Learning-NUM: Network Utility Maximization With Unknown Utility Functions and Queueing Delay

Xinzhe Fu and Eytan Modiano, *Fellow, IEEE*

***Abstract*— Network Utility Maximization (NUM) studies the problems of allocating traffic rates to network users in order to maximize the users' total utility subject to network resource constraints. In this paper, we propose a new NUM framework, Learning-NUM, where the users' utility functions are unknown apriori and the utility function values of the traffic rates can be observed only after the corresponding traffic is delivered to the destination, which means that the utility feedback experiences *queueing delay*. The goal is to design a policy that gradually learns the utility functions and makes rate allocation and network scheduling/routing decisions so as to maximize the total utility obtained over a finite time horizon T. In addition to unknown utility functions and stochastic constraints, a central challenge of our problem lies in the queueing delay of the observations, which may be unbounded and depends on the decisions of the policy. We first show that the expected total utility obtained by the best dynamic policy is upper bounded by the solution to a static optimization problem. Without the presence of feedback delay, we design an algorithm based on the ideas of gradient estimation and Max-Weight scheduling. To handle the feedback delay, we embed the algorithm in a parallel-instance paradigm to form a policy that achieves $\tilde{O}(T^{3/4})$-regret, i.e., the difference between the expected utility obtained by the best dynamic policy and our policy is in $\tilde{O}(T^{3/4})$. Furthermore, we extend our policy to deal with the case where the utility observations are noisy and show that it achieves $\tilde{O}(T^{7/8})$-regret. Finally, to demonstrate the practical applicability of the Learning-NUM framework, we apply it to three application scenarios including database query, job scheduling and video streaming. We further conduct simulations on the job scheduling application to evaluate the empirical performance of our policy.**

***Index Terms*— Optimization methods, queueing analysis.**

I. Introduction

NETWORK Utility Maximization (NUM) has been a central problem in networking research for decades and has become a standard framework for making intelligent network resource allocation decisions. It has found a wide range of applications such as congestion control in the Internet [1]–[3], power allocation in wireless networks [4] and job scheduling in cloud computing [5], [6].

As a network optimization paradigm, NUM studies the problems of user traffic admission control to maximize the users' total utility subject to network resource constraints. Previous works in NUM can be classified into two categories: static and stochastic. In the static approach [1]–[3], [7], [8], the traffic rates are modeled as flow variables, the bandwidth constraints are modeled as network flow constraints, and the analysis focuses on the convergence rates of the optimization algorithms. In the stochastic approach [4]–[6], [9], [10], the traffic rates are determined by the time-average admitted traffic, the resource constraints are captured by the long-term stability of the stochastic queueing networks and the analysis focuses on the tradeoff between the long-term average utility and queue length.

Regardless of the differences in modeling and analysis, previous NUM results rest on a key assumption that the utility functions of network users are known. This is justified when the utility functions are simply optimization proxies for network performance criteria such as fairness [1], [2], [9]. However, when the utility represents more concrete quantities such as power and energy consumption of network links [4], [10], user satisfaction of the quality of video streaming services [12], [13] and completion quality of computation jobs on the clouds [5], [6], [14], often we do not have prior knowledge of the utility functions, i.e., the functional relationship between the traffic rate and its corresponding utility value is unknown in advance. As a concrete example, if we consider a job of training a machine learning model on a cloud computing platform, with the traffic rate in this case representing the computation resource we allocate to the job and the utility value corresponding to the performance of the trained model, then there is no available function that describes the relationship between the traffic rate and the utility value [5], [6].

In this paper, we propose a new NUM framework, Learning-NUM, where the utility functions are unknown but their values can be learned over the process of decision making. Specifically, we consider a time-varying stochastic queueing network in discrete time, which captures both wireline and wireless networks. There are K users, where user k has a concave utility function f_k that is initially unknown to the network operator. Each user has a corresponding source-destination pair in the network. At every time t, for each user k, the network operator injects a "job" of size $r_k(t)$ from the user's source to be delivered to the user's destination. The job size in our framework resembles the admitted traffic rate in the traditional NUM formulation. We will explain the connection between the two notions in Section II. Next, the operator chooses a network action that controls the routing and scheduling, which further determines the queue dynamics of the network. Finally, the utility value $(f_k(r_k))$ of a job (of size r_k) can only be observed

Manuscript received 2 December 2021; revised 11 April 2022; accepted 5 June 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor L. Huang. Date of publication 23 June 2022; date of current version 20 December 2022. This work was supported in part by NSF under Grant CNS-1524317 and in part by the Office of Naval Research (ONR) under Grant N00014-20-1-2119. The early version of this paper appeared in the Proceedings of ACM MboiHoc 2021 [DOI: 10.1145/3466772.3467031]. *(Corresponding author: Xinzhe Fu.)*

The authors are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: xinzhe@mit.edu).

Digital Object Identifier 10.1109/TNET.2022.3182890

after the job gets delivered to the destination as feedback from the destination. We consider both the case of noiseless feedback and the case of noisy feedback, where the utility observations are noisy, i.e., we observe $f_k(r_k) + \epsilon$ where ϵ is a zero-mean noise.

We study the problem of designing a policy that jointly determines the job sizes and network actions based on the utility function values learned from observations. We define the utility achieved by a policy as the total utility of the jobs delivered by a finite time horizon T. This definition naturally enforces network resource constraints as the undelivered jobs in the queues at time T are not counted towards the utility. We seek to design a policy with regret sublinear to T, where regret [11] is defined as the gap between the expected utility of the policy and that of the optimal policy that has full knowledge of the utility functions in advance. As a first step, we establish that the expected utility achieved by the optimal (dynamic) policy is upper bounded by T times the optimal value of a static optimization problem, whose objective is the sum of the (unknown) utility functions and the constraints are implicitly given by the capacity region of the network. This result provides an important insight that a policy achieves low regret if it can closely track the solution to the static optimization problem.

While solving an optimization problem with unknown objective function is a common challenge faced in the online convex optimization literature [20], [21], our problem is further complicated by the facts that the constraints, which essentially enforce network stability, are stochastic and unknown in advance (See Section III for details). Thus, they cannot be handled by techniques in online optimization that require the feasibility region to be known in advance [21]. Moreover, the utility value can be observed only after the delivery of the job, which, in a First-In-First-Out network, happens after the delivery of the jobs injected before it. This means that the feedback in our problem experiences **queueing-style delay** that may be unbounded and depends on the decisions of the policy. Such delay evades existing techniques in the literature as they typically assume (deterministically or stochastically) bounded or decision-independent delay [25]–[28]. Finally, in our problem, the network operator also needs to make routing and scheduling decisions in addition to the job size decisions.

To deal with unknown utility functions and stochastic constraints, we combine the ideas of gradient sampling [15] and max-weight scheduling (back-pressure routing) [16] to propose an online scheduling algorithm that works for the Learning-NUM problem without feedback delay. We next embed the algorithm into a parallel-instance paradigm to obtain a scheduling policy that can handle the queueing-style feedback delay and achieve $\tilde{O}(T^{3/4})$-regret.[1] Furthermore, we show that when the utility observations are noisy, the scheduling policy can still achieve a sublinear $\tilde{O}(T^{7/8})$-regret. Finally, we show how to apply our framework to applications including database query [18], job scheduling [19] and video streaming [12], [13]. We further empirically evaluate the performance of the proposed policy through simulations on job scheduling scenarios.

The rest of the paper is organized as follows. The model and formal definitions of the Learning-NUM framework are presented in Section II. In Section III, we prove the upper bound on the optimal expected utility. In Section IV, we propose the online scheduling algorithm and the parallel-instance paradigm for the Learning-NUM framework. We further illustrate several applications of Learning-NUM in Section V. The empirical performance of the online scheduling policy is evaluated in Section VI. Finally, we conclude the paper with some future directions in Section VII.

II. Model and Problem Formulation

In this section, we specify the general network model and set up the framework of network utility maximization with unknown utility functions. We consider a network $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with $\mathcal{V}$ being the set of nodes and $\mathcal{E}$ being the set of directed links. For each node $i \in \mathcal{V}$, we will denote its set of outgoing neighbors by $\mathcal{N}_i$. There are K classes of users in the network. Each user k corresponds to a job class (also denoted by k), and is mapped to one source-destination pair (s_k, d_k). Multiple job classes can be mapped to the same source-destination pair. Source s_k sends class-k jobs that get delivered to d_k through the network. We will refer to the jobs sent from s_k destined to d_k as class-k traffic. Each node $i \in \mathcal{V}$ has a queue Q_i^k that buffers the incoming class-k traffic of node i. The network operates in discrete time $t = 1, \ldots, T$, where T is the specified time horizon. At each time t, the network is in state $\omega(t) \in \mathcal{W}$, with $\mathcal{W}$ denoting the set of possible network states. In concrete applications, the network states may correspond to channel states of links, service states of servers, or simply a placeholder when the network is static with only one state. We assume $\omega(t)$'s is a sequence of i.i.d. random element with $\mathbb{P}(\omega(t) = \omega) = p(\omega)$. However, the distribution of $\omega(t)$ is unknown to the network operator.

A. Traffic Model and Network Dynamics

At each time t, the network operator first observes the current network state $\omega(t)$. It next chooses job size $r_k(t)$ for each class and sends a job of size $r_k(t)$ to the buffer $Q_{s_k}^k$, where $r_k(t)$ is a real value that satisfies $0 \leq r_k(t) \leq B$. The job size corresponds to the amount of admitted traffic at a time slot. For example, as we will demonstrate in Section V, in video streaming, the job size represents the resolution of a video chunk sent to the user. We adopt this discrete notion of job size rather than the continuous notion of traffic rate in the traditional NUM framework because job size is more suitable for our finite-horizon discrete-time framework. Finally, the network operator chooses a network action $\boldsymbol{x}(t) \in \mathcal{X}$ that incorporates the routing and scheduling decisions of the network. The feasible set of actions $\mathcal{X}$ can be discrete or continuous. For each $\boldsymbol{x} \in \mathcal{X}$, under network state ω, we use $A_{ij}^k(\omega, \boldsymbol{x})$ to denote the offered transmission rate on link (i, j) for class k, i.e., the amount of class-k traffic that can be sent from node i to node j. Each link transmits traffic in a First-In-First-Out (FIFO) basis. $A_{ij}^k(\omega, \boldsymbol{x})$'s are known to the network operators, and are assumed to be non-negative and upper-bounded by A, and lower-bounded by a non-negative

[1] $\tilde{O}(\cdot)$ hides logarithmic factors of T.

constant for all ω and $\boldsymbol{x}$. Based on the definitions above, the dynamics of the queues can be written following the Lindley recursion:

$$Q^k_{s_k}(t+1) = [Q^k_{s_k}(t) + r_k(t) - \sum_{j\in\mathcal{N}_{s_k}} A^k_{s_kj}(\omega(t),\boldsymbol{x}(t))]^+, \quad (1)$$

$$Q^k_i(t+1) = [Q^k_i(t) + \sum_{j:i\in\mathcal{N}_j} A^k_{ji}(\omega(t),\boldsymbol{x}(t)) - \sum_{j\in\mathcal{N}_i} A^k_{ij}(\omega(t),\boldsymbol{x}(t))]^+, \quad (2)$$

and $Q^k_{d_k}(t) = 0$ for all k. We define $\Lambda(\omega) := \{(A(\omega,\boldsymbol{x}))^k_{ij}, \boldsymbol{x}\in\mathcal{X}\}$ as the set of feasible transmission rate vectors under network state ω. Note that the network operator can observe $\Lambda(\omega(t))$ at t but does not know the distribution of $\omega(t)$.[2] Finally, we define $Cap(\mathcal{G})$ as the set of feasible rate vectors $(r_1,\ldots,r_K)$, i.e., there exists $\{\lambda(\omega)\}^k_{ij} \in Conv(\Lambda(\omega))$ with

$$\forall k,\ r_k \le \sum_{\omega\in\mathcal{W}}\sum_{j\in\mathcal{N}_{s_k}} p(\omega)\lambda(\omega)^k_{ij},$$
$$\forall i\in\mathcal{V}, \sum_{\omega\in\mathcal{W}}\sum_{j:i\in\mathcal{N}_j} p(\omega)\lambda(\omega)^k_{ji} \le \sum_{\omega\in\mathcal{W}}\sum_{j\in\mathcal{N}_i} p(\omega)\lambda(\omega)^k_{ij},$$

where $conv(\Lambda(\omega))$ is the convex hull of $\Lambda(\omega)$. $Cap(\mathcal{G})$ resembles the network capacity region in the traditional infinite-horizon network utility maximization problem [9], i.e., the set of traffic rate vectors that can be supported by the network. However, as we consider a finite-horizon setting here, the capacity region here does not exactly characterize the set of job-size vectors that the network can support. Nevertheless, the close connection between the two concepts will be revealed in **Section III**. Furthermore, to prevent trivializing the problem, we enforce the condition on $Cap(\mathcal{G})$ that it has non-empty interior, i.e., there exists $\eta > 0$ such that $(\eta,\ldots,\eta)\in Cap(\mathcal{G})$.

B. Utility Model

Each job class (user) k is associated with some underlying utility function f_k. The utility functions are initially unknown. When a class-k job of size r_k gets delivered to d_k, we observe and obtain utility of value $f_k(r_k)$. We will also consider the case where the utility observations are noisy in Section IV-D. Note that this implies that the utility feedback of each job experiences queueing-style delay, i.e., the time from injecting a job into the network to observing its utility value is equal to the time that the job spends in the network (queues). See Figure 1 for further illustration of the feedback delay.

For each traffic class k, we assume its underlying utility function has the following properties:

1) f_k is monotonically non-decreasing and concave.
2) f_k is bounded on $[0,B]$, i.e., $\forall r\in[0,B]$, $f_k(r)\le D$ for some constant D.
3) f_k is L-Lipschitz continuous, i.e., $\forall r_1,r_2\in[0,B]$, $|f_k(r_2)-f_k(r_1)|\le L\cdot|r_2-r_1|$.

