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, e.g. 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 tested 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 task can be executed at the nodes which generate the data classification, \( \mathcal{C} \) 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 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 \( b_{i,t}^{(c)} \) bits/sec. The capacity of each link \( (i, j) \) may vary over time, \( \{\mu_{ij}\}_{t=1}^{\infty} \), with average value \( \mu_{ij} \) bits/sec. Similarly, every node \( i \) has a possibly time-varying processing capacity \( \{\pi_i\}_{t=1}^{\infty} \) cycles/sec, and \( \pi_i \) is its long-term average. These random variables are uniformly upper-bounded by \( \mu_{max}, \mu_{max} \) and \( \pi_{max} \), 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:

\[
b_{i,t}^{(c)} + \sum_{j \in \mathcal{N}_i} y_{ij}^{(c)} = \sum_{j \in \mathcal{N}_i} y_{ji}^{(c)} + z_i^{(c)}, \quad i \in \mathcal{N}, c \in \mathcal{C},
\]

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 data. Also, the data routed over each link should not exceed its capacity:

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

and the processing is constrained by the computing capacity:

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

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]:

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

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

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

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 \( z_i^{(c)} \). We define the Transmission and Computation Rate Allocation problem:

\[
\begin{array}{ll}
\text{TCRA} : & \text{maximize} \quad \sum_{i \in \mathcal{N}} \sum_{c \in \mathcal{C}} U_i^{(c)}(z_i^{(c)}) \\
\text{subject to} : & (1) - (4)
\end{array}
\]

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:

\bullet (R1): The task requests \( \{b_{i,t}^{(c)}\} \), link capacities \( \{\mu_{ij}\}_{t=1}^{\infty} \), and rewards \( \{w_{i,t}^{(c)}\} \) are time-varying, with unknown mean values.

\bullet (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.:

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

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:
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 assumes 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 $^{\text{on}}$ and $^{\text{off}}$ for the optimal solution of the static and t-slot problems, respectively.

A. Preliminaries

We first define the bounded polytopes:

$$
Y = \left\{ \left( y^{(i,c)}_t \right) \in [0, \mu_{\text{max}}] \mid (4), c \in C, (i,j) \in \mathcal{E} \right\},
$$

and express our variables in a single set $X = Y \times X \subseteq \mathbb{R}^e_+$, with $e = C^2 N E$. Vector $x_t = \left( y^{(i,c)}_t, z^{(i,c)}_t \right)$ is the amount of transmitted and processed data in slot $t$. We assume that TCRA has a finite optimal solution and (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 $A x + \delta \leq 0$, where $A \in \mathbb{R}^{m \times e}$, $\delta \in \mathbb{R}^m$, $m = (CN + E + N)$; and define:

$$
f(x) = -\sum_{i \in N} \sum_{c \in C} u^{(c)}_i (z^{(c)}_i) = -w^T x,
$$

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, $f(x) = -\log(w^T x)$ to enforce some type of fairness or load-balancing. We can now rewrite TCRA as:

**Proposition 1 (Link-Processing schedules):** Any average schedule that satisfies (4) belongs to $\text{conv}(S)$.

Primal: \[ \text{minimize } x \in X \quad f(x) \text{ s.t. } g(x) = Ax + \delta \leq 0, \]

and the respective dual (concave) problem is:

DUAL: \[ \text{maximize } \lambda \geq 0 \quad h(\lambda) \triangleq \min_{x \in X} L(x, \lambda), \]

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

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

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

$$
x_t \in \arg \min_{x \in X} \left\{ f(x) + \lambda^T g(x) \right\}. \quad (6)
$$

One could use (5)-(6) to gradually approach the optimal TCRA solution $f(x^*)$; 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 $f_t(x) = -w^T x, \forall t$, where we assume that the sequence of observed precisions $w_t$ converges to the final precision parameters of TCRA, i.e., $\lim_{t \to \infty} w^{(c)}_{t,i} = w^{(c)}_{i,t}, \forall i, c$, (Assumption 2). This is because, as the nodes collect more data, they eventually achieve their maximum possible analytic performance. Also, the changes across functions are bounded, i.e., $|f_t(x) - f_{t+1}(x)| \leq \sigma_f, \forall t, x$. Similarly, we define:

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

where the perturbations (link/node capacities, and arrivals) are assumed to converge, i.e. $\lim_{t \to \infty} \delta_t = \delta$ (Assumption 3).

