Performance Evaluation 152 (2021) 102229

Contents lists available at ScienceDirect

Performance Evaluation

journal homepage: www.elsevier.com/locate/peva

Elastic job scheduling with unknown utility functions☆

Xinzhe Fu, Eytan Modiano *

LIDS, Massachusetts Institute of Technology, USA

ARTICLE INFO

Article history:
Available online 2 October 2021

Keywords:
Queueing networks
Stochastic optimization
Job Scheduling

ABSTRACT

We consider a bipartite network consisting of job schedulers and parallel servers. Jobs arrive at the schedulers following stochastic processes with unknown arrival rates, and get routed to the servers, which execute the jobs with unknown service rates. The jobs are *elastic*, as their "size", i.e., the amount of service needed for their completion, is determined by the schedulers. After a job finishes execution, some utility is obtained where the utility value depends on the job's size through some underlying concave utility function. We consider the setting where the utility functions are unknown apriori, while a noisy observation of the utility value of each job is obtained upon its completion. Our goal is to design a policy that makes job-size and routing decisions to maximize the total utility obtained by the end of the time horizon T. We measure the performance of a policy by regret, i.e., the gap between the expected utility obtained under the policy and that under the optimal policy. We first establish an upper bound on the regret of a generic policy, that consists of the cumulative difference in utility between the job-size decisions of the policy and the solution to a static optimization problem, and the total backlog of unfinished jobs at the end of the time horizon. We then propose a policy that simultaneously controls the cumulative utility difference and backlog of unfinished jobs, and achieves an order optimal regret of $\tilde{O}(\sqrt{T})$. Our policy solves the elastic job scheduling problem by extending the Stochastic Convex Bandit Algorithm to handle unknown and stochastic constraints, and making routing decisions based on the Join-the-Shortest-Queue rule. It also presents a principled approach to extending algorithms for zeroth-order convex optimization to the settings with unknown and stochastic constraints.

1. Introduction

Job scheduling is a class of problems that study schedule construction and resource allocation to jobs over a set of machines to optimize for performance objectives such as mean completion time [1,2], makespan [3,4], utility [4,5], and system stability [6,7]. Due to the flexibility and versatility in its modeling and formulation, job scheduling has found a wide range of applications such as supply chain management [1], operating system optimization [2], and cloud computing [6–8].

In many job scheduling applications, the jobs to be scheduled are *elastic*, that is, the arriving jobs do not have a pre-determined size or duration but instead their "sizes" are determined by the system scheduler [9,10], and the utility gained from job completion depends on the "allocated" job size [8,11]. A typical example is training tasks for machine learning

☆ This work was funded by NSF, United States grants CNS-1524317, NSF CNS-1907905 and by Office of Naval Research (ONR), United States grant award N00014-20-1-2119.

* Corresponding author.
E-mail address: MODIANO@MIT.EDU (E. Modiano).

https://doi.org/10.1016/j.peva.2021.102229
0166-5316/

models. The training process of many machine learning models (e.g. deep neural network) involves iterative procedures such as gradient descent [12,13]. The model's performance resulting from the training (i.e., utility of the job) depends on the number of iterations completed (i.e., size of the job) [14]. Thus, it is possible to take advantage of such elasticity to dynamically determine the sizes of incoming jobs to achieve considerable gain in terms of the overall performance, as described in [15,16].

An important element in the scheduling of elastic jobs is the jobs' utility functions, i.e., the underlying relationship between the job size and the corresponding utility. Such utility functions are usually non-decreasing with respect to the job size, and are (approximately) concave, which reflects, for example, the observation that model performance increases with more training time while the marginal gain in performance diminishes with training time [17]. Moreover, the utility functions are often unknown apriori, but function values corresponding to job-size decisions can be observed. Again, using machine learning training as an example, the training curve is typically unknown in advance, but the model performance of a certain training time can be observed after a corresponding training task is completed [18,19]. While monotonicity and concavity have often been utilized to design scheduling algorithms with provable guarantees [8,11], the unknown nature of the utility function has been overlooked by most works in the literature, which assume the utility functions to be known beforehand.

In this paper, we study the problem of elastic job scheduling with unknown utility functions. We consider a discrete-time system of a bipartite network with K job schedulers and a set S of parallel servers. There are K classes of jobs, with jobs of each class arriving at their corresponding job scheduler according to a discrete-time stochastic process with mean rate λ_k. Each class is associated with some concave and monotonically non-decreasing underlying utility function f_k. At every time t, each job scheduler k decides for each incoming job j, the job size x_j and its designated server, and then routes the job to the queue of its designated server. The amount of offered service of server m is a random variable with mean μ_m. After a job j of class k finishes its service at its designated server, we obtain a utility of $f_k(x_j)$ and receive a noisy observation of the function value $f_k(x_j) + \epsilon_j$, where ϵ_j is a zero-mean noise and assumed to be independent for different jobs. Note that the underlying utility functions $\{f_k\}$, and the statistics of the arrival and service processes $\{\lambda_k\}, \{\mu_k\}$ are initially unknown. The goal is to design a policy that makes job-size and routing (choice of designated server) decisions based on observed information, in order to maximize the total utility obtained from jobs completed by the end of the time horizon T. We adopt regret, which is equal to the difference between the utility obtained by the optimal policy and that of our policy, as the performance metric. We start by establishing an upper bound on the regret of a generic policy that consists of a term reflecting the cumulative gap with respect to the optimal solution to a static optimization problem and another term reflecting the amount of unfinished workload at the end of the time horizon. Based on this, we propose a policy that simultaneously controls the two terms and achieves order-optimal regret. In doing so, the main challenges we face are that both the objective function and the constraints of the optimization problem are unknown, and the servers' capacity needs to be effectively utilized to minimize the unfinished workload at the end of the time horizon. Finally, we evaluate the empirical performance of our policy and compare it with a related gradient-based algorithm proposed in [20] that can be adapted to the elastic job scheduling problem.

Specifically, our main contributions are as follows. First, we establish an upper bound on the regret of a generic policy for the elastic job scheduling problem that consists of two terms: one is the cumulative utility difference between the job-size decisions of the policy and the optimal solution to a static optimization problem, and the other is the total queue length (of unfinished jobs) at servers at the end of the time horizon T. The objective function of the static optimization problem is the sum of the utility functions, and constraints are specified by the statistics of the arrival and service processes. Note that in the elastic job scheduling setting, the static optimization problem is not explicitly solvable as the objective function is unknown and the constraints are unknown and stochastic. Second, we propose a policy that achieves an order-optimal regret of $\tilde{O}(\sqrt{T})$[1] by combining ideas from the Stochastic Convex Bandit Algorithm [21] and Join-the-Shortest-Queue routing. Although techniques in related fields such as bandit convex optimization [22] or bandits with knapsack constraints [23,24] can be applied to the elastic job scheduling problem, the order-optimal $\tilde{O}(\sqrt{T})$ regret cannot be achieved by application of previous results in the literature (see Section 7 for a more detailed discussion). Furthermore, our policy forms a principled approach to extending existing zeroth-order optimization algorithms (e.g. the algorithm in [21]) to solving problems with unknown and stochastic constraints, which may be of independent interests. In the literature, unknown and stochastic constraints are typically handled through primal–dual gradient-based methods. When applied to zeroth-order optimization algorithms, such methods rely on using zeroth-order feedback to construct approximate gradients, which leads to sub-optimal regret [20,25]. In contrast, our approach works with algorithms that directly utilizes zeroth-order feedback [21,26] and enjoys order-optimal regret.

We conclude the introduction by giving a road map for the rest of the paper. We present the model and formal definitions of the elastic job scheduling problem in Section 2. In Section 3, we prove a general upper bound on the regret. In Section 4, we introduce the algorithm from [21] and extract the key results therein that will be useful in the design and analysis of our policy. The policy we propose for the elastic job scheduling problem is presented in Section 5. We evaluate the empirical performance of our policy in Section 6. A discussion of related works and their relation to the elastic job scheduling problem is presented in Section 7. Finally, we conclude the paper with some future directions in Section 8.

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

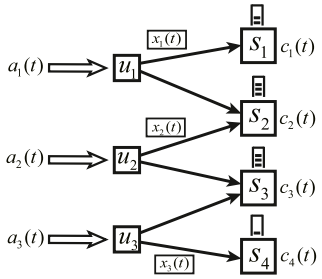

Fig. 1. Illustration of the system model of the elastic job scheduling problem.

2. Model and problem formulation

2.1. System model

Consider a discrete-time system with a set of job schedulers and a set of parallel servers that form a bipartite network (see Fig. 1). We use $U = \{u_1, \ldots, u_K\}$ to denote the set of schedulers and $S = \{s_1, \ldots, s_M\}$ to denote the set of servers. Each scheduler u_k is connected to a subset $S_{u_k} \subseteq S$ of servers. Each server has a buffer that stores the jobs to be processed. There are K classes of elastic jobs in the system, where jobs of class k arrive at scheduler u_k and are sent to a server in S_{u_k} for execution. At each time slot t, a set $A_k(t)$ of class k jobs with $|A_k(t)| = a_k(t)$ arrive at scheduler u_k. For each job j, its corresponding scheduler determines its size $x_j \in [0, B]$.[2] which is the workload it will add to the server and can be interpreted as its resource requirement. The scheduler then sends job j to the buffer of a server $s_j \in S_{u_k}$ for execution, which we will refer to as j's designated server. Server s_m's service rate at time t is denoted by $c_m(t)$. Each server executes the jobs in a non-preemptive fashion. We assume that for each k, $a_k(t)'s$ form a sequence of i.i.d. bounded positive integer random variables, and for each m, $c_m(t)$'s is a sequence of i.i.d. bounded non-negative random variables. We assume, $\mathbb{E}[a_k(t)] = \lambda_k$, $\mathbb{E}[c_m(t)] = \mu_m$ and $1 \leq a_k(t), \lambda_k, c_m(t), \mu_m \leq C$.[3] We will refer to the jobs' arrival rates λ_k's and the servers' service rates c_m's, as *network statistics* In this work, we consider the setting where the network statistics are unknown, but the realizations of arrivals and service are observable (after they occur).

For each server s_m, we use $Q_m(t)$ to denote the workload queued in the buffer of s_m at time t. From the description of the model, the amount of work arriving to Q_m at time t is equal to the total size of the jobs that are sent to server s_m from the schedulers. Hence, we can write the evolution of the workload process as follows where $[\cdot]^+ = \max\{\cdot, 0\}$:

$$Q_m(t+1) = [Q_m(t) + \sum_{k=1}^{K} \sum_{j \in A_k(t)} \mathbb{1}_{\{s_j = s_m\}} \cdot x_j - c_m(t)]^+. \tag{1}$$

Each class k is associated with some underlying utility function f_k that characterizes the relationship between the size and the utility value obtained from jobs of class k. The underlying utility functions are unknown, but we can receive noisy zeroth-order feedback on the utility functions. Specifically, after the server finishes executing a job of size x of class k, we observe $f_k(x) + \epsilon$ and obtain a utility of $f_k(x)$, where ϵ is a zero-mean bounded random noise with $|\epsilon| \leq C$.[4] The noise values of different jobs are independent. It is important to note that the utility of a job is received and observed at the time of its completion rather than the time it is dispatched from the scheduler. We assume that for each job class k, its underlying utility function f_k has the following properties:

1. f_k is monotonically non-decreasing and concave.
2. f_k is bounded on the domain $[0, B]$, i.e., $\forall x_j \in [0, B], f_k(x_j) \leq C$. $f_k(0) = 0$.
3. f_k is L-Lipschitz continuous, i.e., $\forall 0 \leq x_1, x_2 \leq B$, $|f_k(x_2) - f_k(x_1)| \leq L \cdot |x_2 - x_1|$.

2.2. Problem formulation

Under the above system model, we study a finite-horizon elastic job scheduling problem. Given a time horizon T, we seek a scheduling policy that determines the size of arriving jobs and their designated servers such that the total utility obtained from the jobs that are completed in T time slots is maximized. Our scheduling policy needs to be admissible

[2] $B \in \mathbb{R}^+$ is an upper bound on the jobs' size. The job size takes value in $\mathbb{R}^+$ and needs not be an integer.

[3] We restrict $a_k(t), c_m(t)$ to be positive so that at each time we have at least one arrival from each class and the realized service rates are positive, which simplifies the description of the policy we propose. Our results can be shown without this restriction.