[2]We assume that $\Lambda(\omega)$ is downward closing in the sense that if $\boldsymbol{\lambda}\in\Lambda(\omega)$, then any vector $\boldsymbol{\lambda}'$ that equals zero in one coordinate and equals $\boldsymbol{\lambda}$ in all other coordinates is also in Λ.

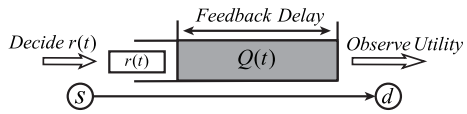


Fig. 1. A single-queue example illustrating the queueing-style feedback delay in the Learning-NUM framework.

C. Problem Formulation

Given the network $\mathcal{G}$ and time-horizon T, we seek to find a scheduling policy that determines the sizes of the jobs sent by the sources and the network actions that maximizes the total utility obtained at the end of the horizon T. Formally, let Π be the collection of *admissible policies* that make scheduling decisions at time t based on observations obtained before time t. Policies in Π do not have access to the underlying utility functions or the distribution of network state, but can learn them through observations of utility values and instantiated network state. We further let $\bar{\Pi}$ be the collection of all policies, including non-admissible policies that know the underlying utility functions and the network state distribution. For a policy π, we define $U(\pi,T)$ to be the total utility obtained from jobs that are delivered by time T under π. Note that $U(\pi,T)$ is a random variables, the randomness of which comes from the time-varying network state and the (possible) inherent randomness of the scheduling policy. We adopt the notion of regret from the online learning literature as the measure of quality of scheduling policies.

Definition 1 Regret: The regret of scheduling policy π is defined as

$$R(\pi,T) = \sup_{\pi^*\in\bar{\Pi}} \mathbb{E}[U(\pi^*,T)] - \mathbb{E}[U(\pi,T)],$$

The regret $R(\pi,T)$ measures the gap between the expected utility obtained under π and the maximum utility achieved by any (even non-admissible) policy for the given instance.

Based on the above preliminaries, we formally pose the problem of network utility maximization with unknown utility functions, which we will refer to as the Learning-NUM problem, as one that asks for an admissible scheduling policy with low regret.

Definition 2 The Learning-NUM Problem: The Learning-NUM problem seeks an admissible policy π with sublinear regret, i.e., $\lim_{T\to\infty}\frac{R(\pi,T)}{T}=0$.

Remark: (i). A policy that has sublinear regret is *asymptotically optimal*, since the gap between time-average utility achieved by the policy and that of the optimal goes to zero. (ii). Although the regret does not explicitly depend on the queue backlogs at the end of the horizon T, the queue backlogs are implicitly accounted for, since the utility $U(\pi,T)$ does not include the jobs that are still in the queue at time T.

III. Upper Bound on the Optimal Utility

If the utility functions are known in advance, Learning-NUM becomes a finite-horizon stochastic optimization problem. Typically, the optimal policy for the problem is a dynamic programming-based policy that is intractable and difficult to

compare to. Therefore, in this section, we relate the expected utility obtained by the best policy in $\bar{\Pi}$ to the optimal value of a static optimization problem, which motivates the design and analysis of the admissible scheduling policy we propose. The optimization problem $\mathcal{P}$ is defined as follows:

$$\mathcal{P}: \quad \max_{\{r\}_k} \sum_{k=1}^{K} f_k(r_k) \tag{3}$$

$$\textbf{s.t.} \quad (r_1, \ldots, r_K) \in Cap(\mathcal{G}) \tag{4}$$

$$r_k \in [0, B], \quad \forall k. \tag{5}$$

Intuitively, the optimization problem characterizes a static version of the Learning-NUM problem over the job-size variables. The decision variables $\{r_k\}$'s can be interpreted as average size of jobs of class k. $\mathcal{P}$ seeks to maximize the total utility obtained by $\{r_k\}$ such that the vector lies inside the network capacity region. Note that $Cap(\mathcal{G})$ is a convex set over $\{r_k\}$. Hence, $\mathcal{P}$ is a convex optimization problem.

Based on the optimization problem $\mathcal{P}$, we are ready to state the main result of this section, i.e., the optimal value of $\mathcal{P}$ multiplied by the time horizon upper-bounds the maximum expected utility over all policies in $\bar{\Pi}$.

Theorem 1: $\sup_{\pi^* \in \bar{\Pi}} \mathbb{E}[U(\pi^*, T)] \leq T \cdot OPT(\mathcal{P})$.

Proof: The main idea of the proof is that, for any given policy, we first take certain averages of the job sizes of each traffic class and then show that the averages satisfy the constraints of $\mathcal{P}$. Next, by the concavity of the underlying utility functions, their corresponding value of the objective function is no less than the expected utility of the policy.

For ease of notations, we prove the theorem for deterministic policies in $\bar{\Pi}$. The case of randomized policies follows similarly. Consider an arbitrary policy $\pi^* \in \bar{\Pi}$ and a sample path θ of its execution on the problem instance. For each traffic class k, let $r_k(1, \theta), \ldots, r_k(T, \theta)$ be the size of the jobs specified by π^* over the time horizon T. Define $\tilde{r}_k(t, \theta) = r_k(t, \theta)$ if the t-th job is delivered by time T and $\tilde{r}_k(t, \theta) = 0$ otherwise. Let $\boldsymbol{x}(t, \theta)$ be the network action chosen by π^* at t and let $\omega(t, \theta)$ be the network state at t under θ. Based on the utility model we have that the utility achieved by π^* on sample path θ is equal to $\sum_{k=1}^{K} \sum_{t=1}^{T} f_k(\tilde{r}_k(t, \theta))$. Let $\bar{r}_k(\theta) = \frac{1}{T} \sum_{t=1}^{T} \tilde{r}_k(t, \theta)$. Since the underlying utility functions are concave, we have

$$\sum_{k=1}^{K} \sum_{t=1}^{T} f_k(\tilde{r}_k(t, \theta)) \leq T \sum_{k=1}^{K} f_k(\bar{r}(\theta)). \tag{6}$$

Furthermore, let $\tilde{A}_{ij}^k(\omega(t, \theta), \boldsymbol{x}(t, \theta))$ be the realized transmission rate on link (i, j) for class-k at t. The realized transmission $\tilde{A}_{ij}^k$ is equal to the offered transmission A_{ij}^k when the queue length is greater than the offered transmission, and the realized transmission is smaller otherwise. From the queue dynamics (Equations 1 and 2), we obtain that

$$\forall k, T\bar{r}_k(\theta) \leq \sum_{t=1}^{T} \sum_{j \in \mathcal{N}_{s_k}} \tilde{A}_{s_k j}^k(\omega(t, \theta), \boldsymbol{x}(t, \theta)), \tag{7}$$

$$\forall i, \sum_{t=1}^{T} \sum_{j: i \in \mathcal{N}_j} \tilde{A}_{ji}^k(\omega(t, \theta), \boldsymbol{x}(t, \theta)) \leq \sum_{t=1}^{T} \sum_{j \in \mathcal{N}_i} \tilde{A}_{ij}^k(\omega(t, \theta), \boldsymbol{x}(t, \theta)). \tag{8}$$

Define $\hat{p}_\theta(\omega), \omega \in \mathcal{W}$ as the empirical distribution of ω,

$$\hat{p}_\theta(\omega) := \frac{\sum_{t=1}^{T} \mathbb{1}\{\omega(t, \theta) = \omega\}}{T}.$$

It follows from (7), (8) that for each $\omega \in \mathcal{W}$, there exists $(\tilde{\lambda}(\omega, \theta))_{ij}^k \in Conv(\Lambda(\omega))$ such that

$$\forall k, \quad \bar{r}_k(\theta) \leq \sum_{\omega \in \mathcal{W}} \sum_{j \in \mathcal{N}_{s_n}} \hat{p}_\theta(\omega) \tilde{\lambda}_{s_k j}^k(\omega, \theta),$$

$$\forall i, \sum_{\omega \in \mathcal{W}} \sum_{j: i \in \mathcal{N}_j} \hat{p}_\theta(\omega) \tilde{\lambda}_{ji}^k(\omega, \theta) \leq \sum_{\omega \in \mathcal{W}} \sum_{j \in \mathcal{N}_i} \hat{p}_\theta(\omega) \tilde{\lambda}_{ij}^k(\omega, \theta).$$

Moreover, as $\Lambda(\omega)$ is downward-closing, we further have that there exists $(\lambda(\omega, \theta))_{ij}^k \in Conv(\Lambda(\omega))$ such that

$$\forall k, \quad \bar{r}_k(\theta) = \sum_{\omega \in \mathcal{W}} \sum_{j \in \mathcal{N}_{s_k}} \hat{p}_\theta(\omega) \lambda_{s_k j}^k(\omega, \theta),$$

$$\forall i, \sum_{\omega \in \mathcal{W}} \sum_{j: i \in \mathcal{N}_j} \hat{p}_\theta(\omega) \lambda_{ji}^k(\omega, \theta) = \sum_{\omega \in \mathcal{W}} \sum_{j \in \mathcal{N}_i} \hat{p}_\theta(\omega) \lambda_{ij}^k(\omega, \theta).$$

Taking expectation over θ, we have $(\mathbb{E}_\theta[\bar{r}_1(\theta)], \ldots, \mathbb{E}_\theta[\bar{r}_1(\theta)]) \in Cap(\mathcal{G})$. Moreover, it is easy to see that $0 \leq \mathbb{E}_\theta[\bar{r}_k(\theta)] \leq B$ for all k. Therefore, the vector $(\mathbb{E}_\theta[\bar{r}_1(\theta)], \ldots, \mathbb{E}_\theta[\bar{r}_1(\theta)])$ is feasible to $\mathcal{P}$. Hence, $OPT(\mathcal{P}) \geq \sum_{k=1}^{K} f_k(\mathbb{E}_\theta[\bar{r}_k(\theta)])$. Invoking the concavity of f_k's again, by Jensen's inequality, we have for all k, $f_k(\mathbb{E}_\theta[\bar{r}_k(\theta)]) \geq \mathbb{E}_\theta[f_k(\bar{r}_k(\theta))]$. Combining this with (6), we obtain

$$\begin{aligned} OPT(\mathcal{P}) &\geq \sum_{k=1}^{K} f_k(\mathbb{E}_\theta[\bar{r}_k(\theta)]) \\ &\geq \mathbb{E}\left[\sum_{k=1}^{K} f_k(\bar{r}_k(\theta))\right] \\ &\geq \frac{1}{T}\mathbb{E}\left[\sum_{k=1}^{K} \sum_{t=1}^{T} f_k(\tilde{r}_k(t, \theta))\right], \end{aligned} \tag{9}$$

which concludes the proof. □

It is worth pointing out that Theorem 1 does not imply that the optimal policy is a static one that assigns the job sizes according to the solution to the optimization problem $\mathcal{P}$. Such a policy would not achieve an expected utility of $T \cdot OPT$ since the expected number of jobs delivered is typically less than T. Indeed, due to the stochastic network dynamics, a portion of the jobs will still remain in the queues by the end of the time horizon. Despite that, the theorem does provide the insight that a policy achieves low regret if it can closely approximate the solution to $\mathcal{P}$ at each time slot. As the objective function of $\mathcal{P}$ is unknown, the problem has similar flavor to online/zeroth-order optimization [20], [23], [24]. However, in the Learning-NUM problem we are facing two additional challenges. First, the feasibility region in the Learning-NUM is stochastic and not explicitly given as the distribution of network states is unknown. Thus, we cannot rely on method that requires the feasibility region to be known in advance [21]. Second, the

queueing-style delay of the feedback compromises the policy's ability to adjust based on utility observations. As the delay is action-dependent and may be unbounded, it also poses more stringent requirement on controlling the network queue lengths.

IV. Online Scheduling Policy

In this section, we introduce the scheduling policy we propose for the Learning-NUM framework – the Parallel Gradient Sampling Max-Weight (P-GSMW) policy. The P-GSMW policy is composed of embedding an algorithm (called Gradient Sampling Max-Weight, GSMW) that makes job-size and scheduling decisions based on immediate feedback (no delay) into a parallel-instance paradigm that handles the feedback delay. The GSMW algorithm essentially combines the ideas of drift-plus-penalty optimization [17], gradient sampling [15], and Max-Weight scheduling. The parallel-instance paradigm invokes multiple parallel instances of the GSMW algorithm such that each instance essentially runs in a no-delay setting. In the following, we first introduce the GSMW algorithm, and then combine it with the parallel-instance paradigm. Finally, we provide discussion on the challenges posed by the feedback delay.

A. The GSMW Algorithm

In the presentation of the GSMW algorithm, we assume a no-delay setting, i.e., the utility values of the jobs can be observed immediately after job-size decision. We will handle the feedback delay with the parallel-instance paradigm in subsequent sections.