Then, we can define the t-slot Lagrangian and dual problem:

$$
\text{maximize } h_t(\lambda_t) \triangleq \min_{x \in X} L_t(x, \lambda_t) = \min_{x \in X} \left\{ f_t(x) + \lambda^T g_t(x) \right\}.
$$

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

$$
\frac{1}{t} \sum_{\tau=1}^t f_t(s_{\tau}) \rightarrow f(x^*) \quad \text{and} \quad \frac{1}{t} \sum_{\tau=1}^t g_t(s_{\tau}) \rightarrow g(x^*),
$$

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

1 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.

2 This in practice means that the rates $y^{(c)}_t$ 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.
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: $s_t \leftarrow u \in \arg\min_{u \in X} \ u^T \nabla f_t(x_t)$
4: $x_{t+1} \leftarrow (1 - \beta)x_t + \beta s_t$
5: end for

Fig. 2: (a): In each iteration we select an extreme point and move the average $x_t$ in that direction, until it converges within $2\eta$ 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 $\|f_t(x_t) - f_t(x)\|$ until it becomes smaller than a set bound $2\eta$. 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 $u^T \nabla f_t(x_t)$, 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 solutions $\{h_t\}_t$ to maximize the time-average $\sum_{t=1}^{T} f_t(x_t)/t$ 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:

$$X(\gamma_t, \lambda_t) := \{x \in X | h_t(\lambda_t) \leq L_t(x, \lambda_t) \leq h_t(\lambda_t) + \gamma_t\}$$

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

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

(i) $\frac{1}{t} \sum_{\tau=1}^{t} f_t(x_\tau) - f(x^0) \leq \frac{\alpha \sigma^2}{2} + \frac{1}{t} \sum_{\tau=1}^{t} \gamma_\tau + \epsilon_\tau \hat{x}_\tau + \lambda_\tau \phi_\tau$

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

where $\epsilon_\tau = w_\tau - w$, $\phi_\tau = \bar{x}_\tau - \delta$, 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 $\beta$ and $\alpha$, 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 $s_t \in S$ 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 $\{s_t\}_{t=1}^{\infty}$ which ensure that the system operation approaches the performance prescribed by the solution of TCRA. The following theorem formalizes the performance of FWDS.

---

**Algorithm 2 Approximate Dual Subgradient**

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

**Algorithm 3**

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

---

**TABLE I: Constants and Bounds**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_f$</td>
<td>$</td>
</tr>
<tr>
<td>$\sigma_g$</td>
<td>$|g_t(x)| \leq \sigma_g, \forall x \in X, t$</td>
</tr>
<tr>
<td>$\sigma_L$</td>
<td>$|h_{t+1}(\lambda_{t+1}) - h_t(\lambda_t)| \leq \sigma_L, \forall \lambda_t$</td>
</tr>
<tr>
<td>$R$</td>
<td>$|y - y| \leq R, \forall y \in X$</td>
</tr>
<tr>
<td>$\Lambda_t$</td>
<td>Upper bound on $|\lambda_t|$; see Lemma 4.</td>
</tr>
<tr>
<td>$\Lambda$</td>
<td>$\Lambda = \max_t \Lambda_t$</td>
</tr>
</tbody>
</table>

**Perturbations**

$\epsilon_t = w_{t+1} - w_t$, $d_t = \bar{d}_t - \delta$ |

**Assumption 2**

$\lim_{t \to \infty} u_{t+1}^{(c)}(\cdot, \cdot) = w_{t}^{(c)}(\cdot, \cdot), \forall t, c$ |

**Assumption 3**

$\lim_{t \to \infty} \delta_t = \delta$ |
Theorem 1: Consider the updates in Algorithm 3 with \( \alpha \) and \( \beta \) selected as per Lemma 5 (see Sec.VIII). It holds:

\[
\text{(i) } \frac{1}{t} \sum_{\tau=1}^{t} f_{\tau}(s_{\tau}) - f(x^{o}) \leq \frac{\alpha \sigma_{g}}{2} + 2\Lambda \sigma_{g} + \frac{1}{\alpha t} \sum_{\tau=1}^{t} \left( \epsilon_{\tau} \bar{x}_{\tau} + \lambda_{\tau} \phi_{\tau} + \max \{ \zeta_{\tau}, 2\eta \} \right)
\]

\[
\text{(ii) } \frac{1}{t} \sum_{\tau=1}^{t} g_{\tau}(s_{\tau}) \leq \frac{\Lambda_{t+1}}{\alpha t} (1 + \frac{1}{\beta}) \zeta_{t+1}(v_{t}, \lambda_{t+1}) - h_{t+1}(\lambda_{t+1}).
\]