[4] We use C to bound various bounded quantities without loss of generality, as C can be taken to be the maximum of realizations of arrivals and service, and observations of utility values.

such that at each time t, the decisions it makes are only based on information observed prior to t, i.e., $a_k(\tau)$'s for $\tau \leq t$, $c_m(\tau)$'s for $\tau < t$ and the utility observations obtained before t. Let Π be the collection of admissible policies. Policies in Π do not have access to the underlying utility functions f_k's or the network statistics in advance, but can only learn them through observations acquired from job executions. Let Π^* be the set of all policies, including the ones that know the underlying utility functions and network statistics. For a policy π, let $U(\pi, T)$ be the total utility obtained under policy π, which is defined as the sum of utility obtained from jobs that have been completed by the end of the time horizon T. Note that $U(\pi, T)$ is a random variable, the randomness of which comes from job arrivals, service rates, noisy utility observations and the (possible) inherent randomness in the scheduling policy π. Instead of directly using $U(\pi, T)$, we adopt the notion of regret as the measure of the quality of scheduling policies.

Definition 1 (*Regret*)**.** The regret of scheduling policy π is defined as

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

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. The goal of the elastic job scheduling problem is to design an admissible scheduling policy with low regret.

Remark. (i). A crude criterion of low regret is that $R(\pi, T) = o(T)$, i.e., the regret of π grows sub-linearly with T. If policy π satisfies this criterion, then the time average utility achieved by π is asymptotically optimal as $T \to \infty$. Applying the same argument as [27], it can be shown that there exists instance of the elastic job scheduling problem where no admissible policy can have regret lower than $\Theta(\sqrt{T})$. The regret of the policy we will propose has regret of order $\tilde{O}(\sqrt{T})$, which implies that it achieves order-optimal regret (ignoring logarithmic factors). (ii). Although the performance measure does not explicitly depend on the unfinished workload in the servers at the end of the horizon T, such unfinished workload is implicitly accounted for, since the utility $U(\pi, T)$ does not include the unfinished jobs in the queues at time T. This means that, to achieve low regret, we cannot blindly increase the job sizes without maintaining the stability of the system.

3. General upper bound on the regret

The regret of a policy π involves the expected utility of the optimal policy in Π^* and the expected utility of π. Due to the dynamic nature of the elastic job scheduling problem, directly computing the regret is challenging. In this section, we prove an upper bound on the regret for policies that satisfy a certain condition (the policy we propose satisfies the condition), which facilitates the analysis and design of our policy. The upper bound is constructed by establishing on upper bound on the expected utility of the optimal policy, and a lower bound on the utility of a general policy.

3.1. Upper bound on the optimal utility

Typically, the policy that achieves $\sup_{\pi^* \in \Pi^*} \mathbb{E}[U(\pi^*, T)]$ 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 Π^* to the optimal value of a (essentially static) convex optimization problem. To begin with, we define a notion of capacity region of the network, denoted by Λ. Intuitively, Λ can be interpreted as the set of job-size vector that can be supported by the network in steady state (i.e., under infinite time horizon). It is formally defined as:

$$\Lambda := \left\{ (x_1, \ldots, x_K) \;\middle|\; \begin{array}{l} \exists \{\alpha\}_{km}, \text{ s.t.} \\ \sum_k \lambda_k \alpha_{km} x_k \leq \mu_m \quad \forall s_m, \\ \alpha_{km} = 0, \quad \forall s_m \notin S_{u_k} \\ \sum_m \alpha_{km} = 1, \forall k \\ \alpha_{km} \geq 0, \quad \forall k, m \\ 0 \leq x_k \leq B, \quad \forall k \end{array} \right\},$$

where the variables $\{\alpha_{km}\}$ can be considered to be the routing variables that determine how the jobs of each class are distributed among the servers. Λ conforms with the capacity region in the network control literature and it is easy to see that Λ is a convex set. Recall that B is the maximum possible size of a job. Consider the following optimization problem $\mathcal{P}$:

$$\mathcal{P}: \max_{\{x_k\}} \sum_{k=1}^{K} \lambda_k f_k(x_k) \tag{2}$$

$$\textbf{s.t.} \quad (x_1, \ldots, x_K) \in \Lambda, \tag{3}$$

$$x_k \in [0, B], \quad \forall k. \tag{4}$$

Intuitively, the optimization problem characterizes the job scheduling problem with full information in steady state. The decision variables $\{x_k\}$ can be interpreted as the steady-state size of jobs of class k. As the objective function of $\mathcal{P}$ is concave while the feasibility region is a convex set, it follows that $\mathcal{P}$ is a convex optimization problem. However, note that although $\mathcal{P}$ is convex when we view $\{x_k\}$ as optimization variables, it loses its convexity if we view $\{x_k\}$ and $\{\alpha_{km}\}$ jointly as optimization variables, since the capacity region Λ involves constraints $\sum_k \lambda_k \alpha_{km} x_k \leq \mu_m$, which makes it non-convex when viewed as a set over $\{x_k\}$ and $\{\alpha_{km}\}$.

We proceed to show that the expected utility of any policy in Π^* is upper-bounded by the optimal value of the $\mathcal{P}$ times the time horizon T.

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

Proof Sketch. For any given policy, we first take weighted averages of the sizes of jobs of each class under the policy over the realizations of the arrival processes. We will then show that the averages satisfy the constraints of $\mathcal{P}$, and by the concavity of the underlying utility functions, the corresponding value of the objective function is no less than the expected utility of the policy. Due to space constraints, we defer the complete proof to Appendix A.1.

It is worth pointing out that Theorem 1 does not imply that the optimal policy is a static one that assigns the execution time of jobs according to the solution to the optimization problem $\mathcal{P}$. Such policy would not achieve an expected utility of $T \cdot OPT$. The reason is that due to the stochasticity of the system, the expected number of class k jobs completed before T is not equal to $\lambda_k T$.

3.2. Lower bound on the general utility

We now give a lower bound on the expected utility achieved by policies that make job-size decisions independently of the number of arrivals at the same time slot. As the policy we will propose satisfies this condition, combined with Theorem 1, it will lead to an upper bound on the regret of our policy. More formally, let π be an arbitrary policy that, for each t, decides on $x_k(t)$ independently of $a_k(t)$ for all k.[5] On a generic sample path ω, let $x_k(t, \omega)$ be size of class k jobs decided by π at time t. Let $Q_m(T, \omega)$ be the workload at server s_m at the end of time horizon T under π. We will write $\mathbb{E}_\omega[f_k(x_k(t, \omega))]$ as $\mathbb{E}[f_k(x_k(t))]$ and $\mathbb{E}_\omega[Q_m(T, \omega)]$ as $\mathbb{E}[Q_m(T)]$. We have the following.

Proposition 1. $\mathbb{E}[U(\pi, T)] \geq \sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k \mathbb{E}[f(x_k(t))] - L \sum_{m=1}^{M} \mathbb{E}[Q_m(T)]$.

Proof. If all the jobs were completed, then under π, due to the independence of arrivals and job size decisions, the expected utility obtained would be

$$\sum_{t=1}^{T} \sum_{k=1}^{K} \mathbb{E}_\omega[a_k(t, \omega) f(x_k(t, \omega))] = \sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k \mathbb{E}_\omega[f(x_k(t, \omega))] = \sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k \mathbb{E}[f(x_k(t))]$$

The utility actually achieved by π is equal to the difference between the aforementioned quantity and the utility of incomplete jobs in the queues of the servers at the end of the time horizon. Since the each f_k is L-Lipschitz-continuous, $f_k(x) \leq Lx$. Thus, the expected total utility of incomplete jobs at the end of time horizon is upper bounded by $L \sum_{m=1}^{M} \mathbb{E}[Q_m(T)]$. □

Let $\boldsymbol{x}^* = (x_1^*, \ldots, x_K^*)$ be the optimal solution to $\mathcal{P}$. It is straightforward to show the following corollary from Theorem 1 and Proposition 1.

Corollary 1. *For any policy π that decides on $x_k(t)$ independently of $a_k(t)$, $R(\pi, T) \leq \sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k \cdot \mathbb{E}[f_k(x_k^*) - f_k(x_k(t))] + L \sum_{m=1}^{M} \mathbb{E}[Q_m(T)]$.*

Corollary 1 indicates that a policy can achieve low regret if it can (i). closely approximate the optimal solution to $\mathcal{P}$, and at the same time, (ii). bound the queue lengths at the servers. The core challenge of achieving (i) lies in that in our job scheduling problem, we do not know the network statistics $\{\lambda_k\}$, $\{c_m\}$. Indeed, if the network statistics were known in advance, then the feasibility region of $\mathcal{P}$ would also be known in advance. Solving $\mathcal{P}$ would become a stochastic zeroth-order convex optimization problem, where we can apply the algorithm in [21] and achieve an order-optimal regret of $\tilde{O}(\sqrt{T})$. However, in the elastic job scheduling problem, the network statistics, and thus the feasibility region, are unknown and stochastic. In other words, our problem has stochastic constraints, which makes the algorithm in [21] not applicable. Furthermore, existing algorithms in stochastic zeroth-order optimization do not involve any scheduling component that controls the queue lengths of the system (i.e., achieving (ii)). In the following, we will propose a scheduling policy that adapts the techniques of [21], employs suitable network scheduling, and still achieves an order-optimal regret of $\tilde{O}(\sqrt{T})$ for the elastic job scheduling problem.

[5] Here we assume that the policy sets the size of all the class-k jobs at time t to be $x_k(t)$. Otherwise, we can take $x_k(t)$ as the average size of the class-k jobs at time t and the results still hold.

4. Preliminaries

In this section, we review the Stochastic Convex Bandit Algorithm (SCBA) from [21] and extract the key results therein. Although the algorithm cannot handle the stochastic and unknown constraints, it can be used in conjunction with the technique developed in this paper to form a policy that achieves order-optimal regret for the elastic job scheduling problem. The Stochastic Convex Bandit Algorithm is a generalization of the ellipsoid algorithm [28] to stochastic zeroth-order convex optimization problems. Formally, let $\mathcal{X} \subseteq \mathbb{R}^d$ be a compact convex set, and $f(\boldsymbol{x}) : \mathcal{X} \mapsto \mathbb{R}$ be a convex and L-Lipschitz continuous function on $\mathcal{X}$. In [21], the Stochastic Convex Bandit Algorithm aims at solving the optimization problem

$$\min_{\boldsymbol{x}} \quad f(\boldsymbol{x}) \tag{5}$$

$$\textbf{s.t.} \quad \boldsymbol{x} \in \mathcal{X}. \tag{6}$$

with access to a noisy zeroth-order oracle on f. Specifically, if we query the oracle at $\boldsymbol{x}$, we observe $\hat{f}(\boldsymbol{x}) := f(\boldsymbol{x}) + \epsilon$ where ϵ is a zero-mean σ-sub-Gaussian random variable and the noise values of different queries are independent. Let $\boldsymbol{x}^*$ be the optimal solution. SCBA outputs a sequence of queries $\boldsymbol{x}(1), \ldots, \boldsymbol{x}(T)$ (with T being the time horizon of SCBA, i.e., the total number of queries), where each query $\boldsymbol{x}(t)$ is computed based on observations obtained from queries before t, such that $\sum_{t=1}^{T} f(\boldsymbol{x}(t)) - f(\boldsymbol{x}^*) = \tilde{O}(\sqrt{T})$ with high probability. We use the one-dimensional special case ($d = 1$) to demonstrate the workflow of the algorithm. As $d = 1$, the feasibility region $\mathcal{X}$ is essentially an interval $[l_0, r_0]$. SCBA proceeds in epochs. It maintains a target region $[l, r]$ that contains the optimal solution $\boldsymbol{x}^*$, with high probability. At every epoch, it choose three points that uniformly span the target region. It then repeatedly queries the points and construct a confidence interval around the underlying function value of each point. Next, it shrinks the target region by eliminating an area that is unlikely to contain $\boldsymbol{x}^*$ based on the confidence intervals.[6] Specifically, at a generic epoch τ with target region $[l_\tau, r_\tau]$ the algorithm executes the following steps (where two intervals $[l_1, u_1]$ and $[l_2, u_2]$ are γ-separated if $l_1 > u_2 + \gamma$):

1. Let $w_\tau = r_\tau - l_\tau$. Set $x_l := l_\tau + \frac{w_\tau}{4}$, $x_c := l_\tau + \frac{w_\tau}{2}$, $x_r := l_\tau + \frac{3w_\tau}{4}$.
2. For $i = 1, \ldots$; $\gamma_i = 1/2^i$:
   (a) Query x_l, x_c, x_r for $\lceil \sigma \log T^2/\gamma_i^2 \rceil$ times. Let $\bar{f}(x_l), \bar{f}(x_c), \bar{f}(x_r)$ be the empirical mean of observations of $f(x_l), f(x_c), f(x_r)$, respectively.
   (b) Set confidence intervals $[LB_{x_l}, UB_{x_l}] := [\bar{f}(x_l) - \gamma_i/2, \bar{f}(x_l) + \gamma_i/2]$, $[LB_{x_c}, UB_{x_c}] := [\bar{f}(x_c) - \gamma_i/2, \bar{f}(x_c) + \gamma_i/2]$, $[LB_{x_r}, UB_{x_r}] := [\bar{f}(x_r) - \gamma_i/2, \bar{f}(x_r) + \gamma_i/2]$.
   (c) If $[LB_{x_l}, UB_{x_l}]$ is γ_i-separated with $[LB_{x_c}, UB_{x_c}]$ or $[LB_{x_r}, UB_{x_r}]$, eliminate $[l_\tau, x_l]$ from the target region (by setting l_τ to x_l) and proceed to the next epoch.
   (d) If $[LB_{x_r}, UB_{x_r}]$ is γ_i-separated with $[LB_{x_c}, UB_{x_c}]$ or $[LB_{x_l}, UB_{x_l}]$, eliminate $[x_r, r_\tau]$ from the target region (by setting r_τ to x_r) and proceed to the next epoch.
   (e) Otherwise, increment i and repeat step (2).