The GSMW algorithm (**Algoritm 1**) maintains a virtual job size variable $\hat{r}_k$ for each class k and utilizes queue lengths to update the virtual job size variables and network actions. The $\hat{r}_k$'s are updated once every two slots, which essentially divides the time horizon into epochs of size two (without loss of generality, we assume the horizon T to be even). For simplicity of notations, we will assume that the network state remains unchanged for each epoch and refer to an epoch as a time slot indexed by $t \in \{1, \ldots, T\}$ for the rest of the paper, i.e., at each slot, we need to make scheduling decision and job-size decision for two incoming jobs of each class.[3]

At each slot $t \in \{1, \ldots, T\}$, the network action is chosen according to a Max-Weight-like rule (Line 3). The decisions on job size are made based on the virtual job size variables at the corresponding epoch. The updates of virtual job size variables are determined by gradient estimates of the utility functions and queue lengths. Since the utility functions are unknown, GSMW constructs the gradient estimates using observations of function values. Specifically, at slot t, each source s_k injects a first job of size $\hat{r}_k(\tau)+\delta$ and a second job of size $\hat{r}_k(\tau)-\delta$ for each $k \in n$ and obtains the feedback of value $f_k(\hat{r}_k(\tau)+\delta)$ and $f_k(\hat{r}_k(\tau)-\delta)$ (Lines 5, 6). The two feedback values obtained are combined to form the gradient estimate of f_k at $\hat{r}_k(\tau)$ (Line 7). The gradient estimate is then fed into the update of the virtual variable $\hat{r}_k$ (Line 10). The projection step $\mathcal{P}_{[\delta, B-\delta]}$ of Line 10, defined as the projection on to interval $[\delta, B-\delta]$ by the Euclidean norm, is to ensure that $\hat{r}_k(\tau)+\delta$ and $\hat{r}_k(\tau)-\delta$ always lie in the domain $[0, B]$. Here, V controls the relative weights of gradient and queue length while α determines the step size.

[3]This assumption is purely made for notational convenience. Our results can be straightforwardly adapted to the original setting without the assumption.

Algorithm 1 The Gradient Sampling Max-Weight Algorithm.

Input: Network $\mathcal{G}(\mathcal{V}, \mathcal{E})$, parameters V, δ, α

1: **Initialize:** $\boldsymbol{x}(0) \in \mathcal{X}, \hat{r}_k(0) = \delta$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $\boldsymbol{x}(t) := \arg\max_{\boldsymbol{x} \in \mathcal{X}} \sum_{i,j \in V} \sum_{k=1}^{K} A_{ij}^k(\omega(t), \boldsymbol{x})[Q_i^k(t) - Q_j^k(t)]$
4: **for** $k = 1, \ldots, K$ **do**
5: s_k injects job of size $\hat{r}_k(t) + \delta$ and observes $f_k(\hat{r}_k(t) + \delta)$.
6: s_k injects job of size $\hat{r}_k(t) - \delta$ and observes $f_k(\hat{r}_k(t) - \delta)$.
7: $\hat{\nabla} f_k(\hat{r}_k(t)) := \frac{f_k(\hat{r}_k(t)+\delta) - f_k(\hat{r}_k(t)-\delta)}{2\delta}$
8: Update queue lengths according to $r_k(t), \boldsymbol{x}(t)$.
9: **for** $k = 1, \ldots, K$ **do**
10: $\hat{r}_k(t+1) := \mathcal{P}_{[\delta, B-\delta]}\left[\hat{r}_k(t) + \frac{1}{\alpha}(V \cdot \hat{\nabla} f_k(\hat{r}_k(t)) - Q_{s_k}^k(t))\right]$

In the no-delay setting, the GSMW algorithm with suitable choices of parameter values can achieve $\tilde{O}(\sqrt{T})$-regret, which will be formally presented in Theorem 2. As the analysis in the no-delay setting can be considered as a special case of the original setting (with feedback delay) as we will show in the next section, we omit the proof of Theorem 2.

Theorem 2: In the no-delay setting, the Gradient Sampling Max-Weight policy π_{GSMW} achieves $\tilde{O}(\sqrt{T})$-regret by setting $\alpha = 2KT/\eta, V = \sqrt{T}, \delta = T^{-1/2}$.

Remark: in the traditional framework of stochastic network optimization where the utility functions are known, the drift-plus-penalty/max-weight approach [9], [10] can achieve an $O(V)$-$O(1/V)$ trade-off in the gap of time-average utility and queue length as $t \to \infty$. Setting the parameter V therein to $\sqrt{T}$, the asymptotic $O(V)$-$O(1/V)$ trade-off can be translated to $O(\sqrt{T})$-regret in the finite horizon framework. Theorem 2 shows that one can achieve the same regret by replacing the penalty term with gradient-sampling when the utility functions are unknown.

1) Challenges Posed by Feedback Delay: In the original setting of the Learning-NUM problem where the observations experience queueing-style feedback delay, the GSMW algorithm no longer works. The main reason is that the update of the job-size variables (Line 10 of Algorithm 1) cannot be executed every time slot, as the gradient estimate $\hat{\nabla} f_k(\hat{r}_k(t))$ cannot be formed without the corresponding utility observations. We now give two straightforward adaptations of the GSMW algorithms that are implementable under the presence of feedback delay, and argue at an intuitive level that they both lead to regrets that grow linearly with the time horizon T. This highlights the challenges posed by the feedback delay, and justifies the necessity of having a more sophisticated scheme that achieves sublinear regret which will be introduced in the next section.

One possible adaptation is to use an "episodic approach," i.e., keep the job-size variables unchanged (for an episode of multiple slots) until the utility observations needed become available, and update job-size variables once every episode. This approach would cause the sizes of the jobs (traffic injected to the network) not be able to adjust timely with respect to the queue lengths,as Line 10 only gets executed once every episode, which will further lead to larger queue lengths. As the feedback delay is essentially proportional to the queue lengths, larger queue lengths would further increase the length of the episodes, which further reduce the frequency and timeliness of the updates. This will result in a "positive feedback loop" that makes the algorithm suffer from linear regret.

Another possible method is to use old gradients, e.g., we execute Line 10 every time slot, but use the most recently available gradient estimates. This makes the algorithm adjusts according to queue length every time slot, and thus can maintain the queue length bound of GSMW. However, the feedback delay (queue lengths) is still non-trivial and cannot be bounded by a constant independent of T, which will result in a large bias in the gradient estimates used in the updates. Such bias in the gradient estimates will compromise the utility regret, which leads to the algorithm having linear regret.

Finally, a third candidate is to stay idle (i.e. setting the job sizes as 0) when the feedback is delayed. This will also lead to linear regret due to the waste of network resources and incoming jobs caused by staying idle. Indeed, even when the network is lightly loaded, due to the stochasticity in arrivals and services, there will be a constant probability that the incoming jobs experience queueing delay. This implies that staying idle will lead to a constant fraction of the total network resources (over the whole time horizon) being wasted, leading to a linear regret.

2) Relation to Feedback Delay in Online Learning: There have been several works in the literature that consider various online learning problems with feedback delay [25]–[28]. Previous works have considered multi-armed bandit problems and online convex optimization problems where the feedback delay is independent of the decisions [25], [26], and multi-armed bandit problems where the feedback delay is arm-dependent [27], [28]. The key distinction of the feedback delay in the L-NUM problem is that it is controlled by our decisions in a continuous fashion, while previous works assume the feedback delay is deterministically or stochastically bounded apriori (and essentially cannot be controlled). Furthermore, as the unfinished jobs in the queues do not contribute to the total utility, in the L-NUM problem, the feedback delay (i.e. queue lengths) directly affects the total utility and the regret. Hence, unlike in previous online-learning problems where the feedback delay only affects the learning process, in the L-NUM problem the feedback delay affects both the learning and the utility aspects of the network. Therefore, on one hand, we cannot over-aggressively reduce the feedback delay to facilitate learning the utility functions. For example, setting the feedback delay to be bounded in $O(1)$ would unnecessarily restrict the decision region of job sizes and lead to linear regret. On the other hand, it is still necessary to control the feedback delay. For example, having a feedback delay of order T would also lead to linear regret as both the learning process and the job completion are hindered by the queueing delay.

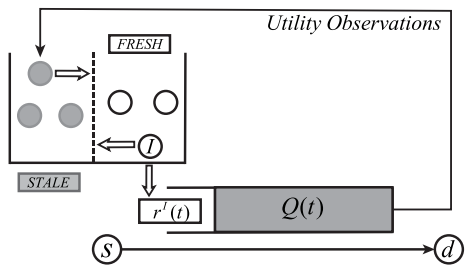


Fig. 2. A single-queue example of the parallel-instance paradigm.

B. The Parallel-Instance Paradigm

In order to handle the feedback delay, we design a parallel-instance paradigm that encapsulates the GSMW algorithm, and forms the Parallel-instance GSMW (P-GSMW) policy. The details of the P-GSMW policy are shown in **Algorithm 2**. Similar to the GSMW algorithm, the network action $\boldsymbol{x}(t)$ at each time slot is still determined by the Max-Weight rule. The key difference is that the paradigm maintains a set of parallel instances of the GSMW algorithm, which we will refer to as the instance reservoir $\mathcal{I}$. Each instance can be in one of the two possible status: FRESH and STALE. FRESH status means that the instance has obtained the corresponding utility feedback and can perform updates on the virtual job-size variables (Line 5 of **Algorithm 2**); STALE status means that the instance is still waiting for utility feedback. We use $\{r_k^I(t)\}$ to denote the virtual job size variables maintained by instance I. When we need to make job size decisions, if there is a FRESH instance available in the reservoir, we "invoke" the instance by performing updates and deciding on the job sizes based on the updated virtual job-size variables of the instance (Line 5 of **Algorithm 2**). If there are multiple FRESH instances, we select one arbitrarily. We then change the instance's status to STALE (Line 7). If there is no FRESH instance available, we initialize a new instance, add it to $\mathcal{I}$ (Lines 9 and 10). The virtual job-size variables of instances that are not invoked remains unchanged (Line 12). Upon delivery of jobs, we observe utility values and feed them to the corresponding instances. If an instance has all the utility observations available for the jobs injected when it was last invoked, we change its status from "STALE" to "FRESH" (Line 15). We further illustrate the parallel-instance paradigm with a single-queue example in Figure 2.

Remark: (i). Our parallel-instance paradigm has a similar flavor to the technique in [25] for online learning with delayed feedback. However, the observation delay in the Learning-NUM framework may be unbounded and is action-dependent, which is more general than the bounded, decision-independent delay considered in [25]. (ii). Line 9 of **Algorithm 2**, i.e., creating a new instance when there is no fresh instance available may seem counter-intuitive, as the absence of new instances is an indication of the network being congested while creating a new instance may add to the congestion. However, this argument overlooks the benefits of creating a new instance: First, it reduces the probability of the event that there is no fresh instance available when job

size decisions need to be made. Second, the job size decisions made based on this new instance will be adjusted based on both utility and queue lengths feedback, which enables them to achieve a small regret.

Algorithm 2 The Parallel-Instance GSMW Policy.

Input: Network $\mathcal{G}(\mathcal{V},\mathcal{E})$, parameters V,δ,α, instance reservoir $\mathcal{I}$

for $t=1,2,\ldots,T$ **do**
  $\boldsymbol{x}(t):=\arg\max_{\boldsymbol{x}\in\mathcal{X}}\sum_{i,j\in V}\sum_{k=1}^{K}A_{ij}^{k}(\omega(t),\boldsymbol{x})[Q_i^k(t)-Q_j^k(t)]$
  if There exists a FRESH instance $I_t\in\mathcal{I}$ **then**
    for $k=1,\ldots,K$ **do**
      $\hat{r}_k^{I_t}(t):=\mathcal{P}_{[\delta,B-\delta]}\left[\hat{r}_k^{I_t}(t-1)+\frac{1}{\alpha}(V\cdot\hat{\nabla}f_k(\hat{r}_k^{I_t}(t-1))-Q_{s_k}^k(t))\right]$
      s_k injects job of size $\hat{r}_k^{I_t}(t)+\delta$ and another job of size $\hat{r}_k^{I_t}(t)-\delta$.
      Change the status of I_t to STALE.
    else
      Create a new instance I_t
      For each k, initialize $\hat{r}_k^{I_t}(t):=\delta$, and s_k injects job of size $\hat{r}_k^{I_t}(t)+\delta$ and another job of size $\hat{r}_k^{I_t}(t)-\delta$
  Update queue lengths according to $r_k(t),\boldsymbol{x}(t)$.
  $\{\hat{r}_k^J(t)\}:=\{\hat{r}_k^J(t)\}$ for $J\in\mathcal{I},J\neq I_t$.
  Collect utility observations from delivered jobs and form gradient estimates $\hat{\nabla}f_k(\hat{r}_k^I(t)):=\frac{f_k(\hat{r}_k^I(t)+\delta)-f_k(\hat{r}_k^I(t)-\delta)}{2\delta}$
  for STALE instance $I\in\mathcal{I}$ **do**
    Change the status of I to FRESH if it has obtained all outstanding gradient estimates.

C. Policy Analysis

In this section, we analyze the regret achieved by the P-GSMW policy π_{P-GSMW}. The main result is presented Theorem 3.

Theorem 3: The Parallel-Instance Gradient Sampling Max-Weight policy π_{P-GSMW} achieves $\tilde{O}(T^{3/4})$ regret by setting $\alpha=2K\sqrt{T}/\eta, V=T^{1/4},\delta=T^{-1/2}$., i.e., $R(\pi_{GSMW},T)=\tilde{O}(T^{3/4})$.

Remark: In the no-delay setting, the GSMW algorithm achieves a regret of order $\tilde{O}(\sqrt{T})$, which matches the established regret lower bound $\Omega(\sqrt{T})$ [21]. Under the queueing-style feedback delay, the P-GSMW policy achieves $\tilde{O}(T^{3/4})$ which is higher than $\tilde{O}(\sqrt{T})$. This raises the question whether the delay of Learning-NUM fundamental increases the difficulty of the problem, i.e., a lower bound better than $\Omega(\sqrt{T})$ can be shown, or that there exists algorithm for Learning-NUM that has regret better than $\tilde{O}(T^{3/4})$. We leave this as a future direction.

The rest of the section is devoted to proving Theorem 3.

1) Preliminary Results: As we focus on bounding the regret with respect to T, we will use C to represent a generic constant that does not depend on T. Note that C may depend on parameters such as A,B,D,L, and the C's that appear in different equations might not be equal. For simplicity of notation, we will use $\hat{r}_k^I(t)$ or $\hat{r}_k(t)$ to denote the virtual job-size variable used at time t (which suppresses the dependence of the invoked instance at t on the time t). We first lay out some preliminary results that will be useful in the subsequent analysis. The proofs of most lemmas will be deferred to Appendix A.