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 \( \alpha, \beta \) and \( \eta \), which need to be carefully selected, namely:

\[
0 < \beta \leq 1, \quad 0 < \alpha < \frac{\beta \eta - (\beta^{2}M_{L}R^{2})/2 - \sigma_{J} - \sigma_{L} - 2\Lambda \sigma_{g}}{\sigma_{g}^{2}}
\]

see Lemma 5 for the proof and Table I for the parameters.\( ^{7} \)

It is evident that \( \alpha \) negatively affects the optimality gap, while both \( \alpha \) and \( \beta \) do not impact the constraint violation as they are amortized by \( t \). Parameter \( \beta \) however, affects \( \eta \) in the selection of \( \alpha \) (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 \( \alpha \). The first term in the summation is upper bounded by \( \epsilon_{\tau} \bar{x}_{\tau} \leq \| \epsilon_{\tau} \| \| \bar{x}_{\tau} \| \) 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 proved in Lemma 4 of the Appendix.

The last term in the summation depends on parameter \( \eta \) and is affected by \( \alpha \) and \( \beta \), as explained above, while \( \zeta_{t} \) is a decreasing sequence as shown in Lemma 5; hence the term \( \max \{ \zeta_{t}, 2\eta \} \) is replaced by \( 2\eta \) for \( \tau \) sufficiently large. Regarding the feasibility bound, we showed that it diminishes with \( \alpha \), but we have used the running average of perturbations on \( g_{\tau} \), i.e., \( \delta_{\tau} \), and not \( \delta_{\tau} \). However, it is easy to see that we can write

\[
\left\| \frac{1}{t} \sum_{\tau=1}^{t} A_{\tau} + \delta_{\tau} \right\| \leq \left\| \frac{1}{t} \sum_{\tau=1}^{t} A_{\tau} + \delta_{\tau} \right\| + \left\| t \sum_{\tau=1}^{t} \delta_{\tau} - \delta_{\tau} \right\|.
\]

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

Finally, in terms of overheads and complexity, the bad news is that FWDS requires the solution of an NP-hard problem, but the good news is that 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 \( x_{t} \in \arg \max_{x \in X} V(x) - J_{t}^{x} \), 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 identical 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 \( L_{t}(v, \lambda) \leq L_{t}(u, \lambda) + \nabla L_{t}(u, \lambda)^{T}(v - u) + 2^{-1}M_{L}\| v - u \|^{2} \) 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 $s_t \in \arg \min_{s \in S} (\nabla f(x_t) + \beta Q_t \Delta Q_t)^\top s$. 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 ${\beta^t}$ 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>
<thead>
<tr>
<th>Algorithm/Node</th>
<th>Accuracy</th>
<th>Cycles/bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBP/NUC</td>
<td>82 %</td>
<td>600</td>
</tr>
<tr>
<td>Eigen/NUC</td>
<td>74 %</td>
<td>520</td>
</tr>
<tr>
<td>Fisher/IoT</td>
<td>42 %</td>
<td>360</td>
</tr>
</tbody>
</table>

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 $100 \times 100$ 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, 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 $ho_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 $w_i(c) = w_i(c), \forall t$, and also $f_i(x) = f(x), \forall t$. 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 $\alpha$ and $\beta$, 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 $\alpha$ and $\beta$ values. Note that the selected values for $\alpha$ and $\beta$ not affect the convergence bounds of FWDS both directly (see Theorem 1) and indirectly through $\eta$.

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 $\alpha$. However, increasing $\alpha$ results

<table>
<thead>
<tr>
<th>Algorithm/Node</th>
<th>Accuracy</th>
<th>Cycles/bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBP/NUC</td>
<td>82 %</td>
<td>600</td>
</tr>
<tr>
<td>Eigen/NUC</td>
<td>74 %</td>
<td>520</td>
</tr>
<tr>
<td>Fisher/IoT</td>
<td>42 %</td>
<td>360</td>
</tr>
</tbody>
</table>
in slightly better convergence performance. This counterintuitive behavior is due to the intricate relations between $\alpha, \beta, \eta$. Reducing $\alpha$ requires decreasing $\beta$, which in turn might increase $\eta$ 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 $\alpha, \beta$ 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 $\tilde{s}$ at random and compare $s^\top \nabla_x L_t(x_t, \lambda_t)$ to $\tilde{s}^\top \nabla_x L_t(x_t, \lambda_t)$. 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.