Note that in step 2(a), each point x_l, x_c, x_r is queried multiple times but each time counts as one query, i.e., the sequence of queries output by the algorithm will have multiple copies of x_l, x_c, x_r. For example, we arrive at step 2(a) for some γ_i and x_l, x_c, x_r at t. Then, SCBA will output $x(t+1) = \cdots = x(t + \lceil \sigma \log T^2/\gamma_i^2 \rceil)$ as x_l, $x(t + \lceil \sigma \log T^2/\gamma_i^2 \rceil + 1) = \cdots = x(t + 2\lceil \sigma \log T^2/\gamma_i^2 \rceil)$ as x_c, and $x(t + 2\lceil \sigma \log T^2/\gamma_i^2 \rceil + 1) = \cdots = x(t + 3\lceil \sigma \log T^2/\gamma_i^2 \rceil)$ as x_r. In high-dimension cases, the algorithm follows the same road map but uses more sophisticated methods for the selection of query points and the shrinking of target regions. Proposition 2 summarizes the performance guarantee of SCBA.

Proposition 2 (*Theorem 1 of [21]*). *The Stochastic Convex Bandit Algorithm (SCBA) outputs a sequence of queries $\boldsymbol{x}(t)$, $t = 1, \ldots, T$ such that $\sum_{t=1}^{T} f(\boldsymbol{x}(t)) - f(\boldsymbol{x}^*) = O(L\sigma\sqrt{T}\log T)$ with probability at least $1 - 1/T$. Moreover, it follows that $\sum_{t=1}^{T} \mathbb{E}[f(\boldsymbol{x}(t))] - f(\boldsymbol{x}^*) = O(L\sigma\sqrt{T}\log T)$.*

The original setting in [21] assumes that the oracle gives unbiased observations of the objective function. However, in our solution framework for the elastic job scheduling problem that will be described in the next section, we do not have unbiased observations of the objective function. Thus, we need to extend SCBA to more general settings, where observations may be biased.

We can see from the high-level description above that the query point selection and target region shrinking are agnostic to how the confidence intervals are obtained. The algorithm would have a similar performance guarantee as long as the confidence intervals used (Step 2(b)) have a certain guarantee of accuracy. Therefore, if we have a procedure that for any $\boldsymbol{x}$, takes in a number of (possibly biased) observations of $f(\boldsymbol{x})$ from the oracle and outputs a confidence interval $[LB_{\boldsymbol{x}}, UB_{\boldsymbol{x}}]$ that is sufficiently accurate, then SCBA would still work in conjunction with the procedure, albeit without the availability of unbiased observations. The aforementioned accuracy requirement of the procedure is formalized in the following definition. Informally speaking, a procedure is qualified if based on a large enough number observations, it can

[6] The shrinkage of target region is a step that explicitly relies on the knowledge of the feasibility region in advance.

construct a confidence interval with width bounded by some quantity that decreases with the number of observations, and the confidence interval contains the true value with high probability.

Definition 2. For a given function f with domain $\mathcal{X}$, an observation oracle and $\sigma > 0$, a procedure is **qualified** if for any $\boldsymbol{x} \in \mathcal{X}$, $\gamma > 0, 0 < \delta \leq 1/2$, given $\lceil \frac{\sigma \log(1/\delta)}{\gamma^2} \rceil$ observations of $f(\boldsymbol{x})$ from the oracle, the procedure outputs a confidence interval $[LB_{\boldsymbol{x}}, UB_{\boldsymbol{x}}]$ such that $UB_{\boldsymbol{x}} - LB_{\boldsymbol{x}} \leq \gamma$ and $\mathbb{P}\{f(\boldsymbol{x}) \in [LB_{\boldsymbol{x}}, UB_{\boldsymbol{x}}]\} \geq 1 - \delta$.

Definition 2 can be seen as a generalization of the availability of unbiased observations, i.e., the original setting of [21]. Since for the setting of [21], if each observation is unbiased with noise being an independent zero-mean σ-sub-Gaussian random variable, then for any $\gamma > 0, 0 < \delta \leq 1/2$, given $\lceil \frac{\sigma \log(1/\delta)}{\gamma^2} \rceil$ such observations, it follows from Hoeffding's inequality that the confidence intervals used in Step 2(b) satisfy the condition $\mathbb{P}\{f(\boldsymbol{x} \in [LB_{\boldsymbol{x}}, UB_{\boldsymbol{x}}])\} \geq 1 - \delta$. In particular, in Step 2(b), we take $\delta = 1/T^2$ and have that each confidence interval constructed by the algorithm contains the true value with probability at least $1 - 1/T^2$, which, combined with the union bound, implies that all the confidence intervals contain the true value with probability at least $1 - 1/T$. Moreover, inspecting the analysis of [21], we can see that SCBA would have the same performance guarantee as in Proposition 2 as long as some qualified procedure is available. As a concrete example, in the one dimensional special case, a qualified procedure can replace Step 2(b) without compromising the performance of the algorithm. We summarize this generalized performance guarantee of SCBA in Proposition 3.

Proposition 3. *The Stochastic Convex Bandit Algorithm (SCBA), in conjunction with a qualified procedure, outputs a sequence of queries $\boldsymbol{x}(t), t = 1, \ldots, T$ such that $\sum_{t=1}^{T} f(\boldsymbol{x}(t)) - f(\boldsymbol{x}^*) = O(L\sigma\sqrt{T}\log T)$ with probability at least $1 - 1/T$.*

Proof. The proposition follows from a similar analysis as in [21]. We defer more details to Appendix A.2. □

Note that if we are maximizing a concave function f over $\mathcal{X}$ (cf. (5), (6)), then a similar guarantee holds with $\sum_{t=1}^{T} f(\boldsymbol{x}(t)) - f(\boldsymbol{x}^*)$ replaced by $\sum_{t=1}^{T} f(\boldsymbol{x}^*) - f(\boldsymbol{x}(t))$.

5. The scheduling policy

In this section, we propose our scheduling policy for the elastic job scheduling problem — the Confidence Interval-based Join-the-Shortest-Queue (CI-JSQ) policy. The CI-JSQ policy has two components: an optimization component that makes job-size decisions at each time slot, and a routing component that selects a designated server for each job following the Join-the-Shortest-Queue rule. Assuming that the utility observations are obtained immediately after the job-size decisions, i.e., there is no feedback delay, we introduce the optimization component in Section 5.1 and the routing component in Section 5.2. We next analyze the performance of the policy in Section 5.3. Finally, we remove the no-feedback delay assumption and extend the policy to the original setting of the elastic job scheduling problem where the utility observations of the jobs are obtained after the jobs' completion in Section 5.4.

5.1. The optimization component

In Corollary 1, we show that a policy can achieve low regret if it can make job-size decisions close to the optimal solution to $\mathcal{P}$ and control the queue length of the system. Therefore, the optimization component of our policy should be able to closely track the optimal solution to $\mathcal{P}$. Despite the similarity of the observation model of $\mathcal{P}$ and that assumed by the algorithm of [21], we cannot directly apply the algorithm due to the unknown and stochastic constraints of $\mathcal{P}$. In this section, we develop techniques that make SCBA applicable, so that the query point output by SCBA can be used as job-size decisions. Our techniques, in conjunction with SCBA, form the optimization component of our policy for the elastic job scheduling problem.

The overall framework of our approach is that, first, we move the stochastic constraint $\boldsymbol{x} \in \Lambda$ of $\mathcal{P}$ in to a penalty term in the objective function and form a (essentially equivalent) transformed optimization problem $\tilde{\mathcal{P}}$. The transformed optimization problem no longer has any stochastic constraints, but unlike utility values, unbiased observations of the penalty term are not available. We thus develop a confidence interval construction procedure and show that under the observation model of our problem, the procedure is "qualified" according to Definition 2. Therefore, we can use our confidence interval construction procedure in conjunction with SCBA to compute job-size decisions for the elastic job scheduling problem.

5.1.1. The transformed optimization problem

For ease of notation, we use the vector $\boldsymbol{x}$ to denote $(x_1, \ldots, x_K)$, $\boldsymbol{\lambda}$ to denote the arrival rates $\{\lambda_k\}$ and $\boldsymbol{\mu}$ to denote the service rates $\{\mu_m\}$. We define the function $\Delta(\boldsymbol{x}, \Lambda)$ as the l_1-distance of $\boldsymbol{x}$ to the capacity region Λ, i.e., $\Delta(\boldsymbol{x}, \Lambda) := \arg\min_{\boldsymbol{x}' \in \Lambda} \sum_{k=1}^{K} |x_k - x'_k|$. $\Delta(\boldsymbol{x}, \Lambda)$ corresponds to a notion of *constraint violation*, i.e., how far away a job-size vector is from the capacity region of the network. We will use $\|\cdot\|_1$ to denote the l_1 norm and $\|\cdot\|$ is reserved for the Euclidean

(l_2) norm. Note that the Lipschitz continuity in this paper is defined with respect to the Euclidean norm. The transformed problem $\tilde{\mathcal{P}}$ is presented at the top of the page (with $\mathcal{P}$ shown on the left for reference).

$$\mathcal{P}: \quad \max_{\boldsymbol{x}} \sum_{k=1}^{K} \lambda_k f_k(x_k) \quad \text{s.t.} \quad \boldsymbol{x} \in \Lambda, \; x_k \in [0, B], \quad \forall k.$$

$$\tilde{\mathcal{P}}: \quad \max_{\boldsymbol{x}} \quad F(\boldsymbol{x}) := \sum_{k=1}^{K} \lambda_k f_k(x_k) - C(L+1)\Delta(\boldsymbol{x}, \Lambda) \quad \text{s.t.} \quad x_k \in [0, B], \quad \forall k.$$

The optimization problem $\tilde{\mathcal{P}}$ is constructed by replacing the constraint $\boldsymbol{x} \in \Lambda$ of $\mathcal{P}$ by a penalty term in the objective function that penalizes the constraint violation of $\boldsymbol{x}$ through $\Delta(\boldsymbol{x}, \Lambda)$. The optimization variables of $\tilde{\mathcal{P}}$ only consist of job-size variables $\{x_k\}$ but not routing variables $\{\alpha_{km}\}$. As $\tilde{\mathcal{P}}$ does not involve stochastic constraints, it may be more amenable to SCBA. To formally justify this, in the following lemmas, we show that $\tilde{\mathcal{P}}$ satisfies the requirement of SCBA in form, i.e., it is a convex optimization problem with Lipschitz objective function. Furthermore, $\mathcal{P}$ and $\tilde{\mathcal{P}}$ have the same set of optimal solutions. The proof of the lemmas is deferred to Appendix B.