To begin with, we show a key upper bound on the regret of π^{P-GSMW}.

Lemma 1: Let $\{r^*\}_k$ be the optimal solution to $\mathcal{P}$,

$$\begin{aligned}R(\pi_{P-GSMW},T)&\leq 2\mathbb{E}\left[\sum_{t=1}^{T}\sum_{k=1}^{K}f_k(r_k^*)-f_k(\hat{r}_k(t))\right]\\&\quad+C\sum_{i\in V}\sum_{k=1}^{K}\mathbb{E}[Q_i^k(T)]+CT\delta.\end{aligned}$$

Lemma 1 shows that an upper bound on the regret essentially consists of two terms (as the third term $T\delta$ is in $O(\sqrt{T})$ and can be ignored without affecting the regret analysis). The first term, $\mathbb{E}\left[\sum_{t=1}^{T}\sum_{k=1}^{K}f_k(r_k^*)-f_k(\hat{r}_k(t))\right]$, which will be referred to as *Utility Regret*, captures the cumulative difference in terms of utility between the policy's decisions and the optimal solution to $\mathcal{P}$. The second term, $\sum_{i\in V}\sum_{k=1}^{K}\mathbb{E}[Q_i^k(T)]$. which will be referred to as *Queueing Regret*, captures the unfinished jobs in the queues at the end of the time horizon. We will show that under the P-GSMW policy, both the utility regret and the queueing regret are in $\tilde{O}(T^{3/4})$, from which, the theorem follows.

Next, we establish some auxiliary results on the gradient estimates.

Lemma 2: For all k,t, $\hat{\nabla}f_k(\hat{r}_k^I(t))\leq L$ with probability 1.

Proof: The lemma follows straightforwardly from the Lipschitz continuity of the underlying utility functions. □

We next show that the gradient estimate is unbiased with respect to a smoothed version of f_k, which is defined as $\tilde{f}_k(r)=\frac{1}{2\delta}\int_{-\delta}^{\delta}f_k(r+z)\mathrm{d}z$. Note that by definition $\tilde{f}_k$ is also concave and Lipschitz-continuous. Moreover, by the concavity and Lipschitz continuity of f_k, for all $r\in[\delta,B-\delta]$, $f_k(r)-C\delta\leq\tilde{f}_k(r)\leq f_k(r)$ [15].

Lemma 3: For all k,t, $\hat{\nabla}f_k(\hat{r}_k(t))=\nabla\tilde{f}_k(\hat{r}_k(t))$.

Proof: The lemma follows from the Fundamental Theorem of Calculus. □

Finally, we establish three basic properties of the updates of the P-GSMW policy. The first involves the update of virtual job size variables, the second considers the Max-Weight rule of choosing actions and the third deals with the queue dynamics.

Lemma 4: For each t, let I be the instance invoked at time t, we have for any $\{r\}_k$ with $r_k\in[\delta,B-\delta]$

$$\begin{aligned}&\sum_{k=1}^{K}\left[V\hat{\nabla}f_k(\hat{r}_k^I(t-1))(r_k-\hat{r}_k^I(t-1))\right]\\&\quad+\sum_{k=1}^{K}\left[Q_{s_k}^k(t)\hat{r}_k^I(t)\right]\\&\leq\sum_{k=1}^{K}\left[Q_{s_k}^k(t)r_k+\alpha[(\hat{r}_k^I(t-1)-r_k)^2-(\hat{r}_k^I(t)-r_k)^2]+C\right]\end{aligned}$$

Lemma 5: At every time slot t, for any $\boldsymbol{x} \in \mathcal{X}$, and for all $\{r\}_k$,

$$\begin{aligned}
&\sum_{k=1}^{K} \sum_{i\in V, i\neq s_k} Q_i^k(t) \\
&\quad \left[\sum_{j:i\in\mathcal{N}_j} A_{ji}^k(\omega(t),\boldsymbol{x}(t)) - \sum_{j\in\mathcal{N}_i} A_{ij}^k(\omega(t),\boldsymbol{x}(t))\right] \\
&\quad + \sum_{k=1}^{K} Q_{s_k}^k(t)\left[r_k - \sum_{j\in\mathcal{N}_{s_k}} A_{s_kj}^k(\omega(t),\boldsymbol{x}(t))\right] \\
&\leq \sum_{k=1}^{K} \sum_{i\in V, i\neq s_k} Q_i^k(t) \\
&\quad \left[\sum_{j:i\in\mathcal{N}_j} A_{ji}^k(\omega(t),\boldsymbol{x}) - \sum_{j\in\mathcal{N}_i} A_{ij}^k(\omega(t),\boldsymbol{x})\right] \\
&\quad + \sum_{k=1}^{K} Q_{s_k}^k(t)\left[r_k - \sum_{j\in\mathcal{N}_{s_k}} A_{s_kj}^k(\omega(t),\boldsymbol{x})\right].
\end{aligned}$$

Proof: By rearranging the terms, we recover exactly the right-hand-side of Line 2 of **Algorithm 2**. The lemma then follows from the construction of the Max-Weight update rule. Note that the inequality holds for all $\{r\}_k$ since the terms involving $\{r\}_k$ do not affect the maximization. □

Lemma 6: For each k, t, recall that $\hat{r}_k(t) = \hat{r}_k^I(t)$ for the invoked instance I.

$$\begin{aligned}
&Q_{s_k}^k(t+1)^2 - Q_{s_k}^k(t)^2 \\
&\quad \leq 4Q_{s_k}^k(t)\left[\hat{r}_k(t) - \sum_{j\in\mathcal{N}_{s_k}} A_{s_kj}^k(\omega(t),\boldsymbol{x}(t))\right] + C, \quad (10)
\end{aligned}$$

and for each $i \in V, k, i \neq s_k, d_k$,

$$\begin{aligned}
&Q_i^k(t+1)^2 - Q_i^k(t)^2 \\
&\quad \leq 4Q_i^k(t)\left[\sum_{j:i\in\mathcal{N}_j} A_{ji}^k(\omega(t),\boldsymbol{x}(t)) \right. \\
&\qquad \left. - \sum_{j\in\mathcal{N}_i} A_{ij}^k(\omega(t),\boldsymbol{x}(t))\right] + C. \quad (11)
\end{aligned}$$

2) Queueing Regret: In this section, we bound the queueing regret by providing a bound on the expected queue size at T. To do so, we will first use $\sum_{i\in V}\sum_{k=1}^{K} Q_i^k(t)^2$ as a Lyapunov function and prove that the Lyapunov function has expected conditional negative drift, which combined with a result on discrete stochastic process from [17], leads to a bound on $\sum_{i\in V}\sum_{k=1}^{K} Q_i^k(t)$ both in expectation and with high probability.

Define $\boldsymbol{Q}(t)$ to be the vector that includes all the queues $\{Q(t)\}_i^k$ as coordinates and $||\cdot||$ as the Euclidean norm. By Lemma 6, we have

$$\begin{aligned}
&||\boldsymbol{Q}(t+1)||^2 - ||\boldsymbol{Q}(t)||^2 \\
&= \sum_{k=1}^{K} Q_{s_k}^k(t+1)^2 - Q_{s_k}^k(t)^2 + \sum_{i,k} Q_i^k(t+1)^2 - Q_i^k(t)^2 \quad (12)\\
&\leq 4\sum_{k=1}^{K} Q_{s_k}^k(t)\left[\hat{r}_k(t) - \sum_{j\in\mathcal{N}_{s_k}} A_{s_kj}^k(\omega(t),\boldsymbol{x}(t))\right] + C \quad (13)\\
&\quad + 4\sum_{i,k} Q_k^n(t)\left[\sum_{j:i\in\mathcal{N}_j} A_{ji}^k(\omega(t),\boldsymbol{x}(t)) \right. \\
&\qquad \left. - \sum_{j\in\mathcal{N}_i} A_{ij}^k(\omega(t),\boldsymbol{x}(t))\right]. \quad (14)
\end{aligned}$$

Next, we prove a conditional drift argument on $||\boldsymbol{Q}(t)||^2$ under the P-GSMW policy.

Lemma 7: There exists $\epsilon > 0$ such that under the P-GSMW policy,

$$\begin{aligned}
&\mathbb{E}[||\boldsymbol{Q}(t+1)||^2 - ||\boldsymbol{Q}(t)||^2 \mid \boldsymbol{Q}(t)] \\
&\leq -\epsilon \sum_{i\in V}\sum_{k=1}^{K} Q_i^k(t) + C\sqrt{T}.
\end{aligned}$$

Lemma 7 establishes that $||\boldsymbol{Q}(t)||^2$ tends to decrease when the queue length is significantly larger than $O(\sqrt{T})$. To bound the queueing regret based on this, we use the following drift lemma for stochastic processes from [17]. We will not need the full generality of the lemma as it provides expectation and with-high-probability bound on stochastic processes that satisfy multi-slot drift condition, but we only need to deal with single-slot drift.

Lemma 8: [17] Let $\{Z(t), t \geq 0\}$ be a discrete time stochastic process adapted to a filtration $\{\mathcal{F}(t), t \geq 0\}$ with $Z(0) = 0$ and $\mathcal{F}(0) = \{\emptyset, \Omega\}$. Suppose there exists an integer $t_0 > 0$, real constants $\theta > 0$, $\delta_{max} > 0$ and $0 < \xi \leq \delta_{max}$ such that

$$|Z(t+1) - Z(t)| \leq \delta_{max} \quad (15)$$

$$\mathbb{E}[Z(t+t_0) - Z(t) \mid \mathcal{F}(t)] \leq t_0\delta_{\max},\ if Z(t) < \theta \quad (16)$$

$$\mathbb{E}[Z(t+t_0) - Z(t) \mid \mathcal{F}(t)] \leq -t_0\zeta,\ if Z(t) \geq \theta. \quad (17)$$

hold for all $t \in \{1, 2, \ldots, \}$, then

$$\begin{aligned}
\mathbb{E}[Z(t)] &\leq \theta + t_0\delta_{max} \\
&\quad + t_0\frac{4\delta_{max}^2}{\xi}\log\frac{8\delta_{max}^2}{\xi^2}, \forall t \in \{1, 2, \ldots, \} \quad (18)
\end{aligned}$$

and

$$\forall\ 0 < \mu < 1, \mathbb{P}(Z(t) \geq z) \leq \mu, \forall t \in \{1, 2, \ldots, \}, \quad (19)$$

where $z = \theta + t_0\delta_{max} + t_0\frac{4\delta_{max}^2}{\xi}\log\frac{8\delta_{max}^2}{\xi^2} + t_0\frac{4\delta_{max}^2}{\xi}\log\frac{1}{\mu}$.

Continuing from Lemma 7, since $\sum_{n\in V,k} Q_n^k(t) \geq ||\boldsymbol{Q}(t)||$ (as l_1 norm is no smaller than the Euclidean norm), we have

$$\mathbb{E}[||\boldsymbol{Q}(t+1)||^2 - ||\boldsymbol{Q}(t)||^2 \mid \boldsymbol{Q}(t)] \leq -\epsilon||\boldsymbol{Q}(t)|| + C\sqrt{T}.$$

It follows that

$$\mathbb{E}[||\boldsymbol{Q}(t+1)||^2 \mid \boldsymbol{Q}(t)] \le ||\boldsymbol{Q}(t)||^2 - \epsilon||\boldsymbol{Q}(t)|| + C\sqrt{T} \le (||\boldsymbol{Q}(t)|| - \epsilon)^2 \quad \text{when } ||\boldsymbol{Q}(t)|| > C\sqrt{T}.$$

It follows that when $||\boldsymbol{Q}(t)|| > C\sqrt{T}$,

$$\mathbb{E}[||\boldsymbol{Q}(t+1)|| \mid \boldsymbol{Q}(t)] \le \sqrt{\mathbb{E}[||\boldsymbol{Q}(t+1)||^2 \mid \boldsymbol{Q}(t)]} \le ||\boldsymbol{Q}(t)|| - \epsilon.$$

Further, since $||\boldsymbol{Q}(t+1)|| - ||\boldsymbol{Q}(t)|| \le ||\boldsymbol{Q}(t+1) - \boldsymbol{Q}(t)|| \le C$. Hence, invoking Lemma 8 with $t_0 = 1$, $\theta = C\sqrt{T}, \delta_{max} = C, \zeta = C$, we obtain that $\mathbb{E}[||\boldsymbol{Q}(t)||] \le \tilde{O}(T^{1/2})$ for all t. By Cauchy-Schwarz inequality, $\mathbb{E}[\sum_{i\in V}\sum_{k=1}^{K} Q_i^k(t)] \le N|V| \cdot \mathbb{E}[||\boldsymbol{Q}(\tau)||] \le \tilde{O}(T^{1/2})$ for all t. Furthermore, by union bound, we also have that there exists a constant C such that with probability at least $1 - 1/T$, $\sum_{i\in V}\sum_{k=1}^{K} Q_i^k(t) \le C\sqrt{T}\log T$.

With the analysis above, we summarize the result on queueing regret in the following theorem.

Theorem 4: Under P-GSMW, $\forall t = 1, \ldots, T$, $\sum_{i\in V}\sum_{k=1}^{K} Q_i^k(t) \le \tilde{O}(\sqrt{T})$ *in expectation and with high probability. In particular,* $\mathbb{E}[\sum_{i\in V}\sum_{k=1}^{K} Q_i^k(T)] \le \tilde{O}(\sqrt{T})$.

