(x_t)+\lambda_t^\top\big(g_t(x_t)-g_t(s_t)\big)$ and by summing telescopically we have

$$\frac{1}{t}\sum_{\tau=1}^t f_\tau(s_\tau)\leq\frac{1}{t}\sum_{\tau=1}^t f_\tau(x_\tau)+2\Lambda\sigma_g.$$

Combining the above with (16) we obtain the claimed bound (i). For the feasibility gap, we use the linearity of g_t to obtain $g_t(s_t)-g_t(x_t)=(1/\beta)g_t(x_{t+1}-x_t)$ which yields bound (ii) after summing over all t and dividing by t.

REFERENCES

[1] E. Siow *et al.*, "Analytics for the internet of things: A survey," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 74:1–74:36, 2018.

[2] G. Ananthanarayanan *et al.*, "Video analytics - killer app for edge computing," in *Proc. of ACM MobiSys*, 2019.

[3] S. K. Sharma and X. Wang, "Live data analytics with collaborative edge and cloud processing in wireless iot networks," *IEEE Access*, vol. 5, pp. 4621–4635, 2017.

[4] Next unit of computing. [Online]. Available: https://www.intel.com/content/www/us/en/products/boards-kits/nuc.html

[5] "Google nest cameras," [Online]: https://nest.com/cameras.

[6] Microsoft, "Azure iot edge software platform," [Online]: https://azure.microsoft.com/en-us/services/iot-edge/.

[7] H. Yu and M. J. Neely, "On the convergence time of the drift-plus-penalty algorithm for strongly convex programs," in *IEEE CDC*, 2015.

[8] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[9] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends in Optimization*, vol. 1, no. 1, pp. 1–144, 2006.

[10] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *IEEE INFOCOM*, 1998.

[11] A. L. Stolyar, "Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm," *Queueing Systems*, vol. 50, no. 4, pp. 401–457, Aug 2005.

[12] M. J. Neely, "Stability and Capacity Regions of Discrete Time Queueing Networks," *ArXiv e-prints*, Mar. 2010.

[13] M. J. Neely, "A Simple Convergence Time Analysis of Drift-Plus-Penalty for Stochastic Optimization and Convex Programs," *ArXiv e-prints*, Dec. 2014.

[14] A. Gupta, X. Lin, and R. Srikant, "Low-complexity distributed scheduling algorithms for wireless networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, pp. 1846–1859, Dec 2009.

[15] M. Kodialam and T. Nandagopal, "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," in *ACM MobiCom*, 2005.

[16] A. Nedić and A. Ozdaglar, "Approximate primal solutions and rate analysis for dual subgradient methods," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1757–1780, 2009.

[17] D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar, *Convex Analysis and Optimization*. Athena Scientific, 2003.

[18] M. Jaggi, "Revisiting Frank-Wolfe: Projection-free sparse convex optimization," in *ICML*, 2013.

[19] V. Valls and D. J. Leith, "Max-weight revisited: Sequences of nonconvex optimizations solving convex optimizations," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2676–2689, Oct 2016.

[20] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, 1992.

[21] ——, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Transactions on Information Theory*, vol. 39, no. 2, pp. 466–478, 1993.

[22] X. Lin, N. B. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE JSAC*, vol. 24, no. 8, pp. 1452–1463, Aug 2006.

[23] Y. Yi, A. Proutière, and M. Chiang, "Complexity in wireless scheduling: Impact and tradeoffs," in *ACM Mobihoc*, 2008.

[24] L. X. Bui, S. Sanghavi, and R. Srikant, "Distributed link scheduling with constant overhead," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1467–1480, Oct 2009.

[25] X. Wei and M. J. Neely, "Primal-dual frank-wolfe for constrained stochastic programs with convex and non-convex objectives," *arXiv preprint arXiv:1806.00709*, 2018.

[26] L. Jiang and J. Walrand, "Stable and utility-maximizing scheduling for stochastic processing networks," in *Conference on Communication, Control, and Computing (Allerton)*, 2009.

[27] L. Huang and M. J. Neely, "Utility optimal scheduling in processing networks," *Performance Evaluation*, vol. 68, no. 11, pp. 1002 – 1021, 2011.

[28] Y. Sahni, J. Cao, and L. Yang, "Data-aware task allocation for achieving low latency in collaborative edge computing," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[29] J. Fan *et al.*, "Deadline-aware task scheduling in a tiered iot infrastructure," in *IEEE GLOBECOM*, 2017.

[30] C. Wang *et al.*, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, Aug 2017.

[31] J. He *et al.*, "Multitier fog computing with large-scale iot data analytics for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 677–686, 2018.

[32] H. M. Raafat *et al.*, "Fog intelligence for real-time iot sensor data analytics," *IEEE Access*, vol. 5, pp. 24 062–24 069, 2017.

[33] Y. Nan *et al.*, "Adaptive energy-aware computation offloading for cloud of things systems," *IEEE Access*, vol. 5, pp. 23 947–23 957, 2017.

[34] Z. Cao *et al.*, "Enhancing mobile networks with software defined networking and cloud computing," *IEEE/ACM Trans. on Networkign*, vol. 25, no. 3, 2017.

[35] A. Destounis, G. Paschos, and I. Koutsopoulos, "Streaming big data meets backpressure in distributed network computation," in *Proc. of IEEE INFOCOM*, 2016.

[36] J. Zhang *et al.*, "Optimal control of distributed computing networks with mixed-cast traffic flows," in *Proc. of IEEE INFOCOM*, 2018.

[37] V. Valls, G. Iosifidis, and T. Salonidis, "Maximum lifetime analytics in iot networks," in *Proc. of IEEE INFOCOM*, 2019.

[38] "Georgia tech face database," http://www.anefian.com/research/face_reco.htm.