Lemma 1. *$\Delta(\boldsymbol{x}, \Lambda)$ is a convex function of $\boldsymbol{x}$, and thus $F(\boldsymbol{x})$ is a concave function of $\boldsymbol{x}$.*

Lemma 2. *$\Delta(\boldsymbol{x}, \Lambda)$ is $\sqrt{K}$-Lipschitz continuous, thus $F(\boldsymbol{x})$ is $KL + C(L+1)\sqrt{K}$-Lipschitz continuous.*

Lemma 3. *$\tilde{\mathcal{P}}$ is a convex optimization problem with the same set of optimal solutions as $\mathcal{P}$.*

5.1.2. Construction of confidence intervals

The transformed optimization problem is more amenable to the algorithm of [21] as it does not involve unknown stochastic constraints. However, in the elastic job scheduling problem, observation of the objective function F of $\tilde{\mathcal{P}}$ is not readily available as after making job-size decision $\boldsymbol{x}$, we only observe (noisy) utility values $f_k(x_k)$'s but not $\Delta(\boldsymbol{x}, \Lambda)$. We get around this roadblock by a procedure that constructs confidence interval around $F(\boldsymbol{x})$ using observations of utility function values and realizations of network statistics. We then show that the procedure is qualified according to Definition 2, and thus can be used in conjunction with SCBA.

Before formally presenting the confidence interval construction procedure, we first make explicit what the observation oracle (Definition 2) of the elastic job scheduling algorithm is. Recall the system model in Section 2, under the current assumption that the utility values are immediately observable after making the job-size decision, at each time t, under job-size decision $\boldsymbol{x}(t)$, we observe noisy utility values $\{\hat{f}_k(x_k(t))\}'s$,[7] realizations of job arrivals $a_k(t)$'s, and realizations of offered service $c_m(t)$'s. We will refer to the observations corresponding to $\boldsymbol{x}(t)$ at t, i.e., $\{\hat{f}_k(x_k(t))\}$ for each k, $\{a_k(t)\}$ for each k, $\{c_m(t)\}$ for each m as the set of *query observations* corresponding to $\boldsymbol{x}(t)$, and will often group them together as $\{\hat{f}_k(x_k(t)), a_k(t), c_m(t)\}$. Note that one set of query observations (and one query of SCBA) corresponds to the job-size decision of one time slot in the elastic job scheduling problem. Based on this, we can interpret the observation model of the elastic job scheduling problem as an oracle that outputs a set of query observations $\{\hat{f}_k(x_k), a_k, c_m\}$ when we query the oracle at $\boldsymbol{x}$, where $\mathbb{E}[\hat{f}_k(x_k)] = f_k(x_k)$, $\mathbb{E}[a_k] = \lambda_k$, and $\mathbb{E}[c_m] = \mu_m$. Note that although a_k's and c_m's are not dependent on the point of query $\boldsymbol{x}$, we can still include them in the output of the oracle.

In SCBA, confidence intervals of the function values are constructed based on the results of repeated queries at the same point, which means we may make the same job-size decisions for multiple time slots. For this purpose, our procedure needs to take as input the query observations of multiple consecutive slots (e.g. $\{\hat{f}_k(x_k(t)), a_k(t), c_m(t)\}, \ldots, \{\hat{f}_k(x_k(t')), a_k(t'), c_m(t')\}$ with $\boldsymbol{x}(t) = \boldsymbol{x}(t+1) = \cdots = \boldsymbol{x}(t') = \boldsymbol{x}$) and output a confidence interval $[LB_{\boldsymbol{x}}, UB_{\boldsymbol{x}}]$ around the true value of $F(\boldsymbol{x})$. Thus, the input to the confidence interval construction procedure can be considered as independent sets of query observations corresponding to a same generic job-size vector $\boldsymbol{x}$. As the observations $\{\hat{f}_k(x_k)\}$ come from utility observations of the jobs, and observations $\{a_k\}$, $\{c_m\}$ correspond to the realizations of the arrival and service processes, all values of the query observations lie in the interval $[0, C]$.

The details of the confidence interval construction procedure are shown in **Algorithm** 1. Since for a given network topology, the set Λ is fully parameterized by the arrival rates $\{\lambda\}_k$ and service rates $\{\mu\}_m$, we will write the function $\Delta(\boldsymbol{x}, \Lambda)$ equivalently as $\Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$, where $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are the vector representation of arrival rates and service rates respectively. The intuition behind **Algorithm** 1 is that, as the observations $\{\hat{f}_k(x_k)\}$, $\{a_k\}$ and $\{c_m\}$ are independent and unbiased samples of $\{f_k(x_k)\}$, $\{\lambda\}_k$ and $\{\mu\}_m$, we can construct confidence intervals of each component of $\{f_k(x_k)\}$, $\{\lambda\}_k$ and $\{\mu\}_m$ such that the true values are constrained to lie in the intervals with high probability. As we will show, the objective function F is Lipschitz continuous with respect to those components. Hence, confidence intervals around $\{f_k(x_k)\}$, $\{\lambda\}_k$ and $\{\mu\}_m$ can be translated into a confidence interval around $F(\boldsymbol{x})$. More specifically, the construction procedure first computes the empirical means of the samples, and then constructs a lower and an upper estimate for $f_k(x_k)$'s (Line 3), λ_k's (Line 5) and μ_m's (Line 8). For notational simplicity, we assume the lower and upper estimates of utility values are all in $[0, C]$,

[7] As we set the size of all class-k jobs at t to be $\boldsymbol{x}(t)$, we may receive multiple observations of $\hat{f}_k(x_k(t))$. It suffices to use one of them for the query observations of $\boldsymbol{x}(t)$. The assumption that there is at least one arrival of each class was made so that we obtain at least one observation for each class every time, which simplifies the presentation of the policy.

and the estimates of λ_k and μ_m are all in $[1, C]$. Otherwise, we can simply project the estimates into the interval $[0, C]$ or $[1, C]$ and the results still hold since the true values of those components lie in the corresponding interval. Then, the lower estimate $LB_{\boldsymbol{x}}$ of $F(\boldsymbol{x})$ is computed by combining the lower estimates of utilities and upper estimates of constraint violation, while the upper estimate $UB_{\boldsymbol{x}}$ is computed by combining the upper estimates of utilities and the lower estimates of constraint violation. Note that in lines 9 and 10, $\Delta(\boldsymbol{x}, \boldsymbol{\lambda}^U, \boldsymbol{\mu}^L)$ ($\Delta(\boldsymbol{x}, \boldsymbol{\lambda}^L, \boldsymbol{\mu}^U)$) is the constraint violation of $\boldsymbol{x}$ with respect to the capacity region of a network with arrival rates $\boldsymbol{\lambda}^U$ ($\boldsymbol{\lambda}^L$) and service rates $\boldsymbol{\mu}^L$ ($\boldsymbol{\mu}^U$), and can be computed by solving a simple linear program (see Appendix C).

Algorithm 1 Confidence Interval Construction

Input: For a given $\boldsymbol{x}$, independent sets of observations $\{\hat{f}_k(x_k), a_k, c_m\}$'s; Width parameter $\bar{\gamma} > 0$.
Output: Confidence Interval $[LB_{\boldsymbol{x}}, UB_{\boldsymbol{x}}]$ for $F(\boldsymbol{x})$
1: **for** $k = 1, \ldots, K$ **do**
2: $\bar{f}_k(x_k) :=$ empirical mean of the observations $\hat{f}_k(x_k)$'s.
3: $f_k^L(x_k) := \bar{f}_k(x_k) - \bar{\gamma}/2$, $f_k^U(x_k) := \bar{f}_k(x_k) + \bar{\gamma}/2$.
4: $\bar{\lambda}_k :=$ empirical mean of the observations a_k's.
5: $\lambda_k^L := \bar{\lambda}_k - \bar{\gamma}/2$, $\lambda_k^U := \bar{\lambda}_k + \bar{\gamma}/2$.
6: **for** $m = 1, \ldots, M$ **do**
7: $\bar{\mu}_m :=$ empirical mean of the observations c_m's.
8: $\mu_m^L := \bar{\mu}_m - \bar{\gamma}/2$, $\mu_m^U := \bar{\mu}_m + \bar{\gamma}/2$
9: $LB_{\boldsymbol{x}} := \sum_{k=1}^K \lambda_k^L f_k^L(x_k) - C(L+1)\Delta(\boldsymbol{x}, \boldsymbol{\lambda}^U, \boldsymbol{\mu}^L)$.
10: $UB_{\boldsymbol{x}} := \sum_{k=1}^K \lambda_k^U f_k^U(x_k) - C(L+1)\Delta(\boldsymbol{x}, \boldsymbol{\lambda}^L, \boldsymbol{\mu}^U)$.

We now establish the validity of the confidence interval construction procedure, showing that it can be used in conjunction with SCBA to solve $\tilde{\mathcal{P}}$.

Proposition 4. *For function F, $\sigma = C^2D^2\log(2K+M)$, **Algorithm** 1 is a qualified confidence interval construction procedure.*

Proof. Recalling Definition 2, to prove the proposition, we will show that for any $\boldsymbol{x} \in [0, B]^K$, $\gamma > 0$, $0 < \delta \leq 1/2$, given $\lfloor \frac{\sigma \log(1/\delta)}{\gamma^2} \rfloor$ independent sets of query observations $\{\hat{f}_k(x_k), a_k, c_m\}$'s, **Algorithm** 1 with width parameter $\bar{\gamma} = \gamma/D$ outputs a confidence interval $[LB_{\boldsymbol{x}}, UB_{\boldsymbol{x}}]$ such that $UB_{\boldsymbol{x}} - LB_{\boldsymbol{x}} \leq \gamma$ and $\mathbb{P}\{F(\boldsymbol{x}) \in [LB_{\boldsymbol{x}}, UB_{\boldsymbol{x}}]\} \geq 1 - \delta$. The proof proceeds in three steps. First, in Lemma 4, we show that the confidence interval output by **Algorithm** 1 has bounded width. Next, in Lemma 5, we prove that if the utility values and network statistics are contained in their lower and upper estimates used in the procedure, then the constructed confidence interval contains the true value of $F(\boldsymbol{x})$. The proof of Lemmas 4 and 5 is deferred to Appendix B.

Lemma 4. *For any $\boldsymbol{x} \in [0, B]^K$, $UB_{\boldsymbol{x}} - LB_{\boldsymbol{x}} \leq D\gamma$ where $D = 3CK + C(L+1)(KB+M)$.*

Lemma 5. *If for all k, $f_k(x_k) \in [f_k^L(x_k), f_k^U(x_k)]$, $\lambda_k \in [\lambda_k^L, \lambda_k^U]$ and for all m, $\mu_m \in [\mu_m^L, \mu_m^U]$, then $F(\boldsymbol{x}) \in [LB_{\boldsymbol{x}}, UB_{\boldsymbol{x}}]$*

Finally, based on the two lemmas, we show that the confidence interval construction procedure is qualified by establishing that the utility values and network statistics are contained in their lower and upper estimates with high probability. Specifically, given a width parameter of value γ/D, by Lemma 4, the confidence interval output by **Algorithm** 1 satisfies $UB_{\boldsymbol{x}} - LB_{\boldsymbol{x}} \leq \gamma$. Furthermore, as the noise of each utility observation is a variable bounded in $[0, C]$, if given $\lceil \frac{\sigma \log(1/\delta)}{\gamma^2} \rceil \geq \frac{\log((2K+M)/\delta)}{(\gamma/D)^2}$ independent samples of $f_k(\boldsymbol{x}_k)$, by Hoeffding's inequality, the lower and upper estimates used by **Algorithm** 1 satisfies $\mathbb{P}\{f_k(x_k) \in [f_k^L(x_k), f_k^U(x_k)]\} \geq 1 - \delta/(2K+M)$. The same holds for $\{\lambda_k\}$ and $\{\mu_m\}$. Hence, by union bound, with probability at least $1 - \delta$, all the components of $\{f_k(x_k)\}$, $\{\lambda_k\}$ and $\{\mu_m\}$ lie in the lower and upper estimates used in **Algorithm** 1. Therefore, invoking Lemma 5, we have $\mathbb{P}\{F(\boldsymbol{x}) \in [LB_{\boldsymbol{x}}, UB_{\boldsymbol{x}}]\} \geq 1 - \delta$, and conclude the proof. □

To summarize, for the optimization component of the policy, we consider the transformed optimization problem $\tilde{\mathcal{P}}$ associated with the elastic job scheduling problem. The optimization component uses **Algorithm** 1 in conjunction with SCBA to make job-size decisions. Starting from $t = 1$, each job-size decision $\boldsymbol{x}(t) = (x_1(t), \ldots, x_K(t))$ corresponds to a query output by SCBA. **Algorithm** 1 takes the query observations as input and computes the confidence intervals as required by SCBA to generate subsequent queries. The size of each class-k job in $A_k(t)$ is set to $x_k(t)$, which makes the job size decisions of one time slot exactly correspond to one query of SCBA. The time-horizon of the elastic job scheduling problem is thus the same as the time horizon of SCBA. Note that the decision made at time t by the optimization component is based on query observations at $1, \ldots, t-1$, which makes it independent of the arrivals at t. Hence, our policy satisfies the condition of Corollary 1. We give the detailed workflow of the optimization component of our policy in Appendix D.