3) Utility Regret: In this section, we bound the utility regret term. We will use $\sum_{t,k}$ as a short-hand for $\sum_{t=1}^{T}\sum_{k=1}^{T}$. and $\sum_{i,k}$ as a shorthand for $\sum_{i\in V}\sum_{k=1}^{K}$. We first decompose the utility regret into four components as follow

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t,k} f_k(r_k^*) - f_k(\hat{r}_k(t))\right] \\
&= \mathbb{E}\left[\sum_{t,k} f_k(r_k^*) - f_k(\hat{r}_k^*)\right] \\
&\quad + \mathbb{E}\left[\sum_{t,k} f_k(\hat{r}_k^*) - \tilde{f}_k(\hat{r}_k^*)\right] \\
&\quad + \mathbb{E}\left[\sum_{t,k} \tilde{f}_k(\hat{r}_k^*) - \tilde{f}_k(\hat{r}_k(t))\right] \\
&\quad + \mathbb{E}\left[\sum_{t,k} \tilde{f}_k(\hat{r}_k(t)) - f_k(\hat{r}_k(t))\right], \qquad (20)
\end{aligned}$$

where $(\hat{r}_1^*, \ldots, \hat{r}_K^*)$ is the vector that maximizes $\sum_{k=1}^{K} f_k(r_k)$ subject to $(r_1, \ldots, r_K) \in Cap(\mathcal{G})$ and $\forall k,\ r_k \in [\delta, B - \delta]$, i.e., the optimal solution to $\mathcal{P}$ restricting to each $r_k \in [\delta, B - \delta]$. As f_k is Lipschitz continuous, by Lemma 10 (in Appendix A), we have $\sum_{k=1}^{K} f_k(r_k^*) - f_k(\hat{r}_k^*) \le C\delta$. Further, $\sum_{k=1}^{K} f_k(\hat{r}_k^*) - \tilde{f}_k(r_k^*) \le KL\delta$. Since f_k is concave, $\sum_{k=1}^{K} \tilde{f}_k(\hat{r}_k(t)) - f_k(\hat{r}_k(t)) \le 0$. It follows that

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t,k} f_k(r_k^*) - f_k(\hat{r}_k(t))\right] \\
&\le \mathbb{E}\left[\sum_{t,k} \tilde{f}_k(\hat{r}_k^*) - \tilde{f}_k(\hat{r}_k(t))\right] + C\sqrt{T}
\end{aligned}$$

Hence, we can focus on bounding $\mathbb{E}\left[\sum_{t,k} \tilde{f}_k(\hat{r}_k^*) - \tilde{f}_k(\hat{r}_k(t))\right]$.

Again, starting from Lemma 4 and plugging in $\{\hat{r}^*\}_k$, we have

$$\begin{aligned}
&\sum_{k=1}^{K}\left[V\hat{\nabla} f_k(\hat{r}_k^I(t-1))(\hat{r}_k^* - \hat{r}_k^I(t-1))\right] \\
&\qquad + \sum_{k=1}^{K}\left[Q_{s_k}^k(t)\hat{r}_k^I(t)\right] \\
&\le \sum_{k=1}^{K}\left[Q_{s_k}^k(t) r_k + \alpha[(\hat{r}_k^I(t-1) - \hat{r}_k^*)^2 \right. \\
&\qquad \left. -(\hat{r}_k^I(t) - \hat{r}_k^*)^2] + C.\right]
\end{aligned}$$

Again, multiplying both sides by two and adding the same terms on both sides lead to

$$\begin{aligned}
&\sum_{k=1}^{K}\left[V\hat{\nabla} f_k(\hat{r}_k^I(t-1))(\hat{r}_k^* - \hat{r}_k^I(t-1))\right] \\
&\qquad + \sum_{k=1}^{K} Q_{s_k}^k(t)\left[\hat{r}_k(t) - \sum_{j\in\mathcal{N}_{s_k}} A_{s_k j}^k(\omega(t), \boldsymbol{x}(t))\right] \\
&\qquad + \sum_{i,k} Q_i^k(t)\left[\sum_{j: i\in\mathcal{N}_j} A_{ji}^k(\omega(t), \boldsymbol{x}(t)) \right. \\
&\qquad \left. - \sum_{j\in\mathcal{N}_i} A_{ij}^k(\omega(t), \boldsymbol{x}(t))\right] \qquad (21) \\
&\le \sum_{k=1}^{K} Q_{s_k}^k(\tau)\left[\hat{r}_k^* - \sum_{j\in\mathcal{N}_{s_k}} A_{s_k j}^k(\omega(t), \boldsymbol{x}(t))\right] \\
&\qquad + \sum_{i,k} Q_i^k(t)\left[\sum_{j: i\in\mathcal{N}_j} A_{ji}^k(\omega(t), \boldsymbol{x}(t)) \right. \\
&\qquad \left. - \sum_{j\in\mathcal{N}_i} A_{ij}^k(\omega(t), \boldsymbol{x}(t))\right] \\
&\qquad + \sum_{k=1}^{K} \alpha[(\hat{r}_k^I(t) - \hat{r}_k^*)^2 - (\hat{r}_k^I(t+1) - \hat{r}_k^*)^2] + C \qquad (22)
\end{aligned}$$

By (14), for the left-hand-side of (22),

$$\begin{aligned}
&\sum_{k=1}^{K} Q_{s_k}^k(t)\left[\hat{r}_k(t) - \sum_{j\in\mathcal{N}_{s_k}} A_{s_k j}^k(\omega(t), \boldsymbol{x}(t))\right] \\
&\qquad + \sum_{i,k} Q_i^k(t)\left[\sum_{j: i\in\mathcal{N}_j} A_{ji}^k(\omega(t), \boldsymbol{x}(t)) \right. \\
&\qquad \left. - \sum_{j\in\mathcal{N}_i} A_{ij}^k(\omega(t), \boldsymbol{x}(t))\right] \\
&\ge \frac{||\boldsymbol{Q}(t+1)||^2 - ||\boldsymbol{Q}(t)||^2}{4} + C. \qquad (23)
\end{aligned}$$

As $(\hat{r}_1^*, \ldots, \hat{r}_K^*) \in Cap(G)$ by definition, for each ω, there exists a set of real numbers $\{a(\omega, \boldsymbol{x}), \boldsymbol{x} \in \mathcal{X}\}$,

$0 \leq a(\omega, \boldsymbol{x}) \leq 1$ and $\sum_{\boldsymbol{x}\in\mathcal{X}} a(\boldsymbol{x}, \omega) = 1$ such that[4]

$$\forall k,\ \hat{r}_k^* \leq \sum_{\omega\in\mathcal{W}} p(\omega) \sum_{j\in\mathcal{N}_{s_k}} \sum_{\boldsymbol{x}\in\mathcal{X}} a(\boldsymbol{x}) A_{s_k j}^k(\omega, \boldsymbol{x}),$$

$$\forall i, k, i \neq s_k,\ \sum_{\omega\in\mathcal{W}} p(\omega) \sum_{j: i\in\mathcal{N}_j} \sum_{\boldsymbol{x}\in\mathcal{X}} a(\omega, \boldsymbol{x}) A_{ji}^k(\omega, \boldsymbol{x}) \leq \sum_{\omega\in\mathcal{W}} p(\omega) \sum_{j\in\mathcal{N}_i} \sum_{\boldsymbol{x}\in\mathcal{X}} a(\omega, \boldsymbol{x}) A_{ij}^k(\omega, \boldsymbol{x}).$$

Hence, by Lemma 5 and the argument that the conditional drift under the max-weight policy is smaller than any stationary randomized policy, for the right-hand-side of (22) we have,

$$\mathbb{E} \sum_{k=1}^{K} Q_{s_k}^k(t) \left[\hat{r}_k^* - \sum_{j\in\mathcal{N}_{s_k}} A_{s_k j}^k(\omega(t), \boldsymbol{x}(t)) \right] \leq 0 \quad (24)$$

Therefore, taking expectation of both sides of (22) and combining (23) and (24) yields

$$\mathbb{E}\left[\sum_{k=1}^{K} V \hat{\nabla} f_k(\hat{r}_k^I(t-1))(\hat{r}_k^* - \hat{r}_k(t-1)) \right] + \frac{\mathbb{E}[||\boldsymbol{Q}(t+1)||^2 - ||\boldsymbol{Q}(t)||^2]}{4} \leq \mathbb{E}\left[\alpha[(\hat{r}_k^I(t-1) - \hat{r}_k^*)^2 - (\hat{r}_k^I(t) - \hat{r}_k^*)^2] \right] + C \quad (25)$$

By Lemma 3 and the concavity of $\tilde{f}_k$,

$$\mathbb{E}\left[\sum_{k=1}^{K} V \hat{\nabla} f_k(\hat{r}_k^I(t-1))(\hat{r}_k^* - \hat{r}_k^I(t-1)) \right] \geq \mathbb{E}[V \sum_{k=1}^{K} \tilde{f}(\hat{r}_k^*) - \tilde{f}(\hat{r}_k^I(t-1))].$$

Plugging this in (25) and rearranging terms, we get

$$\mathbb{E}[V \sum_{k=1}^{K} \tilde{f}(\hat{r}_k^*) - \tilde{f}(\hat{r}_k^I(t-1))] \leq \mathbb{E}\left[\alpha[(\hat{r}_k^I(t-1) - \hat{r}_k^*)^2 - (\hat{r}_k^I(t) - \hat{r}_k^*)^2] \right] + \frac{\mathbb{E}[||\boldsymbol{Q}(t)||^2 - ||\boldsymbol{Q}(t+1)||^2]}{4} + C. \quad (26)$$

Ideally, we would want to sum (26) over t and obtain a bound on $\mathbb{E}\left[\sum_{t=1}^{T}\sum_{k=1}^{K} \tilde{f}_k(\hat{r}_k^*) - \tilde{f}_k(\hat{r}_k(t))\right]$. However, the parallel instance paradigm brings intricacy to the argument. It stems from the fact that the job-size variables at different time slot may belong to different instances. To make the reasoning clearer, we will write the invoked instance at t at I_t (i.e., $\hat{r}^k(t) = \hat{r}_k^{I_t}(t)$), which makes the dependence explicit but may compromise readability. First, note that at time t, our job-size decisions are $\{r_k^{I_t}(t)\}$, while the left-hand-side of (26) is $\mathbb{E}[V \sum_{k=1}^{K} \tilde{f}(\hat{r}_k^*) - \tilde{f}(\hat{r}_k^{I_t}(t-1))]$. Since the job-size variables of an instance remain unchanged during the intervals when the instance is not invoked, summing the left-hand-side over time t, the resulting term differs from $\sum_{t=1}^{T} \mathbb{E}[V \sum_{k=1}^{K} \tilde{f}(\hat{r}_k^*) - \tilde{f}(\hat{r}_k^{I_t}(t))] = \sum_{t=1}^{T} \mathbb{E}[V \sum_{k=1}^{K} \tilde{f}(\hat{r}_k^*) - \tilde{f}(\hat{r}_k(t))]$ by at most $CV|\mathcal{I}|$, where $|\mathcal{I}|$ is the total number of instances in the reservoir at the end of the time horizon. Second, summing the right-hand-side of (26) over time, the term $\frac{\mathbb{E}[||\boldsymbol{Q}(t)||^2 - ||\boldsymbol{Q}(t+1)||^2]}{4}$ telescopes, but the term $\alpha[(\hat{r}_k^{I_t}(t-1) - \hat{r}_k^*)^2 - (\hat{r}_k^{I_t}(t) - \hat{r}_k^*)^2]$ only partially telescopes as the invoked instance I_t may be different for different t. More specifically, again due to that the job-size variables of an instance do not change when un-invoked summing the right-hand-side of (26) from $t = 1$ to $T - 1$, we obtain

$$\sum_{I\in\mathcal{I}} \alpha[(\hat{r}_k^I(t_I) - \hat{r}_k^*)^2 - (\hat{r}_k^I(T) - \hat{r}_k^*)^2] + \frac{\mathbb{E}[||\boldsymbol{Q}(1)||^2 - ||\boldsymbol{Q}(T)||^2]}{4} + CT \leq C\alpha|\mathcal{I}| + CT, \quad (27)$$

where t_I is the time that instance I is created, and (27) follows from that $||\boldsymbol{Q}(1)||$ is bounded by a constant while $||\boldsymbol{Q}(T)||^2$ is non-negative.

By the reasoning above, we can see that the key to bound the utility regret is to bound the total number of instance created $|\mathcal{I}|$. By the construction of the parallel-instance paradigm, $|\mathcal{I}|$ is bounded by the maximum delay experienced by the jobs. As the links are FIFO and have offered service rates lower-bounded by a constant, using standard queueing-theoretic arguments, we have $|\mathcal{I}| \leq C \max_t \sum_{n\in V,k} Q_n^k(t)$. Using Theorem 4, it follows that $|\mathcal{I}| \leq \tilde{O}(\sqrt{T})$ with probability at least 1-1/T, which implies that $\mathbb{E}[|\mathcal{I}|] \leq \tilde{O}(\sqrt{T})$ Therefore, summing (26) over time, using (27) and plugging in the value of α, V, we have

$$\sum_{t=1}^{T} \mathbb{E}[V \sum_{k=1}^{K} \tilde{f}(\hat{r}_k^*) - \tilde{f}(\hat{r}_k(t))] \leq C\alpha\mathbb{E}[|\mathcal{I}|] + CT + CV\mathbb{E}[|\mathcal{I}|].$$

Hence, we have

$$\sum_{t=1}^{T} \mathbb{E}[\sum_{k=1}^{K} \tilde{f}(\hat{r}_k^*) - \tilde{f}(\hat{r}_k(t))] \leq \tilde{O}(T^{3/4}),$$

which demonstrates that the utility regret is of order $\tilde{O}(T^{3/4})$ and finishes the proof of Theorem 3.

D. Extension to Noisy Observations

We now extend our results to the case where the utility observations are noisy. Specifically, we consider the case where after a class-k job of size r_k gets delivered to d_k, we obtain and observe $\hat{f}_k(r_k) := f_k(r_k) + \epsilon$, where ϵ is a bounded zero-mean random variable with $|\epsilon| \leq C$. The noisy values (ϵ's) of different jobs are independent and the noise values are independent of the job size decisions. In this case, it is easy to see that Theorem 1 still holds. We now show that the Parallel-Instance GSMW policy can still achieve sublinear regret even when the utility observations are noisy.