Next we evaluate the average network utility obtained by each algorithm (Fig. 5a). We consider different scenarios where we modify the number of NUCs, 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
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:

$$f(x_{t+1}) \leq f(x_t) + \beta \nabla_x f(x_t)^\top (s_t - x_t) + \beta^2 M R^2 / 2.$$  

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

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. $\xi_{ij} + \sum_{(k,l) \notin I(i,j)} \xi_{kl} = 1$, 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.,

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

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 $Q_\lambda = \{ \lambda \geq 0 | h_t(\lambda) \geq h_t(\lambda) \}$ is bounded. That is, for any $\lambda \in Q_\lambda$ we have

$$\|\lambda\| \leq \left( f_t(x_s) - h_t(\lambda) \right) / q_t, \text{ where } q_t = \min_{i} \{-g_{t,(i)}(x_s)\}.$$  

Proof: $\forall \lambda \in Q_\lambda$ it is $h_t(\lambda) \leq h_t(\lambda) = \min_{i} \{ f_t(x) + \lambda^\top g_t(x) \} \leq f_t(x_s) + \lambda^\top g_t(x_s) = f_t(x_s) + \sum_{i=1}^{m} \lambda(i) g_{t,(i)}(x_s)$.

Since $g_{t,(i)}(x_s) < 0$, and $\lambda(i) \geq 0$ we have:

$$\min_{i} \{-g_{t,(i)}(x_s)\} \sum_{i=1}^{m} \lambda(i) \leq f_t(x_s) - h_t(\lambda) \Rightarrow$$

$$\|\lambda\| \leq \frac{1}{q_t} \left( f_t(x_s) - h_t(\lambda) \right).$$  

Lemma 4 (Bounded Multipliers): The multiplier sequence generated by Algorithm 2 is bounded, i.e., $\|\lambda_t\| \leq \Lambda_t \triangleq 2f_t(x_s) - h_t(\lambda^0) + \max \left\{ \|\lambda_t\|, \frac{f_t(x_s) - h_t(\lambda^0)}{q_t} + \frac{\alpha \sigma^2}{2q_t} + \alpha \sigma_g \right\}$

Proof: We define the superlevel set $Q_\alpha = \{ \lambda \geq 0 | h_t(\lambda) \geq h_t(\lambda^0) - \alpha \sigma^2 / 2 \}$ and firstly prove that $\|\lambda_t - \lambda^0\| \leq \max \left\{ \|\lambda_t - \lambda^0\|, \frac{f_t(x_s) - h_t(\lambda^0)}{q_t} + \frac{\alpha \sigma^2}{2q_t} + \alpha \sigma_g + \|\lambda^0\| \right\}$

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): $h_t(\lambda_t) \geq h_t(\lambda^0) - \alpha \sigma^2 / 2$. By the dual update rule:

$$\|\lambda_t + \lambda^0\|^2 \leq \|\lambda_t + \alpha g_t(x_t(1))\|^2 - \lambda^0\|^2$$

$$\leq \|\lambda_t\|^2 + \alpha \sigma_g$$

$$\leq \frac{1}{q_t} \left( f_t(x_s) - h_t(\lambda^0) + \frac{\alpha \sigma^2}{2} + \|\lambda^0\| + \alpha \sigma_g \right)$$

where the last inequality follows from Lemma 3 for $Q_\alpha$, hence (8) holds. Case (ii): $h_t(\lambda_t) < h_t(\lambda^0) - \alpha \sigma^2 / 2$. We have:

$$\|\lambda_t + \lambda^0\|^2 \leq \|\lambda_t + \alpha g_t(x_t(1)) - \lambda^0\|^2$$

$$\leq \|\lambda_t - \lambda^0\|^2 + 2\alpha g_t(x_t(1))^\top (\lambda_t - \lambda^0) + \alpha^2 \sigma_g^2$$

$$\leq \|\lambda_t - \lambda^0\|^2 - 2\alpha h_t(\lambda^0) - h_t(\lambda_t)) + \alpha^2 \sigma_g^2$$

$$\leq \|\lambda_t - \lambda^0\|^2,$$

where the last inequality holds by the assumption of this case, making (8) true. Now we can rewrite (8) as $\|\lambda_t - \lambda^0\| \leq \max \left\{ \|\lambda_t - \lambda^0\|, \frac{f_t(x_s) - h_t(\lambda^0)}{q_t} + \frac{\alpha \sigma^2}{2q_t} + \alpha \sigma_g + \|\lambda^0\| \right\}$

$$\leq \|\lambda^0\| + \max \left\{ \|\lambda_t\|, \frac{f_t(x_s) - h_t(\lambda^0)}{q_t} + \frac{\alpha \sigma^2}{2q_t} + \alpha \sigma_g \right\}$$.  