5.2. The routing component

The routing component of our policy chooses a designated server for each job. It is based on a Join-the-Shortest-Queue type rule. At time t, we route all the class k jobs that arrived, to the server with the smallest queue length (workload backlog) among the ones to which scheduler u_k is connected to. More formally, each job $j \in A_k(t)$ is sent to the server s_j such that $s_j \in \arg\min_{s_m \in S_{u_k}} Q_m(t)$. Note that our scheduling component can be interpreted as a special case of Back-Pressure routing in the context of a single-hop network. Combining the optimization component (Section 5.1 and Appendix D) and the routing component, we summarize our policy, CI-JSQ, for the elastic job scheduling problem in **Algorithm 2**.

At each time t, we can construct a set of routing variables $\{\alpha_{km}(t)\}$ in the definition of the capacity region Λ based on the decisions made by **Algorithm 2** by setting $\alpha^{jsq}_{km}(t) := 1$ if the jobs in $A_k(t)$ are sent to server s_m and $\alpha^{jsq}_{km}(t) := 0$ otherwise. The set of routing variables will be used in the analysis of the policy.

Algorithm 2 The CI-JSQ Policy

1: **for** $t = 1, \ldots, T$ **do**
2: $\boldsymbol{x}(t) :=$ the job size vector output by SCBA on $\tilde{\mathcal{P}}$ using **Algorithm 1** as the confidence interval construction procedure.
3: **for** $k = 1, \ldots, K$ **do**
4: Set the size of each job in $A_k(t)$ to $x_k(t)$, i.e., the k-th coordinate of $\boldsymbol{x}(t)$.
5: Send the jobs in $A_k(t)$ to server $s_j \in \arg\min_{s_m \in S_{u_k}} Q_m(t)$

5.3. Performance analysis

In this section, we analyze the theoretical performance guarantee of the CI-JSQ policy. We will show that the policy achieves $\tilde{O}(\sqrt{T})$-regret. From Corollary 1, we can see that the upper bound on the regret can be decomposed into two terms $\sum_{t=1}^{T}\sum_{k=1}^{K}\lambda_k \mathbb{E}[f_k(x_k^*) - f_k(x_k(t))]$ and $L\sum_{m=1}^{M}\mathbb{E}[Q_m(T)]$. The first term tracks the cumulative loss with respect to the utility functions, which we will refer to as **utility regret**, while the second term tracks the queue backlog at the end of the time horizon, which we will refer to as **queueing regret**. To analyze the regret of the CI-JSQ policy, we first invoke the regret guarantee of the optimization component to provide bounds on the utility regret and cumulative constraint violation $\sum_{t=1}^{T}\Delta(\boldsymbol{x}(t), \Lambda)$. Subsequently, based on the cumulative constraint violation and properties of the routing component, we bound the queueing regret.

In the following, we will show that the utility regret $\sum_{t=1}^{T}\sum_{k=1}^{K}\lambda_k[f_k(x_k^*) - f_k(x_k(t))]$, cumulative constraint violation $\sum_{t=1}^{T}\Delta(\boldsymbol{x}(t), \Lambda)$, and the queueing regret $\sum_{m=1}^{M}Q_m(T)$ are all in $O(\sqrt{T})$ with probability at least $1 - 1/T$ (or $1 - O(1/T)$). That the expectation of these quantities are in $\tilde{O}(\sqrt{T})$ immediately follow from the with-high-probability bounds since they are all almost surely bounded by $O(T)$. To avoid unnecessary repetition, we will only state the with-high-probability bounds and it should be understood that the bounds also hold in expectation.

5.3.1. Utility regret

To bound the utility regret, we start from the performance guarantee of the optimization component which will give a bound on $\sum_{t=1}^{T}\sum_{k=1}^{K}F_k(x_k^*) - F_k(x_k(t))$, where F is the objective function of $\tilde{\mathcal{P}}$. Note that this does not directly lead to a bound on the utility regret $\sum_{t=1}^{T}\sum_{k=1}^{K}\lambda_k[f_k(x_k^*) - f_k(x_k(t))]$, which is essentially with respect to the objective function of $\mathcal{P}$. We further use the structure of F to bound the utility regret.

Lemma 6. *The job size vectors $\boldsymbol{x}(t), t = 1, \ldots, T$ of the CI-JSQ policy satisfy $\sum_{t=1}^{T}F(\boldsymbol{x}^*) - F(\boldsymbol{x}(t)) = \tilde{O}(\sqrt{T})$ with probability at least $1 - 1/T$, where $\boldsymbol{x}^*$ is the optimal solution to $\tilde{\mathcal{P}}$.*

Proof. The lemma directly follows from Proposition 4, which establishes that **Algorithm 1**, and the generalized performance guarantee of SCBA (Proposition 3). □

Recall that from Lemma 3, the optimal solution $\boldsymbol{x}^*$ to $\tilde{\mathcal{P}}$ is also optimal for $\mathcal{P}$. Thus, we can bound the utility regret based on Lemma 6 by exploiting the structure of F.

Theorem 2. *The job size vectors $\boldsymbol{x}(t), t = 1, \ldots, T$ of the CI-JSQ policy satisfy $\sum_{t=1}^{T}\sum_{k=1}^{K}\lambda_k[f_k(x_k^*) - f_k(x_k(t))] = \tilde{O}(\sqrt{T})$ with probability at least $1 - 1/T$.*

Proof. Recall that $\boldsymbol{x}^* \in \Lambda$, which implies that $\Delta(\boldsymbol{x}^*, \Lambda) = 0$. Thus, we obtain

$$\sum_{t=1}^{T}F(\boldsymbol{x}^*) - F(\boldsymbol{x}(t)) = \sum_{t=1}^{T}\sum_{k=1}^{K}\lambda_k[f_k(x_k^*) - f_k(x_k(t))] + C(L+1)\sum_{t=1}^{T}\Delta(\boldsymbol{x}(t), \Lambda) \tag{7}$$

$$\geq \sum_{t=1}^{T}\sum_{k=1}^{K}\lambda_k[f_k(x_k^*) - f_k(x_k(t))]. \tag{8}$$

Invoking Lemma 6, we conclude the proof. □

5.3.2. Queueing regret

The queueing regret measures the unfinished workload in the servers at the end of the time horizon. It is related to the excess workload generated by the jobs that the servers cannot process before T. This latter quantity has close connection to the cumulative constraint violation $\sum_{t=1}^{T}\Delta(\boldsymbol{x}(t), \Lambda)$. Therefore, to analyze the queueing regret, we first study the cumulative constraint violation, on which a bound can be recovered from Lemma 6 and will provide a handle on the queueing regret.

Lemma 7. *$\sum_{t=1}^{T}\Delta(\boldsymbol{x}(t), \Lambda) = \tilde{O}(\sqrt{T})$ with probability at least $1 - 1/T$.*

Proof Sketch. Similarly as in the proof of Theorem 2, we start from the performance guarantee with respect to F and show that the cumulative constraint violation $\sum_{t=1}^{T}\Delta(\boldsymbol{x}(t), \Lambda)$ can be upper bounded by a constant times $\sum_{t=1}^{T}F(\boldsymbol{x}^*) - F(\boldsymbol{x}(t))$. We defer the details to Appendix B.

Lemma 7 establishes that the cumulative constraint violation under the zeroth-order scheduling policy is of order $\tilde{O}(\sqrt{T})$. Under the interpretation of Λ as the capacity region of the network, each term $\Delta(\boldsymbol{x}(t), \Lambda)$ represents the excess workload that cannot be handled by the network. The cumulative constraint violation can thus be considered as the total excess workload injected by the end of the time horizon T. As the excess workload (constraint violation) is defined with respect to the full capacity region of the network, a bound on total excess workload can only translate to a bound on total workload backlog at T under a routing policy that effectively utilizes the service capacity of a network. We will show that the routing component of the CI-JSQ policy enjoys such property, and thus establish the bound on the queueing regret of CI-JSQ.

To do so, we first prove some preliminary properties of the routing component of the CI-JSQ policy. We will use the quadratic Lyapunov function of queue length (workload) $\|\boldsymbol{Q}(t)\|^2 = \sum_{m=1}^{M}Q_m^2(t)$. By the dynamics of the workload process (1), under the CI-JSQ policy, the one slot drift of the quadratic Lyapunov function satisfies

$$\frac{1}{2}\|\boldsymbol{Q}(t+1)\|^2 - \frac{1}{2}\|\boldsymbol{Q}(t)\|^2 \leq \sum_{m=1}^{M}Q_m(t)\cdot\left[\sum_{k=1}^{K}a_k(t)\alpha_{km}^{jsq}(t)x_k(t) - c_m(t)\right] + C_1, \tag{9}$$

with C_1 being a constant independent of T, and the routing variables $\{\alpha_{km}^{jsq}(t)\}$ are the ones associated with the CI-JSQ policy. Inequality (9) is formally justified in Appendix B.

In the network scheduling and routing literature, the stability of queues is usually established by showing that the right-hand-side of (9) is upper-bounded by some non-positive quantity and thus the Lyapunov function of queue lengths has non-positive one-slot conditional expected drift. In the analysis of queue length regret of the CI-JSQ policy, the aforementioned argument does not work since the job-size variable $x_k(t)$'s may not be in the network capacity region, so the right-hand-side of (9) may not be non-positive. Instead, we will analyze the upper bound of the drift (9) in a different approach, showing that although the one-slot drift may be positive at some time slots, the cumulative drift is upper-bounded throughout the whole time horizon. This is done in Lemma 8, the proof of which is deferred to Appendix B.

Lemma 8. *With probability at least $1 - O(1/T)$, for all t,*

$$\sum_{m=1}^{M}Q_m(t)\cdot\left[\sum_{k=1}^{K}a_k(t)\alpha_{km}^{jsq}(t)x_k(t) - c_m(t)\right] \leq \sum_{m=1}^{M}Q_m(t)(C_3C_2 + C_4)\sqrt{T}\log T,$$

where C_2, C_3, C_4 are constants independent of T.

Now, we are ready to prove that the queueing regret of CI-JSQ is in $\tilde{O}(\sqrt{T})$ with high probability.

Theorem 3. *Under the CI-JSQ policy, $\sum_{m=1}^{M}Q_m(T) = \tilde{O}(\sqrt{T})$ with probability at least $1 - O(1/T)$.*

Proof. Let C_0 be a constant such that $C_0 \geq 2M\sqrt{C_0}(C_3C_2 + C_4) + 2C_1$, e.g., $C_0 = 4M^2(C_3C_2 + C_4 + 2C_1)^2$. We will show by induction that $\|\boldsymbol{Q}(t)\|^2 \leq C_0T\log^2(T+1)$ for $t = 1, \ldots, T$. The theorem will then follow from that $\sum_{m=1}^{M}Q_m(T) \leq \sqrt{M}\|\boldsymbol{Q}(T)\|$. The base case of the induction ($t = 1$) holds trivially. Assume that the statement holds for $\tau = 1, \ldots, t$. Summing over (9) for $\tau = 1, \ldots, t$ and using Lemma 8, we have

$$\|\boldsymbol{Q}(t+1)\|^2 \leq 2\sum_{\tau=1}^{t}\sum_{m=1}^{M}Q_m(\tau)\cdot(C_3C_2 + C_4)\sqrt{T}\log T + 2C_1T$$

$$\leq 2M\sqrt{C_0 T}\log T \cdot (C_3C_2 + C_4)\sqrt{T}\log T + 2C_1T \tag{10}$$

$$= 2M\sqrt{C_0}(C_3C_2 + C_4)T\log^2 T + 2C_1T, \tag{11}$$

where we have used the induction hypothesis in (10). As $C_0 \geq 2M\sqrt{C_0}(C_3C_2 + C_4) + 2C_2$, we have from (11) that $\|\boldsymbol{Q}(t+1)\| \leq C_0\sqrt{T}\log T$. Thus, we finish the induction and conclude the proof. □

Combining Theorems 2 and 3 and Corollary 1, we have that the CI-JSQ policy achieves $\tilde{O}(\sqrt{T})$ regret, which we formally summarize in the following theorem.