The key distinction in the case of noisy observations is that the approximate gradients $\hat{\nabla} f_k(\hat{r}_k^I(t))$ no longer have magnitude upper bounded by a constant, i.e. Lemma 2 does not hold. Instead, we have the following bound.

Lemma 9: In the case of noisy observations, for all k, t, $\hat{\nabla} f_k(\hat{r}_k^I(t)) \leq C/\delta$ with probability 1 for some constant C.

Proof: Recall that the gradient estimate in this case is equal to $\hat{\nabla} f_k(\hat{r}_k^I(t)) = \frac{\hat{f}_k(r_k(t)+\delta) - \hat{f}_k(r_k(t)-\delta)}{2\delta}$. It follows that

$$\hat{\nabla} f_k(\hat{r}_k^I(t)) \leq \frac{(f_k(r_k(t)+\delta)+\epsilon_1) - (f_k(r_k(t)-\delta)+\epsilon_2)}{2\delta}, \quad (28)$$

[4]Here we assume $\mathcal{X}$ to be discrete. The continuous case follows similarly.

where ϵ_1, ϵ_2 are the noises associated with the two utility observations. Since ϵ_1 and ϵ_2 are bounded, the lemma follows. □

Furthermore, Lemma 3 holds in expectation, i.e., for all k, t, $\mathbb{E}[\hat{\nabla} f_k(\hat{r}_k(t))] = \nabla \tilde{f}_k(\hat{r}_k(t))$, which can be easily seen from (28) as the noises are zero-mean. With Lemma 9, we can carry out the same analysis for the P-GSMW policy as in the previous case without observation noise, where instead of bounding the magnitude of $\hat{\nabla} f_k(\hat{r}_k^I(t))$ by a constant, we bound it with C/δ, which may scale with the time horizon T. This result in the following regret guarantee.

Theorem 5: In the case of noisy observations, the Parallel-Instance Gradient Sampling Max-Weight policy π_{P-GSMW} *achieves* $\tilde{O}(T^{7/8})$ *regret by setting* $\alpha = 2K\sqrt{T}/\eta, V = T^{1/8}, \delta = T^{-1/8}$.

The theorem shows that although the performance of the P-GSMW policy degrades with observation noise, it can still achieve sublinear regret.

V. Applications

In this section, we apply our Learning-NUM framework to example applications including database query, job scheduling and video streaming.

A. Database Query

In this example, we consider a setting where there are K users $\{u_1, \ldots, u_K\}$ querying a central database.[5] At each time t, user u_k issues a query of size r_k to the central database, with r_k representing the processing requirement of the query. The issued queries get buffered in the queue of the database and the database can process c unit of requests in a first-come-first-serve order at each time slot. Each use u_k is associated with an underlying utility function f_k that captures the relationship between the processing requirement and utility gained from the query. f_k is Lipschitz continuous and concave, which reflects the diminishing return property of query processing. Over a time horizon of T, the goal is to maximize the total utility of the processed queries.

Applying our framework to the database query example, the network is a simple one with a single state, one source node, one destination node and a link between them (See Figure 3). All the users are mapped to the source node and the database corresponds to the link with the transmission rate of the link at each time slot being equal to the processing capacity c of the database. The network action component of the framework is not needed. The queue at the source node, corresponding to the buffer of the database, buffers the jobs (query requests) of all users. P-GSMW policy adjusts the size of the query according to the gradient estimates and the queue size at the source node, and achieves $\tilde{O}(T^{3/4})$-regret.

B. Job Scheduling

Consider a discrete-time system with with a set of job schedulers (dispatchers) $\{u_1, \ldots, u_K\}$ and a set of parallel servers $\{s_1, \ldots, s_M\}$ that form a bipartite graph. We use S_{u_k} to denote the set of servers that dispatcher u_k is connected to. At each time, a class-k job arrives at the dispatcher u_k and the dispatcher sends the job to one of the servers in S_{u_k} for execution. The job dispatcher also determines the resource (e.g. computation, memory) requirement of each job. Each server s_m can provide $c_m(t)$ amount of resources at time t with $c_m(t)$ being a sequence of i.i.d. discrete random variables. Class-k jobs have underlying utility function f_k. A utility of $f_k(r_k)$ is obtained when a class-k job of resource requirement r_k is completed at a server. We seek a scheduling policy that determines the resource requirement and target server of each job. The goal is to maximize the total utility gained from jobs completed over the time horizon T. This example particularly mirrors applications where the jobs are flexible in terms of resource requirement (e.g., model training for machine learning tasks in cloud computing [5], [19]).

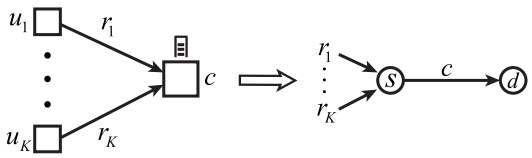


Fig. 3. Correspondence between database query and the Learning-NUM framework.

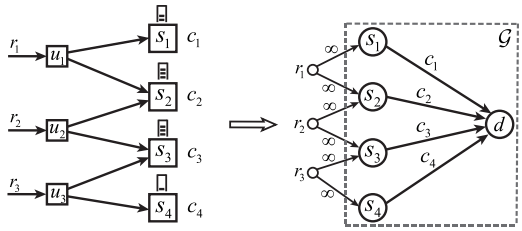


Fig. 4. Correspondence between job scheduling and the Learning-NUM framework.

We apply the Learning-NUM framework to the job scheduling application by creating a source node for each job classes, an intermediate node corresponding to each server and a virtual destination node (See Figure 4). The offered transmission rates of the links between server node and the virtual destination is equal to the time-varying capacity $c_m(t)$ of the servers, and the offered transmission rate between source nodes and intermediate nodes are infinity. The job size $r_k(t)$ corresponds to the resource requirement of class k jobs sent at t. Based on this correspondence, the P-GSMW policy achieves $\tilde{O}(T^{3/4})$-regret. Note that the max-weight scheduling component of the P-GSMW is equivalent to the Join-the-Shortest-Queue policy.

C. Video Streaming

In this example, we consider a network shared by K users streaming video from K corresponding servers. At each time slot, each server sends a chunk of the video file through the network to its corresponding user. The network operator determine the size of the chunks, which correspond to the rates of the video streams. It also controls the routing and scheduling in the network. User k has a utility function f_k that is unknown to the network operator, and obtains utility of value $f_k(r_k)$ after receiving a video chunk of size r_k. Here, we seek a policy that jointly adapts the video rates, i.e., determines the

[5]Note that in this example, we only consider the access to the database and not the problem of routing the queries trough the network.

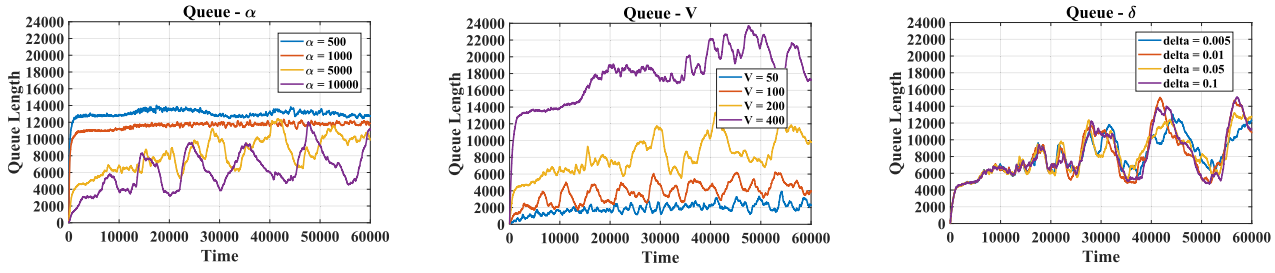


Fig. 5. The queue length under the P-GSMW policy with different parameter values.

size of the video chunks and the routing and scheduling of the network such that the total utility obtained from the delivered video chunks is maximized.

It is natural to map the Learning-NUM framework to the video streaming application. The network shared by the users plays the role of the network $\mathcal{G}$ in the Learning-NUM framework. Each user represents a traffic class. Each traffic class has the user's corresponding video server as the source node with the user node being the destination. The network states capture the possible time-variability in the network links (e.g. in wireless networks). The network action encapsulates the routing and scheduling actions of the network. The feasible action set $\mathcal{X}$ can captures constraints on network operations such as interference constraints and capacity constraints. Applying the P-GSMW policy, we obtain a joint rate-adaptation and network scheduling/routing policy with $\tilde{O}(T^{3/4})$-regret. The network action component here resembles the back-pressure algorithm.

VI. Simulations

In this section, we evaluate the empirical performance of the P-GSMW policy under the Learning-NUM framework. We will also compare P-GSMW policy with GSMW algorithm (in an imaginary no-delay setting) to see the impact of feedback delay on the problem.

We instantiate the Learning-NUM framework on the job scheduling application. The example we construct for the simulation has 50 job schedulers (corresponding to 50 job classes) and 100 parallel servers. The links between job schedulers and servers are randomly generated with each scheduler having expected degree 6 (i.e., connected to 6 servers). The service rate of each server is generated by a uniform random variable with range $[0.5, 1.5]$. We assign an underlying utility function to each class chosen from the four types: $f_k(r) = a_k r$ (linear function), $f_k(r) = a_k\sqrt{r + b_k} - a_k\sqrt{b_k}$ (square root function), $f_k(r) = -a_k r^2 + b_k r$ (quadratic function), $f_k(r) = a_k \log(b_k r + 1)$ (logarithmic function).

Applying the Learning-NUM framework to the example, we first form the corresponding optimization problem $\mathcal{P}$ and obtain that the optimal value $OPT(\mathcal{P})$ is equal to 84.4. We next run the P-GSMW policy and also the GSMW algorithm (**Algorithm 1**). Note that for the GSMW algorithm, we assume an imaginary no-delay setting where the utility values are immediately observable after decisions.

We first investigate the effects of the parameter values (α, V, δ) on the performance of the policy, then compare P-GSMW and GMSW, and finally study the impact of observation noise.

A. Choice of Parameter Values

We vary the values of the parameters (α, V, δ) in the GSMW policy and demonstrate their effects of the policy. The time horizon T is set to 60000. When changing one parameter, the others are held fixed $(\alpha = 5000, V = 200, \delta = 0.005)$. We plot the queue length, defined as the sum of queue length at each server as the time evolves. We will also study the instantaneous utility, defined as $\sum_k f_k(r_k(t))$. The results on queue length are shown in Figures 5. The trajectory of instantaneous utility are much less readible nor informative, and is thus omitted.

1) Parameter α: The parameter α essentially controls the step size of the P-GSMW policy with a larger α indicating a smaller step size. We vary α in $\{500, 1000, 5000, 10000\}$. From the results, the average queue length decreases with the increase of α. The queue length of a larger α tends to have larger oscillation. Recalling the update of job sizes of the P-GSMW policy (Line 5), such behavior can be attributed to that a larger α leads to a smaller "negative feedback" that the queue length has on job sizes.

2) Parameter V: The parameter V adjusts the relative weights of the P-GSMW policy on utility maximization and queue stability, with a larger V indicating that the policy tries to increase the job sizes (and thus the instantaneous utility) more aggressively. We vary V in $\{50, 100, 200, 400\}$. Such behavior is clearly reflected in Figure 5(b) as a larger V leads to a larger steady-state queue size, but the difference is more obscure in the plot of instantaneous utility (figure omitted due to space constraint). We further calculate the time-average instantaneous utility of the P-GSMW policy under different values of V. Corresponding to $V = 50, 100, 200, 400$, the time-average instantaneous utility are $84.8, 85.3, 86.1, 87.1$, respectively. Note that the instantaneous utility can be larger than $OPT(\mathcal{P})$ since the virtual job size variables may not satisfy the capacity constraint of $\mathcal{P}$. The result further supports that a larger V leads to more aggressive increase in the job sizes.

3) Parameter δ: The parameter δ controls the approximation error of our estimate gradients with respect to the true gradients. We vary δ in $\{0.005, 0.01, 0.05, 0.1\}$. Due to that the underlying utility functions in our example do not have large curvature, the value of δ does not have significant effect on the policy.

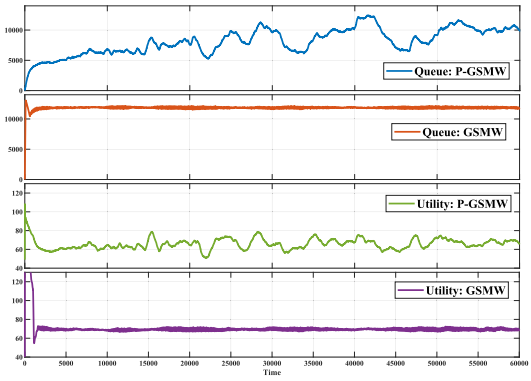


Fig. 6. Queue length and instantaneous utility behavior under P-GSMW and GSMW.

Remark: From the simulations, we can see that the parameters α and V have major influence on the behavior of instantaneous utility and queue length. Although Theorems 3 and 5 provide the correct regimes of α and V that leads to the best regret bound with respect to the scaling of T, it does not give the values of α and V with the best empirical performance. In practice, one can choose the values of α and V by starting from the correct regimes and tuning the values through empirical testing.

B. P-GSMW vs. GSMW

We compare the behaviors of total queue length and instantaneous utility under P-GSMW and GSMW with the same parameter values of $(\alpha = 5000, V = 200, \delta = 0.005)$. Note that P-GSMW is run in our original setting (with queueing-style feedback delay) while GSMW is run in an imaginary no-delay setting. The results on queue length and instantaneous utility (averaging over a sliding window of 1000 time slots) are plotted in Figure 6. It can be seen that, as in terms of job-size variables, P-GSMW switches between different instances of GSMW algorithms, both the queue length trajectory and the instantaneous utility trajectory under P-GSMW exhibits larger oscillation compared to those of GSMW.