FIG. 6: Network congestion in tasks per commodity and task execution delay CDF with $N = 40, C = 20$.
Since $||\lambda_t|| \leq ||\lambda_t - \lambda^*|| + ||\lambda^*||$ we obtain $||\lambda_t|| \leq 2||\lambda^*|| + \max \{||\lambda_1||, \frac{1}{q_t}(h(x) - h(\lambda_1^*) + \frac{\sigma_t^2}{\sigma_t^2} + \alpha \sigma_t^2)\}$, and applying Lemma 3 for $\lambda = \lambda^*$ 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.

$$h_t(\lambda) = \min_{x \in X} L_t(x, \lambda) = \min_{x \in X} f(x) + e^T x + \lambda^T (Ax + \delta - \tilde{d}_t - \delta) = \min_{x \in X} f(x) + (e^T B + \lambda^T)(Ax + \delta) + \lambda^T \phi_t - e^T \tilde{d}B \delta$$

$$= h(B^T e + \lambda) + \lambda^T \phi_t - e^T \tilde{d}B \delta. \tag{9}$$

where $B \in \mathbb{R}^{c \times m}$ such that $BA = I_c$, and $I_c$ is the unit matrix of dimension $c$. By using the above bound on $h_t(\lambda)$, we can write:

$$h(\mu_t) \leq h(\lambda_t) + (\mu_t - \lambda_t)^T h'(\lambda_t),$$

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

$$h_t(\lambda_t) \leq h(\lambda_t) + (B^T \epsilon_t)^T (A\tilde{x}_t + \delta) + \lambda^T \phi_t - e^T \tilde{d}B \delta$$

$$\leq h(\lambda_t) + e^T \tilde{x}_t + \lambda^T \phi_t,$$

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

$$f(x^\circ) = h(\lambda^*) \geq \frac{1}{t} \sum_{t=1}^{T} h_t(x, \lambda_t) \geq \frac{1}{t} \sum_{t=1}^{T} h_t(\lambda_t) - e^T \tilde{x}_t - \lambda^T \phi_t$$

$$\geq \frac{1}{t} \sum_{t=1}^{T} f_t(x_t) + \lambda^T g_t(x_t) - \gamma_t - e^T \tilde{x}_t - \lambda^T \phi_t$$

$$\geq \frac{1}{t} \sum_{t=1}^{T} f_t(x_t) + \lambda^T g_t(x_t) - \gamma_t - e^T \tilde{x}_t - \lambda^T \phi_t,$$

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

$$\frac{1}{T} \sum_{t=1}^{T} f_t(x_t) - f(x^\circ) \leq - \frac{1}{T} \sum_{t=1}^{T} \lambda^T g_t(x_t) - \gamma_t - e^T \tilde{x}_t - \lambda^T \phi_t. \tag{10}$$

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

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

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

$$-2\alpha \sum_{t=1}^{T} g_t(x_t) \leq \alpha^2 \sum_{t=1}^{T} ||g_t(x_t)||^2,$$

where $-||\lambda_{t+1}||^2$ and $||\lambda_{t}||^2$ 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 $||g_t(x_t)|| \leq \sigma_g \forall t$, for all $x \in X_t$, and diving across by $2\alpha t$ we obtain:  

$$\frac{1}{t} \sum_{t=1}^{T} \lambda^T g_t(x_t) \leq \frac{\alpha^2}{\sigma_g^2}.$$

By using the above bound on (10) we prove the first claim.

From the dual update rule recursive expansion we have $\lambda_{t+1} \geq \lambda_t + \alpha \sum_{t=1}^{T} g_t(x_t)$. By dropping $\lambda_t$, dividing by $\alpha t$, and taking the norms:

$$\frac{1}{t} \sum_{t=1}^{T} g_t(x_t) \leq \frac{||\lambda_{t+1}||}{\alpha t}, \tag{11}$$

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 $0 < \alpha < \frac{\eta}{(\beta^2 M_L^2 R^2/4) - \sigma_g^2}$ and $\beta \in (0, 1)$. $M_L$ is a constant independent of $\alpha, \beta$. Algorithm 3 guarantees that if $L_t(v_t, \lambda_t) - h_t(\lambda_t) \geq \eta; t$:

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

**Proof:** Based on Lemma 4, the sequence of vectors $\{\lambda_t\}$ is bounded. Hence, $L_t(\cdot, \lambda_t)$ is an $M_L$-smooth convex function of $v$ for any parameter $\lambda$. We consider two cases. Case (i): $L_t(v_t, \lambda_t) - h_t(\lambda_t) > \eta$, where recall that $h_t(\lambda_t) \leq L_t(v_t, \lambda_t), \forall t$. Due to its smoothness and the FW rule applied on (7), we have $L_t(v_{t+1}, \lambda_t) - L_t(v_t, \lambda_t) \leq \beta \nabla_v L_t(v_t, \lambda_t)^T (s_t - v_t) + \frac{\beta^2 M_L^2 R^2}{2}, \tag{12}$

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

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

Now, it holds that $L_{t+1}(v_{t+1}, \lambda_{t+1}) - L_{t+1}(v_{t+1}, \lambda_t) = f_{t+1}(v_{t+1}) - f_t(v_{t+1}) + \lambda_{t+1}^T g_{t+1}(v_{t+1}) - \lambda_t^T g_{t+1}(v_{t+1})$$

$$= f_{t+1}(v_{t+1}) - f_t(v_{t+1}) + \lambda_t^T g_{t+1}(v_{t+1}) - g_t(v_{t+1}) + \alpha g_t(v_t)^T g_{t+1}(v_{t+1}) \leq \sigma_f + 2 \lambda ||\sigma_g + \alpha \sigma_g^2,$$

where we used Cauchy-Schwarz and triangle inequalities. Adding (13) and (14), we get: $L_{t+1}(v_{t+1}, \lambda_{t+1}) \leq L_t(v_t, \lambda_t) - \beta \eta + \beta^2 M_L^2 R^2 + \sigma_f + 2 \lambda ||\sigma_g + \alpha \sigma_g^2,$

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 $\alpha, \beta$ 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 $u^T \nabla_v L_t(v_t, \lambda_t)$, the term $\nabla_v L_t(v_t, \lambda_t)^T (s_t - v_t)$ is non-positive and hence can be dropped from (12):

$$L_t(v_{t+1}, \lambda_t) - L_t(v_t, \lambda_t) \leq (\beta^2 M_L^2 R^2)/2. \tag{15}$$
Adding (14) to (15) and using \( L_t(v_t, \lambda_t) - h_t(\lambda_t) \leq \eta \) we end up with
\[
L_{t+1}(v_{t+1}, \lambda_{t+1}) - h_{t+1}(\lambda_{t+1}) \leq \eta + \frac{\beta^2 M_L R^2 t}{2} + |\alpha + 2Mg + \alpha^2| \leq \eta + \beta + \left( L_{t+1}(v_{t+1}, \lambda_{t+1}) - h_{t+1}(\lambda_{t+1}) \right) \leq 
\]
where we used the assumption of this case for the last inequality. Substituting \( h_{t+1}(\lambda_{t+1}) \) in both sides and rearranging terms yields
\[
L_{t+1}(v_{t+1}, \lambda_{t+1}) - h_{t+1}(\lambda_{t+1}) \leq \eta + \frac{\beta^2 M_L R^2 t}{2} + |\alpha + 2Mg + \alpha^2|, \]
where we used \(| h_{t+1}(\lambda_{t+1}) - h_t(\lambda_t) | \leq \sigma_L \), and the stated \( \alpha, \beta \), completing the second case.

**Proof of Theorem 1.** By the stated selection of \( \alpha, \beta \), 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 \( \zeta_t = L_{t+1}(v_{t+1}, \lambda_{t+1}) - h_{t+1}(\lambda_{t+1}) \). By applying Lemma 2 with \( \gamma_t = \max\{\zeta_t, 2\eta\} \) we obtain
\[
\frac{1}{t} \sum_{i=1}^{t} f_i(x_i) - f(x^\star) \leq \frac{\alpha \sigma_t^2}{2} + \frac{1}{t} \sum_{i=1}^{t} \max\{\zeta_t, 2\eta\} + t_\star \gamma_t + \lambda_{2} \phi_t,
\]
Combining the above with (16) we obtain the claimed bound (i). For the feasibility gap, we use the linearity of \( g_i \) to obtain
\[
g_i(s_t) - g_i(x_t) = (1/\beta) g_i(x_{t-1} - x_t)
\]
which yields bound (ii) after summing over all \( t \) and dividing by \( t \).

**REFERENCES**