Theorem 4. *Let π^{CI-JSQ} denote the CI-JSQ policy. $R(\pi^{CI-JSQ}, T) = \tilde{O}(\sqrt{T})$.*

5.3.3. Remarks

- **Computational Aspect:** The main computational cost of the CI-JSQ policy comes from the confidence interval construction procedure and SCBA. As SCBA is a generalization of the ellipsoid algorithm, it has a similar computational complexity as the ellipsoid algorithm. We refer the reader to [21] for a more detailed that front. For the confidence interval construction procedure, the most computational intensive steps are computing the functions $\Delta(\boldsymbol{x}, \boldsymbol{\lambda}^L, \boldsymbol{\mu}^U)$ and $\Delta(\boldsymbol{x}, \boldsymbol{\lambda}^U, \boldsymbol{\mu}^L)$ (Lines 9 and 10 of Algorithm 1). Each of the function can be computed by solving a linear program. Due to space limitation, we defer further details to Appendix C.
- **Relation to Zeroth-order Optimization with Stochastic Constraints:** We believe that the recipe of constructing a transformed optimization problem by converting the stochastic constraints as a penalty term in the objective function, designing a procedure that outputs confidence interval of the value of the transformed objective function using observations available in the problem setting, and using it in conjunction with SCBA can be extended to handle more general zeroth-order optimization problems. The recipe may also be able to work with other algorithms for zeroth-order optimization, extending them to handle unknown and stochastic constraints, of which a common example is the capacity constraints in network routing and scheduling. In previous works, unknown and stochastic constraints are typically handled through primal–dual gradient-based methods. When applied to zeroth-order optimization problems, those methods rely on using zeroth-order feedback to construct approximate gradients. Such approximation has large variance when the feedback is noisy and it typically leads to sub-optimal regret [20,25]. In contrast, our approach works with algorithms that directly utilizes zeroth-order feedback [21,26], which are more robust against noisy feedback and can achieve order-optimal regret.
- **Dependence of Regret on Dimension:** Although the CI-JSQ policy achieves optimal regret with respect to the time horizon T, its regret dependence on dimension (number of job classes), which is dominated by the regret bound of SCBA can be large and sub-optimal. We further discuss the implication of this in simulations. Improving the dependence of the regret on number of classes is an important future direction.

5.4. Dealing with feedback delay

So far, we have assumed that the utility observations are immediately available after job size decisions. In the original setting, the utility values are observed after the completion of the jobs. We now extend the CI-JSQ policy to deal with such feedback delay.

We propose an episodic version of the CI-JSQ policy (E-CI-JSQ) that achieves the same order of $\tilde{O}(\sqrt{T})$-regret. In the E-CI-JSQ policy, the time horizon is divided into $\lfloor T/(KB)\rfloor$ episodes with each episode having KB slots. The job-size decision remains unchanged during an episode. In each episode, the E-CI-JSQ policy designates the first job of each class as a sampling job. The sampling jobs receive priority service at the servers while the other jobs get served in a First-Come-First-Serve order. Note that as the size of each job is at most B and the realized service rate at each time is at least 1, we have that for any job-size decision $\boldsymbol{x}$, it takes the system at most KB time slots to complete one job of each class. Hence, the sampling jobs always complete by the end of each episode. Therefore, for each episode, the utility observations of the sampling jobs and the realizations of arrivals and services at the first slot of the episode can be used as a sample for constructing confidence intervals. Based on this, at the episode level, the confidence interval construction procedure can still be used in conjunction with SCBA to compute job-size decisions for the E-CI-JSQ policy. Note that here only the utility observations of the sampling jobs are fed into the confidence interval construction procedure. Since we only update the job-size decision vector for $\lfloor T/(KB)\rfloor$ times (every episode), the time horizon of SCBA is set to be $\lfloor T/(KB)\rfloor$. The routing decisions are still made based on the Join-the-Shortest-Queue rule. Unlike the job-size decisions, the routing decisions are still updated every time slot. The details of the E-CI-JSQ policy are shown in **Algorithm 3**. The E-CI-JSQ policy also enjoys $\tilde{O}(\sqrt{T})$-regret, the proof of which is shown in Appendix A.3.

6. Simulations

In this section, we evaluate the empirical performance of the CI-JSQ policy. We will first study the behavior of the job size decisions and queue length under the policy, and then compare its regret performance with the policy proposed in [20].

Algorithm 3 The E-CI-JSQ Policy

for $t = 1, \ldots, T$ **do**
  if t is the first slot of an episode **then**
    $\boldsymbol{x}(t) :=$ the vector output by **Algorithm 1** in conjunction with SCBA (with time horizon $\lfloor T/(KB) \rfloor$) on $\tilde{\mathcal{P}}$ executed at the episodic level.
    Set one job of each class as sampling job that will receive priority service.
  else
    Keep the job-size decision vector unchanged, i.e., $\boldsymbol{x}(t) := \boldsymbol{x}(t-1)$.
  for $k = 1, \ldots, K$ **do**
    Set the size of jobs in $A_k(t)$ as $x_k(t)$.
    Send the jobs in $A_k(t)$ to server $s_j \in \arg\min_{s_m \in S_{u_k}} Q_m(t)$

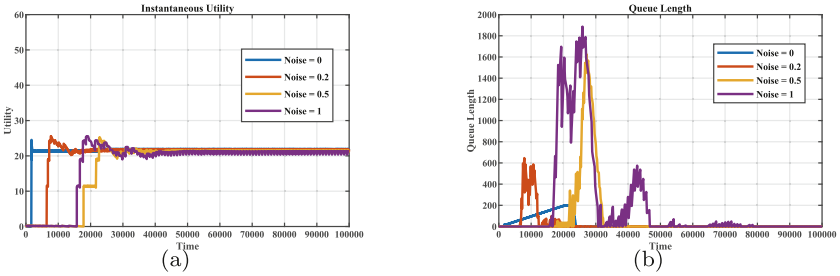

Fig. 2. Instantaneous Utility and Queue Length of CI-JSQ under Different Noise Levels.

6.1. Instantaneous utility and queue length

We construct a bipartite network with 10 job schedulers (corresponding to 10 job classes) and 20 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). At every time slot, the arrival rate of each class is a uniform random variable in $\{2, 3, 4, 5\}$ while the service rate of each server is a uniform random variable in $[2, 4]$. We assign an underlying utility function to each class of one of 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). The time horizon is set to $T = 100000$ slots. We vary the level of the observation noise from 0 (no noise) to 1 (each observation is corrupted with noise that is uniformly distributed in $[-1, 1]$).

We first plot the instantaneous utility of the policy with time in Fig. 2(a). The instant utility at time t is defined as $\sum_{k=1}^{K} \lambda_k f_k(x_k(t))$ where $\{x_k(t)\}$ is the job-size decision by the CI-JSQ policy. The optimal value of the optimization problem $\mathcal{P}$ corresponding to our simulation setup is 22. From Fig. 2(a), we see that the instantaneous utility of the CI-JSQ converges to the optimal value under all noise levels, where the convergence time increases with the noise level.

Next, we plot the evolution of (total) queue length with time in Fig. 2(b) under CI-JSQ. For most of the time, the queue length of CI-JSQ stays at a low level, suggesting that the job-size vector approaches the optimal from within the capacity region. Furthermore, we can observe that generally higher noise levels result in larger fluctuation in queue length.

6.2. Regret performance

We proceed to evaluate the regret performance of the CI-JSQ policy. As the utility of the optimal dynamic policy is difficult to compute, we use $T \cdot OPT(\mathcal{P})$ as an approximation to the optimal utility. In addition to the previously constructed 10-class network, we apply the policies to another larger network with 50 classes (job schedulers) and 100 servers (all other parameters are the same as the previous small network) to evaluate the policies' scalability with respect to the network size.

Algorithm for Comparison: We compare our CI-JSQ policy to the Gradient-Sampling Max-Weight (GSMW) policy proposed in [20]. The GSMW policy is a general policy for network utility maximization with unknown utility function. Applied to the elastic job scheduling problem, GSMW uses observations of utility value to construct approximate gradient of the utility function, i.e. $\nabla f_k(x) \simeq \frac{\hat{f}_k(x+\delta) - \hat{f}_k(x-\delta)}{2\delta}$ and also employs Join-the-Shortest-Queue routing. GSMW policy can only achieve a sub-optimal $\tilde{O}(T^{3/4})$-regret for the elastic job scheduling problem. For completeness, we provide further details and discussion of the GSMW policy in Appendix E.

We plot the regrets of CI-JSQ and GSMW in the 10-class and 50-class networks in Figs. 3 and 4 respectively. We can see that in the 10-class network, when the noise level is 0, GSMW and CI-JSQ have comparable regret performance.

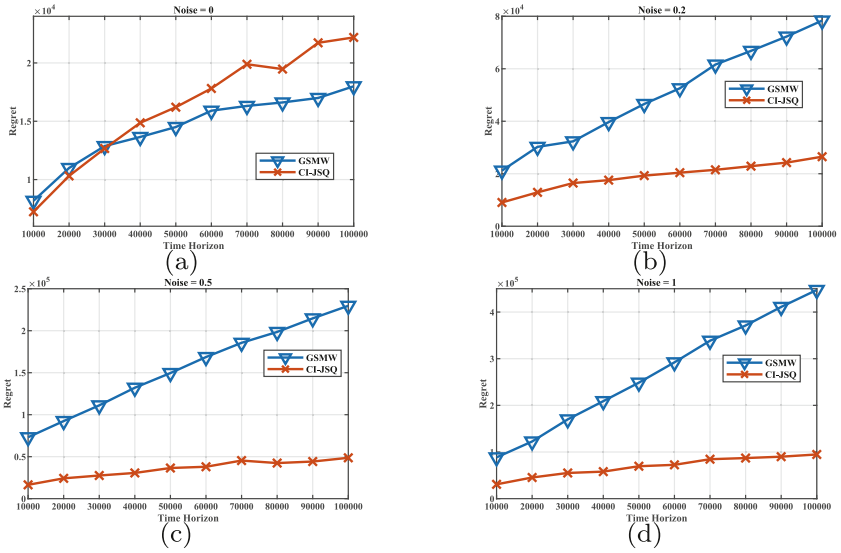

Fig. 3. Regret of CI-JSQ and GSMW under different noise levels in the 10-class network.

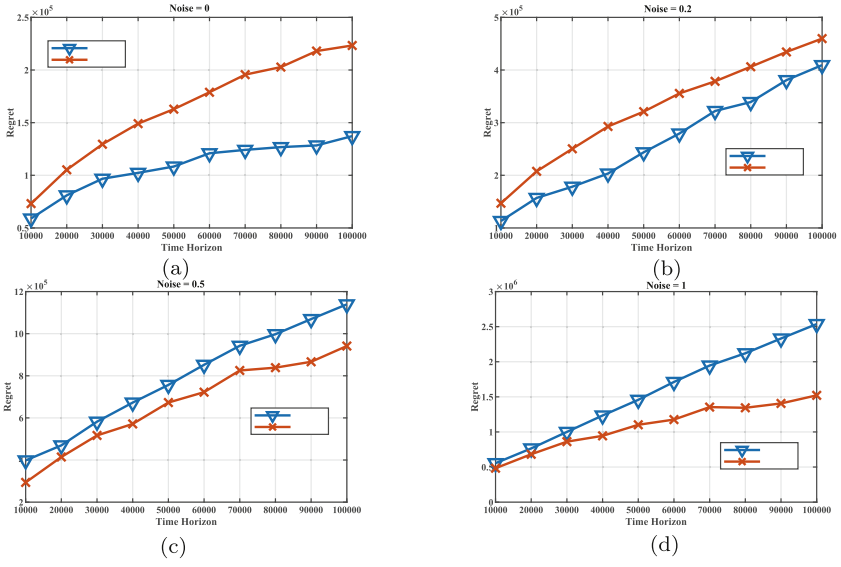

Fig. 4. Regret of CI-JSQ and GSMW under different noise levels in the 50-class network.

However, CI-JSQ gradually outperforms GSMW by larger margin as the noise level increases. This can be attributed to that CI-JSQ directly utilizes the utility value observations rather than using them to construct approximate gradients for job-size decisions, which is more robust to noise in the observations. On the other hand, GSMW scales better than CI-JSQ with the network size, as can be seen from Fig. 4. Indeed, as mentioned before, one drawback of the CI-JSQ policy is that, despite achieving optimal regret with respect to the time horizon T, its regret bound has large dependence on the dimension (i.e., number of classes in the elastic job scheduling problem) of the problem.

7. Related works

In this section, we survey existing results in the related fields and discuss the potential of applying those results to solve the elastic job scheduling problem.