Furthermore, varying the time horizon T in $\{10000, 20000, \ldots, 100000\}$ and setting $\alpha = 50\sqrt{T}, V = T^{1/4}, \delta = 1/\sqrt{T}$, we compare how the regret of P-GSMW and GSMW scales with the time horizon. Since it is computationally infeasible to compute the optimal strategy, we use T times $OPT(\mathcal{P})$ as an upper bound of the expected utility achieved by the optimal strategy (see Theorem 1) and bound the regret by $T \cdot OPT(\mathcal{P})$ minus the utility achieved by the policies. We can see from Figure 7 that the regret of GSMW is lower by that of P-GSMW, which suggests that the feedback delay hurt the performance of the policy.

C. Observation Noise

We explore the situation where the utility observations are corrupted with noise and study the robustness of P-GSMW against such noise. We change the noise level from 0 (no noise) to 0.2 (each observation is corrupted with noise that is uniformly distributed in $[-0.2, 0.2]$). Varying the time horizon in $\{10000, 20000, \ldots, 100000\}$ and setting $\alpha = 50\sqrt{T}$,

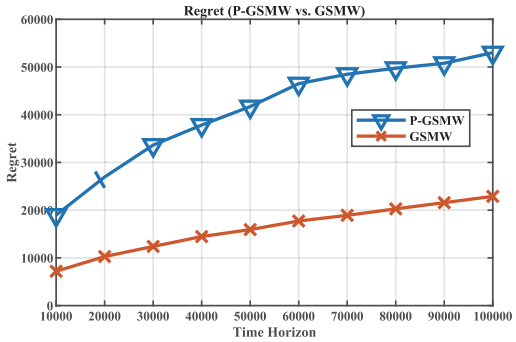


Fig. 7. Regrets of P-GSMW and GSMW.

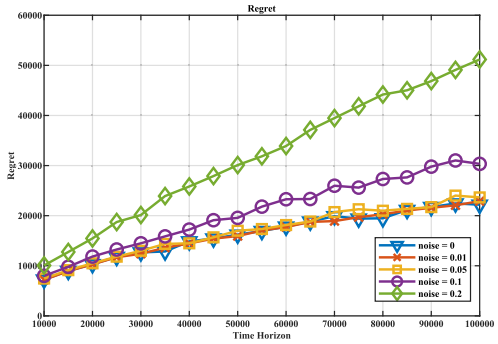


Fig. 8. Regret of P-GSMW under different noise levels.

$V = T^{1/8}, \delta = T^{-1/8}$, we evaluate the scaling of regret under different noise levels. The results are plotted in Figures 8.

From Figures 8, we see that the regrets of P-GSMW sub-linear growth with time horizon even under a noise level of 0.2. Generally, the regret increases with noise level, but the difference is not significant for noise under 0.05.[6]

VII. Conclusion and Future Directions

In this paper, we proposed a new NUM framework, Learning-NUM, where the utility functions are only accessible through zeroth-order feedback and the feedback experiences queueing-style delay. We upper bounded the expected utility achieved by any dynamic policy by the solution to a static optimization problem and designed an online scheduling policy (P-GSMW) that achieves sub-linear regret.

Our scheduling policy achieves a regret of order $\tilde{O}(T^{3/4})$ in the case of noiseless feedback and a regret of order $\tilde{O}(T^{7/8})$ in the case of noisy feedback. As the existing regret lower bound in the case of noiseless feedback in the no-delay case is $\Omega(\sqrt{T})$, our policy does not achieve the lower bound. Hence, an important future direction is to determine whether the queueing-style delay of Learning-NUM fundamental increases the difficulty of the problem, i.e., a lower bound better than $\Omega(\sqrt{T})$ can be established, or that algorithm for Learning-NUM that has regret better than $\tilde{O}(T^{3/4})$ exists. Finally, since the regret of P-GSMW degrades from $\tilde{O}(T^{3/4})$ to $\tilde{O}(T^{7/8})$ in the presence of observation noise, how to minimize the adverse impact of the noise on the policy, and

[6]To put this into perspective, there are 50 job classes and $OPT(\mathcal{P})$ is 84.4. The magnitude of the noise is about $0.05 \times 50/84.4 \simeq 3\%$ of the time-average utility.

are there other methods that are more robust to noise are both questions of future interests.

APPENDIX A
PROOFS OF LEMMAS

A. Proof of Lemma 1

Proof: First, observe that the utility obtained by π_{P-GSMW} over the time horizon T is equal to the total utility of the jobs sent from the sources minus the utility of the jobs that are not delivered at T. Recall that at time t, the two jobs sent from source s_k have size $\hat{r}_k(t)+\delta$ and $\hat{r}_k(t)-\delta$, respectively. Since the utility of a single job is bounded, we have

$$\begin{aligned}&\mathbb{E}[U(\pi_{P-GSMW},T)]\\&\quad\geq \mathbb{E}\left[\sum_{t,k} f_k(\hat{r}_k(t)+\delta)+f_k(\hat{r}_k(t)-\delta)\right]\\&\qquad -C\sum_{i,k}\mathbb{E}[Q_i^k(T)].\end{aligned}$$

By property (2) (Lipschitz continuity) of the underlying utility functions, we have

$$\begin{aligned}&\mathbb{E}\left[\sum_{t,k} f_k(\hat{r}_k(t)+\delta)+f_k(\hat{r}_k(t)-\delta)\right]\\&\qquad\geq 2\mathbb{E}\left[\sum_{t,k} f_k(\hat{r}_k(t))\right]-C\cdot T\delta,\end{aligned}$$

where the last inequality follows from property (2) (Lipschitz continuity) of the underlying utility functions.

By Theorem 1 and that we are now assuming there are two injected jobs of each class at each time slot, $\sup_{\pi^*\in\bar{\Pi}}\mathbb{E}[U(\pi^*,T)]\leq 2T\cdot OPT(\mathcal{P})=2\sum_{t=1}^{T}\sum_{k=1}^{K} f_k(r_k^*)$. Putting the above analysis together, we obtain that

$$\begin{aligned}&R(\pi_{P-GSMW},T)\\&\quad\leq 2\mathbb{E}\left[\sum_{t,k} f_k(r_k^*)-f_k(\hat{r}_k(t))\right]+C\sum_{i,k}\mathbb{E}[Q_i^k(T)]+CT\delta.\end{aligned}$$

□

B. Proof of Lemma 4

Proof: From Line 5 of **Algorithm 2**, since the projection operator is a contraction, we have for each k

$$\begin{aligned}&(\hat{r}_k^I(t)-r_k)^2\\&\quad\leq\left[\hat{r}_k^I(t-1)+\frac{1}{\alpha}(V\cdot\hat{\nabla}f_k(\hat{r}_k(t-1))-Q_{s_k}^k(t))-r_k\right]^2\\&\quad=(\hat{r}_k^I(t-1)-r_k)^2+\frac{V^2(\hat{\nabla}f_k(\hat{r}_k^I(t-1))^2}{\alpha^2}\\&\qquad-\frac{2V\cdot\hat{\nabla}f_k(\hat{r}_k^I(t-1))Q_{s_k}^k(t)}{\alpha^2}\\&\qquad+\frac{1}{\alpha}[\hat{\nabla}f_k(\hat{r}_k^I(t-1))(\hat{r}_k(t-1)-r_k)\\&\qquad-Q_{s_k}^k(t)(\hat{r}_k^I(t-1)-r_k)]+\frac{Q_{s_k}^k(t)^2}{\alpha^2}.\end{aligned}$$

Since $\alpha=2K\sqrt{T}/\eta, V=T^{1/4}$ for all k,τ, we have $\frac{V^2(\hat{\nabla}f_k(\hat{r}_k^I(t-1))^2}{\alpha^2}\leq\frac{C}{\sqrt{T}}$. Plugging these in and rearranging the term, we obtain

$$\begin{aligned}&V\hat{\nabla}f_k(\hat{r}_k^I(t-1))(r_k-\hat{r}_k^I(t-1))+Q_{s_k}^k(t)\hat{r}_k^I(t-1)\\&\quad\leq Q_{s_k}^k(t)r_k+\alpha[(\hat{r}_k^I(t-1)-r_k)^2\\&\qquad-(\hat{r}_k^I(t)-r_k)^2]+\frac{Q_{s_k}^k(t)^2}{\alpha}\\&\qquad-\frac{2V\cdot\hat{\nabla}f_k(\hat{r}_k^I(t-1))Q_{s_k}^k(t)}{\alpha^2}+C.\end{aligned}$$

As $\delta=T^{-1/2}$, the lemma trivially hold for $\hat{r}_k^I(t)=\delta$ for all k. Hence, we only need to consider the case where $\hat{r}_k^I(t)>\delta$, which implies that $\hat{r}_k^I(t)-\hat{r}_k^I(t-1)\leq\frac{1}{\alpha}(V\cdot\hat{\nabla}f_k(\hat{r}_k(t-1))-Q_{s_k}^k(t))$. It follows that

$$\begin{aligned}&V\hat{\nabla}f_k(\hat{r}_k^I(t-1))(r_k-\hat{r}_k^I(t-1))+Q_{s_k}^k(t)\hat{r}_k^I(t)\\&\quad\leq Q_{s_k}^k(t)r_k+\alpha[(\hat{r}_k^I(t-1)-r_k)^2-(\hat{r}_k^I(t)-r_k)^2]\\&\qquad-\frac{V\cdot\hat{\nabla}f_k(\hat{r}_k^I(t-1))Q_{s_k}^k(t)}{\alpha^2}+C.\end{aligned}$$

By Lemma 3, since $\tilde{f}$ is non-decreasing, $\hat{\nabla}f_k(\hat{r}_k^I(t-1))=\nabla\tilde{f}_k(\hat{r}_k^I(t-1))\geq 0$, we have $\frac{2V\cdot\hat{\nabla}f_k(\hat{r}_k^I(t-1))Q_{s_k}^k(t)}{\alpha^2}\geq 0$. Therefore,

$$\begin{aligned}&V\hat{\nabla}f_k(\hat{r}_k^I(t-1))(r_k-\hat{r}_k^I(t-1))+Q_{s_k}^k(t)\hat{r}_k^I(t)\\&\leq Q_{s_k}^k(t)r_k+\alpha[(\hat{r}_k^I(t-1)-r_k)^2-(\hat{r}_k^I(t)-r_k)^2]+C.\end{aligned}$$

The lemma follows by summing over k. □

C. Proof of Lemma 6

Proof: Note that each time slot now correspond to two time slots in our original model of Section II. The lemma follows directly from the dynamics of the queue evolution. For (10), we have

$$\begin{aligned}&Q_{s_k}^k(t+1)^2-Q_{s_k}^k(t)^2\\&\quad\leq\left[Q_{s_k}^k(t)+2\hat{r}_k(t)-2\sum_{j\in\mathcal{N}_{s_k}}\right.\\&\qquad\left.A_{s_kj}^k(\omega(t),\boldsymbol{x}(t))\right]^2-Q_{s_k}^k(t)^2\\&\quad=Q_{s_k}^k(t)^2+4Q_{s_k}^k(t)\left[\sum_{k\in n}\hat{r}_k(t)\right.\\&\qquad\left.-\sum_{j\in\mathcal{N}_{s_k}}A_{s_kj}^k(\omega(t),\hat{\boldsymbol{x}}(t))\right]\\&\qquad+4\left[\hat{r}_k(t)-\sum_{j\in\mathcal{N}_{s_k}}A_{s_kj}^k(\omega(t),\boldsymbol{x}(t))\right]^2-Q_{s_k}^n(t)^2\\&\quad\leq 4Q_{s_k}^k(t)\left[\hat{r}_k(t)-\sum_{j\in\mathcal{N}_{s_k}}A_{s_kj}^k(\omega(t),\hat{\boldsymbol{x}}(t))\right]+C.\end{aligned}$$

Inequality (11) follows similarly. □

D. Proof of Lemma 7

Proof: Continuing from Lemma 4, rearranging terms, we have for any $\{r\}_k$ with $r_k \in [\delta, B-\delta]$,

$$\begin{aligned}
&\sum_{k=1}^{K} Q_{s_k}^k(t)\hat{r}_k(t) \\
&\quad \le \sum_{k=1}^{K} \big[Q_{s_k}^k(t) r_k + \alpha[(\hat{r}_k^I(t-1) - r_k)^2 \\
&\qquad -(\hat{r}_k^I(t) - r_k)^2] + C\big] \\
&\qquad + \sum_{k=1}^{K} V\hat{\nabla} f_k(\hat{r}_k(t-1))(r_k - \hat{r}_k(t-1)) \\
&\quad \le CV + C\alpha + \sum_{k=1}^{K} Q_{s_k}^k(t) r_k \le C\sqrt{T} + \sum_{k=1}^{K} Q_{s_k}^k(t) r_k,
\end{aligned} \tag{29}$$

where inequality (29) follows from that $\alpha = O(\sqrt{T}), V = O(T^{1/4})$. Adding same terms to both sides of (29),

$$\begin{aligned}
&\sum_{i,k} Q_i^k(t) \left[\sum_{j:i\in\mathcal{N}_j} A_{ji}^k(\omega(t), \boldsymbol{x}(t)) - \sum_{j\in\mathcal{N}_i} A_{ij}^k(\omega(t), \boldsymbol{x}(t))\right] \\
&\quad + \sum_{k=1}^{K} Q_{s_k}^k(\tau) \left[\hat{r}_k(t) - \sum_{j\in\mathcal{N}_{s_k}} A_{s_k j}^k(\omega(t), \boldsymbol{x}(t))\right] \\
&\le \sum_{i,k} Q_i^k(t) \left[\sum_{j:i\in\mathcal{N}_j} A_{ji}^k(\omega(t), \boldsymbol{x}(t)) - \sum_{j\in\mathcal{N}_i} A_{ij}^k(\omega(t), \boldsymbol{x}(t))\right] \\
&\quad + \sum_{k=1}^{K} Q_{s_k}^k(t) \left[r_k - \sum_{j\in\mathcal{N}_{s_k}} A_{s_k j}^k(\omega(t), \boldsymbol{x}(t))\right] + C\sqrt{T}.
\end{aligned} \tag{30}$$