7.1. Zeroth-order/bandit convex optimization

Zeroth-order convex optimization refers to the problem of minimizing a convex function f over a (known) convex and compact domain $\mathcal{X}$ with access to a zeroth-order oracle of f [21,27–31]. At each iteration t, we are allowed to

query a point $\boldsymbol{x}(t)$ and receive noisy/noiseless feedback $\hat{f}(\boldsymbol{x}(t))$ with $\mathbb{E}[\hat{f}(\boldsymbol{x}(t))] = f(\boldsymbol{x}(t))$. The goal is to design algorithms with low regret or optimization error over a time horizon of T. The noiseless feedback scenario was studied in [28]. For the noisy feedback scenario, which is more relevant to the setting of the elastic job scheduling problem, a regret lower bound of $\Omega(\sqrt{T})$ has been established even for simple functions such as linear [27] and quadratic [30] functions. While the algorithm with order-optimal regret for linear functions was proposed in [27], the stochastic convex bandit algorithm [21] is the state-of-the-art algorithm that achieves order-optimal regret for general convex functions. Bandit convex optimization considers a more challenging setting, where an adversary chooses a sequence of functions $f_1, \ldots, f_T$ and at each time t we receive noisy feedback of $f_t(x_t)$. The goal is to minimize the regret compared to the best fixed point, i.e., $\mathbb{E}[\sum_{t=1}^{T} f_k(x_t)] - \min_{x\in\mathcal{X}} \sum_{t=1}^{T} f_t(x)$. The setting is harder than the zeroth-order convex optimization one and the regret bounds achieved by algorithms in the literature are typically larger. For general convex functions, $\tilde{O}(T^{3/4})$-regret is achieved by algorithms in [22] and [32]. The algorithm in [33] improves the bound to $\tilde{O}(T^{8/13})$. For strongly convex and smooth functions, [34] achieves $\tilde{O}(\sqrt{T})$-regret. Algorithms proposed for zeroth-order or bandit convex optimization rely on the feasibility region $\mathcal{X}$ to be known in advance, which prevents them from being applied to the elastic job scheduling problem.

7.2. Bandits with knapsacks

Bandits with knapsacks problem [23,24,35] is a version of the multi-armed bandits where each arm is associated with a reward and a resource consumption vector. When an arm is pulled, we obtain noisy observations of the arm's reward and resource consumption vector. The goal is to maximize the total reward over a time horizon T subject to knapsack constraints, that the total resource consumption does not exceed some budget. Let m be the number of arms. The state-of-art algorithms achieve a regret of $\tilde{O}(\sqrt{mT})$, which matches the lower bound of the problem.

We briefly discuss the potential of applying bandits with knapsack algorithms to the elastic job scheduling problem. The knapsack constraints share similarity with the constraints in the elastic job scheduling problem. However, in the bandits with knapsacks problem, the decision region is a discrete set of arms whereas in our problem the decision region is a continuous set. Therefore, in order to apply the algorithms in [23,24] to our problem, we need to discretize the decision region, and thereby creating a correspondence between job-sizes and arms. For intuitive argument, assume that the decision region is uniformly discretized into m arms, then the discretization will incur $O(1/m)$ error every time slot, which will accumulate to $O(T/m)$-regret over the time horizon. Combining this with the inherent regret of the algorithms, we see that applying bandit with knapsack algorithms would achieve $O(\sqrt{mT} + T/m)$-regret for the elastic job scheduling problem. The optimal number of arms is $m = O(T^{1/3})$, which leads to $O(T^{2/3})$-regret that is worse than the $\tilde{O}(\sqrt{T})$ of CI-JSQ. This can be attributed to that bandits with knapsack techniques do not take advantage of the concavity of the utility functions.

7.3. Reinforcement learning

Reinforcement learning studies finite-horizon Markov Decision Processes (MDPs) with unknown dynamics. Recently, there have been several works that propose reinforcement learning methods [36–40] that explicitly learn the parameters of the MDPs through empirical observations and have provable regret bounds. Our elastic job scheduling problem can be modeled in the MDP framework, with the state being the queue lengths and the action being the job-size decision. However, applying state-of-the art reinforcement learning methods cannot achieve an order-optimal regret bound of $\tilde{O}(\sqrt{T})$. For example, the UCRL algorithm proposed in [36] has a regret bound of $\tilde{O}(S\sqrt{AT})$ where S is the cardinality of the state space, A is the cardinality of the action space and T is the time horizon. Since in the elastic job scheduling problem, our state space and action space are both continuous, to apply UCRL, we must first perform necessary discretization. Even if we ignore the discretization error from state space and focus on the action space (set of job-size decisions), by discretizing the set of job-size decisions into A actions, similar to the preceding argument with bandits with knapsacks, the resulting discretization error leads to a regret of $O(T/A)$. Combining this with the inherent regret of UCRL, we arrive at a total regret of $\tilde{O}(\frac{T}{A} + S\sqrt{AT})$. Selecting the optimal value of A that minimizes the regret bound ($A = O(T^{1/3})$), the resulting regret is still $\tilde{O}(T^{2/3})$, which is worse than the $\tilde{O}(\sqrt{T})$ regret achieved by our CI-JSQ algorithm. The key reason is that reinforcement learning methods are designed for general MDPs and do not exploit the special structure of the elastic job scheduling problem. The lower bounds [36,37] proposed in reinforcement learning also do not apply since the problem instances constructed to prove the lower bounds do not satisfy the properties of the elastic job scheduling problem.

7.4. Network utility maximization and queueing bandits

Network utility maximization (NUM) is class of problems that study allocating network resources (e.g. traffic rates, link bandwidth) so as to maximize overall network utility [41–44]. If we view the job size in the elastic job scheduling problem as traffic rate, then philosophically, the elastic job scheduling problem can be considered as one instance of NUM in single-hop networks, with the unknown stochastic constraints corresponding to network stability constraints in stochastic NUM formulations [43,44]. As most previous works on NUM focus on the setting where the utility functions are known in advanced, their results cannot be applied to the elastic job scheduling problem. An exception is the GSMW algorithm proposed in [20]. However, as discussed before, GSMW can only achieve a sub-optimal regret of $\tilde{O}(T^{3/4})$ in the elastic job scheduling problem.

7.5. Queueing bandits

Queueing bandits studies the problem of routing unit-size packets to servers to minimize the expected queue length (queueing regret with respect to the optimal policy) in a multi-server system with each server having an unknown service rate [45–48]. It is in principle similar to the classical multi-armed bandit problem, but the queueing dynamics make the queueing regret exhibit more complex behavior than the regret in classical bandit problems [45–48]. The techniques therein are not applicable for the elastic job scheduling problem as queueing bandits does not involve making decisions on job sizes.

8. Conclusion

In this paper, we have studied the problem of elastic job scheduling with unknown utility functions. We established an upper bound on the regret of a generic policy that consists of cumulative utility difference between the job-size decisions of the policy and the optimal solution to a static optimization problem, and the total queue length at servers at the end of the time horizon T. The upper bound connects the elastic job scheduling problem to zeroth-order convex optimization with bandit feedback and routing for network stability. Based on the connection, we proposed a policy that achieves an order-optimal regret of $\tilde{O}(\sqrt{T})$ by simultaneously bounding the cumulative utility difference and controlling the total queue length. The policy can also be interpreted as a principled approach to enabling existing algorithms for zeroth-order convex optimization with bandit feedback [21,26] to handle parameterized unknown and stochastic constraints.

Although our policy achieves order-optimal regret with respect to the time-horizon T, its does not scale well with the number of job classes, which is mainly due to the regret bound of the Stochastic Convex Bandit Algorithm embedded in the policy, that has large dependence on the dimension of the problem. While improving the regret dependence on dimension for general zeroth-order convex optimization is a challenging problem and little progress has been made [26], special structural properties of the elastic job scheduling problem, e.g., the separability of the objective function of $\mathcal{P}$, can be exploited to design policies with better dependence on dimension (number of job classes). We leave this as an important direction for future research.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Proof of theorems and propositions

A.1. Proof of Theorem 1

Proof. For any given policy, we first take weighted averages of the sizes of jobs of each class under the policy over the realizations of the arrival processes. We will then show that the averages satisfy the constraints of $\mathcal{P}$, and by the concavity of the underlying utility functions, the corresponding value of the objective function is no less than the expected utility of the policy.

To facilitate the proof, we define the following optimization problem $\mathcal{P}'$:

$$\mathcal{P}': \max_{\{r_k\},\{b_{km}\}} \sum_{k=1}^{K} \lambda_k f_k\left(\frac{r_k}{\lambda_k}\right) \tag{A.1}$$

$$\textbf{s.t.} \quad \sum_m b_{km} = r_k, \quad \forall k \tag{A.2}$$

$$\sum_k b_{km} \le \mu_m \quad \forall m \tag{A.3}$$

$$b_{km} = 0, \quad \forall s_m \notin S_{u_k} \tag{A.4}$$

$$b_{km} \ge 0, \quad \forall k, m \tag{A.5}$$

$$0 \le r_k \le B\lambda_k, \forall k. \tag{A.6}$$

$\mathcal{P}'$ can be interpreted as an reformulation of $\mathcal{P}'$ where r_k is the total load of class k jobs (i.e., $\lambda_k x_k$) and b_{km} is the class k load routed to server s_m (i.e., $\lambda_k \alpha_{km} x_k$). The reason that we define such a reformulation is that while $\mathcal{P}$ is convex over $\{x_k\}$ as optimization variables but possibly non-convex over $\{x_k\}$ and $\{\alpha_{km}\}$, $\mathcal{P}'$ is convex over both $\{r_k\}$ and $\{b_{km}\}$.

We now argue that $\mathcal{P}$ and $\mathcal{P}'$ are equivalent. For any feasible $\{x\}_k$ solution to $\mathcal{P}$, since $(x_1, \ldots, x_K) \in \Lambda$, we have there exists $\{\alpha\}_{km}$ such that $\{x\}_k$, $\{\alpha\}_{km}$ satisfy the conditions of Λ. It follows that we can construct a feasible solution to $\mathcal{P}'$ by

setting $r_k = \lambda_k x_k$ and $b_{km} = \alpha_{km} r_k$. Similarly, given any feasible solution to $\mathcal{P}'$, we can construct a corresponding feasible solution to $\mathcal{P}$ by setting $x_k = r_k/\lambda_k$. Thus, we establish the equivalence of $\mathcal{P}$ and $\mathcal{P}'$.

Proceeding to the proof of Theorem 1, for an arbitrary $\pi \in \Pi^*$, consider a sample path ω of an execution of π. Let $a_k(t,\omega)$ be the number of job arrivals of class k at t, $c_m(t,\omega)$ be the realization of the service rate of s_m at t on the sample path ω. Define $N_k(\omega) := \sum_{t=1}^{T} a_k(t,\omega)$ be the total number of class k jobs that arrived before T, and $C_m(\omega) := \sum_{t=1}^{T} c_m(t,\omega)$ be the total offered service of server m. Further, we define $\bar{x}_k(\omega)$ to be the average (over all arrived jobs of class k) amount of service that class k-jobs *received* before T. Note that $\bar{x}_k(\omega)$ here is the average of received service rather than job-size decision made by the policy. For a job that is finished before T, its received service is equal to the its job size, otherwise its received service is smaller than its size determined by the policy. Let $U(\pi, T, \omega)$ be the utility achieved under policy π under sample path ω. By Jensen's inequality, we have $U(\pi, T, \omega) \le \sum_{k=1}^{K} N_k(\omega) f_k(\bar{x}_k(\omega))$.

For each k, define $x_k := \sum_\omega p(\omega) N_k(\omega)\bar{x}_k(\omega) / \sum_{\omega'} p(\omega') N_k(\omega')$ with $p(\omega)$ being the probability mass of ω.[8] We claim that $(x_1, \ldots, x_K)$ is a feasible solution to $\mathcal{P}$. Indeed, for each sample path ω, define $r_k(\omega) := \frac{N_k(\omega)\bar{x}_k(\omega)}{T}$ and $c_m(\omega) := \frac{C_m(\omega)}{T}$. It follows from the physical constraints of the network that there exists $\{b_{km}(\omega)\}$ such that $\sum_m b_{km}(\omega) = r_k(\omega)$ for all k and $\sum_k b_{km}(\omega) \le c_m(\omega)$ for all m. Also, $\{b_{km}(\omega)\}$ satisfy (A.3) and (A.4). Hence, setting $r_k := \sum_\omega p(\omega) r_k(\omega)$, $b_{km} := \sum_\omega p(\omega) b_{km}(\omega)$ and noting that by definition $c_m = \sum_\omega p(\omega) c_m(\omega)$, we have $\{r_k\}, \{b_{km}\}$ is a feasible solution to $\mathcal{P}'$. Therefore, $(x_1, \ldots, x_K)$ with $x_k = \frac{r_k}{\lambda_k}$ is feasible to $\mathcal{P}$. The claim follows from the fact that $T\lambda_k = \sum_\omega p(\omega) N_k(\omega)$.