We take $\{r\}_k$ to be $r_k = \delta$. By Lemma 5, we have for any $\boldsymbol{x} \in \mathcal{X}$,

$$\begin{aligned}
&\sum_{k=1}^{K} Q_{s_k}^k(t) \left[r_k - \sum_{j\in\mathcal{N}_{s_k}} A_{s_k j}^k(\omega(t), \boldsymbol{x}(t))\right] \\
&\qquad + \sum_{i,k} Q_i^k(t) \left[\sum_{j:i\in\mathcal{N}_j} A_{ji}^k(\omega(t), \boldsymbol{x}(t)) \right. \\
&\qquad \left. - \sum_{j\in\mathcal{N}_i} A_{ij}^k(\omega(t), \boldsymbol{x}(t))\right] \\
&\quad \le \sum_{k=1}^{K} Q_{s_k}^k(t) \left[\delta - \sum_{j\in\mathcal{N}_{s_k}} A_{s_k j}^k(\omega(t), \boldsymbol{x}(t))\right] \\
&\qquad + \sum_{i,k} Q_i^k(t) \left[\sum_{j:i\in\mathcal{N}_j} A_{ji}^k(\omega(t), \boldsymbol{x}(t)) \right. \\
&\qquad \left. - \sum_{j\in\mathcal{N}_i} A_{ij}^k(\omega(t), \boldsymbol{x}(t))\right].
\end{aligned} \tag{31}$$

Combining (14), (30) and (31), we obtain that for any $\boldsymbol{x} \in \mathcal{X}$

$$\begin{aligned}
&||\boldsymbol{Q}(t+1)||^2 - ||\boldsymbol{Q}(t)||^2 \\
&\quad \le 4\sum_{k=1}^{K} Q_{s_k}^k(t) \left[\delta - \sum_{j\in\mathcal{N}_{s_k}} A_{s_k j}^k(\omega(t), \boldsymbol{x}(t))\right] + C\sqrt{T} \\
&\qquad + 4\sum_{i,k} Q_i^k(t) \left[\sum_{j:i\in\mathcal{N}_j} A_{ji}^k(\omega(t), \boldsymbol{x}(t)) \right. \\
&\qquad \left. - \sum_{j\in\mathcal{N}_i} A_{ij}^k(\omega(t), \boldsymbol{x}(t))\right].
\end{aligned}$$

Since $\omega(t)$'s are i.i.d. $\omega(t)$ is independent of $\boldsymbol{Q}(t)$ which only depends on system information before t, we have for each fixed $\boldsymbol{x}$,

$$\begin{aligned}
&\mathbb{E}\left[Q_{s_k}^k(t) \left[\delta - \sum_{j\in\mathcal{N}_{s_k}} A_{s_k j}^k(\omega(t), \boldsymbol{x}) - \eta\right] \mid \boldsymbol{Q}(t)\right] \\
&\quad = Q_{s_k}^k(t) \cdot \mathbb{E}\left[\delta - \sum_{j\in\mathcal{N}_{s_k}} A_{s_k j}^k(\omega(t), \boldsymbol{x})\right] \\
&\quad = Q_{s_k}^k(t) \cdot \sum_{\omega\in\mathcal{W}} p(\omega) \left[\delta - \sum_{j\in\mathcal{N}_{s_k}} A_{s_k j}^k(\omega, \boldsymbol{x})\right].
\end{aligned} \tag{32}$$

Similarly,

$$\begin{aligned}
&\mathbb{E}\left[Q_i^k(t) \left[\sum_{j:i\in\mathcal{N}_j} A_{ji}^k(\omega(t), \boldsymbol{x}(t)) \right. \right. \\
&\qquad \left. \left. - \sum_{j\in\mathcal{N}_i} A_{ij}^k(\omega(t), \boldsymbol{x}(t))\right] \mid \boldsymbol{Q}(t)\right] \\
&\quad \le Q_i^k(t) \cdot \sum_{\omega\in\mathcal{W}} p(\omega) \left[\sum_{j:i\in\mathcal{N}_j} A_{ji}^k(\omega, \boldsymbol{x}) \right. \\
&\qquad \left. - \sum_{j\in\mathcal{N}_i} A_{ij}^k(\omega, \boldsymbol{x})\right]
\end{aligned} \tag{33}$$

Let $\epsilon = \frac{\eta-\delta}{2} > 0$. By the Slater's condition and that $\Lambda(\omega)$ is downward closing, combining (32) and (33), we have

$$\begin{aligned}
&\mathbb{E}[||\boldsymbol{Q}(t+1)||^2 - ||\boldsymbol{Q}(t)||^2 \mid \boldsymbol{Q}(t)] \\
&\quad \le -\epsilon \sum_{i\in V} \sum_{k=1}^{K} Q_i^k(t) + C\sqrt{T}.
\end{aligned} \tag{34}$$

□

E. Statement and Proof of Lemma 10

Lemma 10: Let $\boldsymbol{r}^* = (r_1^*, \ldots, r_k^*)$ be the optimal solution to $\mathcal{P}$. Let $\hat{\boldsymbol{r}}^* = (\hat{r}_1^*, \ldots, \hat{r}_k^*)$ be the optimal solution to $\mathcal{P}$ restricting to each $r_k \in [\delta, B-\delta]$. $\sum_{k=1}^{K} f_k(r_k^*) - f_k(\hat{r}_k^*) \le C\delta$.

Proof: Since $\boldsymbol{\eta} = (\eta, \ldots, \eta)$ is feasible to $\mathcal{P}$ and $\mathcal{P}$ has convex feasibility region, we have $\tilde{\boldsymbol{r}}^* = \frac{\delta}{\eta}\boldsymbol{\eta} + (1 - \frac{\delta}{\eta})\boldsymbol{r}^*$ is feasible to $\mathcal{P}$. Furthermore, observe that for each k, $\tilde{r}_k^* \ge \delta$,

and by Lipschitz-continuity of f_k, $f_k(r_k^*) - f_k(\tilde{r}_k^*) \leq C\delta$. Next, define $\bar{\boldsymbol{r}}^*$ as $\bar{r}_k^* = \tilde{r}_k^*$ if $\tilde{r}_k^* \leq B - \delta$ and $\bar{r}_k^* = B - \delta$ otherwise. Note that for each k, $|\bar{r}_k^* - \tilde{r}_k^*| \leq \delta$ and $\delta \leq \bar{r}_k^* \leq B - \delta$. Also, $\bar{\boldsymbol{r}}^*$ is feasible to $\mathcal{P}$. Hence, by Lipschitz-continuity of f_k, $f_k(\tilde{r}_k^*) - f_k(\bar{r}_k^*) \leq C\delta$. Finally, from the definition of $\hat{\boldsymbol{r}}^*$, we have $\sum_{k=1}^K f_k(\hat{r}_k^*) - f_k(\bar{r}_k^*) \geq 0$. Combine the analysis above and the lemma follows. □

REFERENCES

[1] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, Apr. 1998.

[2] S. H. Low and D. E. Lapsely, "Optimization flow control. I. Basic algorithm and convergence," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 861–874, Dec. 1999.

[3] D. P. Palomar and M. Chiang, "Alternative distributed algorithms for network utility maximization: Framework and applications," *IEEE Trans. Autom. Control*, vol. 52, no. 12, pp. 2254–2269, Dec. 2007.

[4] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," in *Proc. IEEE INFOCOM*, Mar. 2003, pp. 745–755.

[5] S. T. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 702–710.

[6] J. Ghaderi, S. Shakkottai, and R. Srikant, "Scheduling storms and streams in the cloud," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 1, no. 4, pp. 1–28, 2016.

[7] X. Lin and N. B. Shroff, "Utility maximization for communication networks with multipath routing," *IEEE Trans. Autom. Control*, vol. 51, no. 5, pp. 766–781, May 2006.

[8] E. Wei, A. Ozdaglar, and A. Jadbabaie, "A distributed Newton method for network utility maximization," in *Proc. 49th IEEE Conf. Decis. Control (CDC)*, Dec. 2010, pp. 1816–1821.

[9] M. J. Neely, E. Modiano, and C.-P. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 396–409, Apr. 2008.

[10] L. Huang and M. J. Neely, "Utility optimal scheduling in energy-harvesting networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 4, pp. 1117–1130, Aug. 2013.

[11] Q. Liang and E. Modiano, "Network utility maximization in adversarial environments," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 594–602.

[12] M. H. Hajiesmaili, A. Khonsari, A. Sehati, and M. S. Talebi, "Content-aware rate allocation for efficient video streaming via dynamic network utility maximization," *J. Netw. Comput. Appl.*, vol. 35, no. 6, pp. 2016–2027, Nov. 2012.

[13] D. Bethanabhotla, G. Caire, and M. Neely, "Adaptive video streaming for wireless networks with multiple users and helpers," *IEEE Trans. Commun.*, vol. 63, no. 1, pp. 268–285, Jan. 2014.

[14] Y. Zheng, B. Ji, N. Shroff, and P. Sinha, "Forget the deadline: Scheduling interactive applications in data centers," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, Jun. 2015, pp. 293–300.

[15] A. D. Flaxman, A. T. Kalai, and H. B. McMahan, "Online convex optimization in the bandit setting: Gradient descent without a gradient," in *Proc. ACM-SIAM Symp. Discrete Algorithms*, 2005, pp. 385–394.

[16] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," in *Proc. 29th IEEE Conf. Decis. Control*, Dec. 1990, pp. 2130–2132.

[17] H. Yu, M. Neely, and X. Wei, "Online convex optimization with stochastic constraints," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017.

[18] H. Pang, M. J. Carey, and M. Livny, "Multiclass query scheduling in real-time database systems," *IEEE Trans. Knowl. Data Eng.*, vol. 7, no. 4, pp. 533–551, Aug. 1995.

[19] H. Zhang, L. Stafman, A. Or, and M. J. Freedman, "SLAQ: Quality-driven scheduling for distributed machine learning," in *Proc. Symp. Cloud Comput.*, Sep. 2017, pp. 390–404.

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

[21] A. Agarwal, D. P. Foster, D. J. Hsu, S. M. Kakade, and A. Rakhlin, "Stochastic convex optimization with bandit feedback," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 1035–1043.

[22] S. Yang and M. Mohri, "Optimistic bandit convex optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2297–2305.

[23] O. Shamir, "On the complexity of bandit and derivative-free stochastic convex optimization," in *Proc. Conf. Learn. Theory*, 2013, pp. 3–24.

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

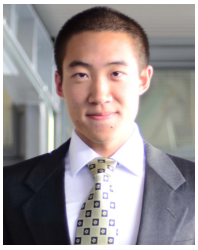
[25] P. Joulani, A. György, and C. Szepesvári, "Online learning under delayed feedback," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1453–1461.

[26] P. Joulani, A. György, and C. Szepesvári, "Delay-tolerant online convex optimization: Unified analysis and adaptive-gradient algorithms," in *Proc. AAAI*, vol. 16, 2016.

[27] C. Pike-Burke, S. Agrawal, C. Szepesvari, and S. Grunewalder, "Bandits with delayed, aggregated anonymous feedback," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4105–4113.

[28] M. Gael, C. Vernade, A. Carpentier, and M. Valko, "Stochastic bandits with arm-dependent delays," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3348–3356.

[29] X. Fu and E. Modiano, "Learning-NUM: Network utility maximization with unknown utility functions and queueing delay," in *Proc. ACM Mobihoc*, 2021, pp. 21–30.



Xinzhe Fu received the B.S. degree (Hons.) in computer science from Shanghai Jiao Tong University in 2017. He is currently pursuing the degree with the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology. His current research focuses on stochastic optimization in queueing networks.

Eytan Modiano (Fellow, IEEE) received the B.S. degree in electrical engineering and computer science from the University of Connecticut, Storrs, in 1986, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, MD, USA, in 1989 and 1992, respectively. Prior to joining a faculty at MIT in 1999, he was a fellow at the Naval Research Laboratory from 1987 to 1992, a Post-Doctoral Fellow at the National Research Council from 1992 to1993, and a member of the Technical Staff at the MIT Lincoln Laboratory from 1993 to 1999. He is a Richard C. Maclaurin Professor with the Department of Aeronautics and Astronautics and the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology. His research is on modeling, analysis, and design of communication networks and protocols.

Dr. Modiano had served for the IEEE Fellows Committee in 2014 and 2015 and an Associate Fellow for AIAA. He received the Infocom Achievement Award in 2020, for contributions to the analysis and design of cross-layer resource allocation algorithms for wireless, optical, and satellite networks. He was a co-recipient of the Infocom 2018 Best Paper Award, the MobiHoc 2018 Best Paper Award, the MobiHoc 2016 Best Paper Award, the Wiopt 2013 Best Paper Award, and the Sigmetrics 2006 Best paper Award. He was the Technical Program Co-Chair for IEEE Wiopt 2006, IEEE Infocom 2007, ACM MobiHoc 2007, and DRCN 2015, and the General Co-Chair of Wiopt 2021. He was the Editor-in-Chief for IEEE/ACM TRANSACTIONS ON NETWORKING from 2017 to 2020, and served as an Associate Editor for IEEE TRANSACTIONS ON INFORMATION THEORY and IEEE/ACM TRANSACTIONS ON NETWORKING.