Finally, we complete the proof by establishing that $\sum_\omega U(\pi, T, \omega) \le \sum_{k=1}^{K} T\lambda_k f_k(x_k)$. The theorem will then follow from the feasibility of $(x_1, \ldots, x_K)$. Indeed, we use $T\lambda_k = \sum_\omega p(\omega) N_k(\omega)$ and have,

$$\sum_\omega p(\omega) U(\pi, T, \omega) \le \sum_{k=1}^{K} \sum_\omega p(\omega) N_k(\omega) f_k(\bar{x}_k(\omega))$$
$$= \sum_{k=1}^{K} T\lambda_k \frac{\sum_\omega p(\omega) N_k(\omega) f_k(\bar{x}_k(\omega))}{\sum_{\omega'} p(\omega') N_k(\omega')} \le \sum_{k=1}^{K} T\lambda_k f_k(x_k),$$

where the last inequality follows from Jensen's inequality. □

A.2. Proof of Proposition 3

Proof. The proof of Proposition 3 essentially follows from the same analysis as Theorem 1 of [21]. Since results of [21] are not the contribution of this paper, we do not reiterate the analysis here but instead present the main idea behind the proof of Theorem 1 of [21] and demonstrate how it can be applied to prove Proposition 3.

The proof of Theorem 1 of [21] consists mainly of a probability argument and a geometry argument. The probability argument establishes that each confidence interval constructed throughout the algorithm contains the true value with probability $1 - 1/T^2$. In the original setting of [21], the argument holds since for each query, the algorithm receives a noisy but unbiased observation of the function value with the noise being a zero-mean σ-sub-Gaussian random variable. Applying the union bound, it follows that all the confidence intervals contain the true value with probability $1 - 1/T$. Since an error probability of $1/T$ does not affect the order of expected regret, based on the probability argument, the analysis can be carried on in a deterministic fashion assuming that all the confidence intervals contain the true value. Next, the geometry argument (c.f. the one dimensional special case in Section 4) establishes that: (i). the optimal point is always contained in the target region and never eliminated, and (ii). the elimination procedure shrinks the target region fast enough so that the query points (Step 2(a)), which always lie in the target region, approach the optimal point quickly and the regret accumulated through the queries can be bounded by $\tilde{O}(\sqrt{T})$. Note that the geometry argument holds as long as the confidence intervals used by the algorithms have widths bounded by the parameter γ_i (c.f. Step 2) and contain the true function value.

Now, if we plug in a qualified procedure to SCBA, the probability argument holds by taking δ in Definition 2 as $1/T^2$. The procedure constructs confidence intervals that satisfies the width requirement by setting the parameter γ in Definition 2 as the desirable width value (γ_i) throughout the execution of the algorithm. Thus, the geometry argument also holds. It follows that, even without unbiased observations, a qualified procedure in conjunction with the Stochastic Convex Bandit Algorithm has the regret guarantee, which proves Proposition 3. □

A.3. Proof of regret bound of E-CI-JSQ

In this section, we prove the regret bound of the episodic CI-JSQ policy, which is formally summarized in the following theorem.

Theorem 5. *Let $\pi^{E-CI-JSQ}$ denote the E-CI-JSQ policy. $R(\pi^{E-CI-JSQ}) = \tilde{O}(\sqrt{KBT}) = \tilde{O}(\sqrt{T})$.*

[8] For ease of notation, we assume ω lies in a discrete set.

Proof. We index the episodes with $e = 1, \ldots, \lfloor T/(KB) \rfloor$ and denote the job-size vector in episode e as $\boldsymbol{x}(e)$. We first claim that the job-size decision sequence at each episode $\boldsymbol{x}(e)$, $e = 1, \ldots, \lfloor T/(KB) \rfloor$, is equivalent to the query sequence output by SCBA (with horizon $\lfloor T/(KB) \rfloor$) in conjunction with the confidence interval construction procedure (Algorithm 1). Indeed, for the SCBA and the confidence interval construction procedure to work at an episodic level, for each episode e, we need the set of query observations $\{\hat{f}_k(x_k(e)), a_k(e), c_m(e)\}$ corresponding to $\boldsymbol{x}(e)$ to be available by the start of the next episode $e+1$. $\{a_k(e)\}$ ($\{c_m(e)\}$) correspond to realized arrivals (services) for each class (server) and can be taken as the realizations of any time slot during the episode. The utility observations $\{\hat{f}_k(x_k(e))\}$ may not be available right after the job-size decisions are made because of the feedback delay. However, since we designate the first job of each class at the beginning of each episode as a sampling job and the sampling jobs receive priority services, the queueing delay experienced by the sampling jobs are only caused by other sampling jobs in the queue. As each job has size at most B, there are K sampling jobs in each episode, and the realized service of each server is lower bounded by 1, with probability one, all the sampling jobs of one episode finish execution in KB time (the length of an episode). It follows that we receive the utility observations of the sampling jobs, i.e., $\{\hat{f}_k(x_k(e))\}$ by the end of the episode e. Therefore, the set of query observations $\{\hat{f}_k(x_k(e)), a_k(e), c_m(e)\}$ is available to SCBA and the confidence interval construction procedure before computing $\boldsymbol{x}(e+1)$. Hence, the job-size decision sequence at the episode level is equivalent to the query sequence computed by SCBA (with horizon $\lfloor T/(KB) \rfloor$) in conjunction with Algorithm 1.

Using Propositions 3 and 4, we have $\sum_{e=1}^{\lfloor T/(KB) \rfloor} F(\boldsymbol{x}^*) - F(\boldsymbol{x}(e)) = \tilde{O}(\sqrt{T/(KB)})$ both in expectation and with probability at least $1 - \lfloor T/(KB) \rfloor$. Since the job-size decision remains unchanged during each episode, for E-CI-JSQ, $\sum_{t=1}^{T} F(\boldsymbol{x}^*) - F(\boldsymbol{x}(t)) = KB\tilde{O}(\sqrt{T/(KB)}) = \tilde{O}(\sqrt{KBT}) = \tilde{O}(\sqrt{T})$. Therefore, we can carry out the same analysis as in Theorems 2 and 3, and show that the regret of the E-CI-JSQ policy is also in $\tilde{O}(\sqrt{T})$. □

Appendix B. Proof of lemmas

B.1. Proof of Lemma 1

Proof. We show the first part of the lemma. The second part follows immediately from the first part since F is the sum of concave utility functions minus $C(L+1)$ times Δ. Consider any $\boldsymbol{x}$, $\boldsymbol{y}$, we will show that for any $0 \le p \le 1$, $p\Delta(\boldsymbol{x}, \Lambda) + (1-p)\Delta(\boldsymbol{y}, \Lambda) \ge \Delta(p\boldsymbol{x} + (1-p)\boldsymbol{y}, \Lambda)$. As Λ is a closed convex set, there exist $\boldsymbol{z}_1, \boldsymbol{z}_2$ such that $\Delta(\boldsymbol{x}, \Lambda) = \|\boldsymbol{x} - \boldsymbol{z}_1\|_1$ and $\Delta(\boldsymbol{y}, \Lambda) = \|\boldsymbol{y} - \boldsymbol{z}_2\|_1$. Then, using the convexity of l_1 norm, we have

$$p\Delta(\boldsymbol{x}, \Lambda) + (1-p)\Delta(\boldsymbol{y}, \Lambda) = p\|\boldsymbol{x} - \boldsymbol{z}_1\|_1 + (1-p)\|\boldsymbol{y} - \boldsymbol{z}_2\|_1 \ge \|p\boldsymbol{x} + (1-p)\boldsymbol{y} - p\boldsymbol{z}_1 - (1-p)\boldsymbol{z}_2\|_1.$$

Also, $\boldsymbol{z} = p\boldsymbol{z}_1 + (1-p)\boldsymbol{z}_2 \in \Lambda$. It follows from the definition of Δ that

$$p\Delta(\boldsymbol{x}, \Lambda) + (1-p)\Delta(\boldsymbol{y}, \Lambda) \ge \|p\boldsymbol{x} + (1-p)\boldsymbol{y} - p\boldsymbol{z}_1 - (1-p)\boldsymbol{z}_2\|_1 \ge \Delta(p\boldsymbol{x} + (1-p)\boldsymbol{y}, \Lambda). \quad \square$$

B.2. Proof of Lemma 2

Proof. Again, we prove the first part of the lemma, and the second part follows from the first by definition. Consider two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, w.l.o.g., assume $\Delta(\boldsymbol{x}, \Lambda) \ge \Delta(\boldsymbol{y}, \Lambda)$. As Λ is a closed convex set, there exist $\boldsymbol{x}'$, $\boldsymbol{y}'$ such that $\|\boldsymbol{x} - \boldsymbol{x}'\|_1 = \Delta(\boldsymbol{x}, \Lambda)$ and $\|\boldsymbol{y} - \boldsymbol{y}'\|_1 = \Delta(\boldsymbol{y}, \Lambda)$. We have

$$\begin{aligned}
\Delta(\boldsymbol{x}, \Lambda) - \Delta(\boldsymbol{y}, \Lambda) &= \|\boldsymbol{x} - \boldsymbol{x}'\|_1 - \|\boldsymbol{y} - \boldsymbol{y}'\|_1 \\
&\le \|\boldsymbol{x} - \boldsymbol{y}'\|_1 - \|\boldsymbol{y} - \boldsymbol{y}'\|_1 \qquad \text{(B.1)} \\
&\le \|\boldsymbol{x} - \boldsymbol{y}\|_1 \le \sqrt{K}\|\boldsymbol{x} - \boldsymbol{y}\|, \qquad \text{(B.2)}
\end{aligned}$$

where Inequality (B.1) follows from the definition of $\Delta(\cdot, \Lambda)$ and Inequality (B.2) follows from Cauchy–Schwarz inequality. Therefore, the function $\Delta(\cdot, \Lambda)$ is $\sqrt{K}$-Lipschitz Continuous (with respect to the Euclidean norm). □

B.3. Proof of Lemma 3

Proof. By Lemma 1, Δ is a convex function. It follows that the objective function of $\tilde{\mathcal{P}}$ is concave. As the constraint $\forall k, x_k \in [0, B]$ is easily seen to be convex, it follows that $\tilde{\mathcal{P}}$ is a convex problem.

We now proceed to prove that $\mathcal{P}$ and $\tilde{\mathcal{P}}$ have the same set of optimal solutions. Let $\tilde{\boldsymbol{x}}^*$ be an optimal solution to $\tilde{\mathcal{P}}$. We first show that $\tilde{\boldsymbol{x}}^* \in \Lambda$. For the sake of contradiction, if $\tilde{\boldsymbol{x}}^* \notin \Lambda$, i.e., $\Delta(\tilde{\boldsymbol{x}}^*, \Lambda) > 0$, then there exists $\boldsymbol{x}' \ne \tilde{\boldsymbol{x}}^*$ such that $\boldsymbol{x} \in \Lambda$ and $\|\tilde{\boldsymbol{x}}^* - \boldsymbol{x}\| = \Delta(\tilde{\boldsymbol{x}}^*, \Lambda)$. As each f_k is monotonically non-decreasing and L-Lipschitz continuous, we have

$$\sum_{k=1}^{K} \lambda_k f_k(\tilde{x}_k^*) - \sum_{k=1}^{K} \lambda_k f_k(x_k) \le \sum_{k:\tilde{x}_k^* \ge x_k} \lambda_k L |\tilde{x}_k^* - x_k|$$

$$\leq CL \sum_{k:\tilde{x}_k^* \geq x_k} |\tilde{x}_k^* - x_k|$$
$$\leq CL\|\tilde{\boldsymbol{x}}^* - \boldsymbol{x}\|_1$$
$$< C(L+1)\Delta(\tilde{\boldsymbol{x}}^*, \Lambda). \tag{B.3}$$

It follows from (B.3) that

$$\sum_{k=1}^{K} \lambda_k f_k(x_k) - C(L+1)\Delta(\boldsymbol{x}, \Lambda) > \sum_{k=1}^{K} \lambda_k f_k(\tilde{x}_k^*) - C(L+1)\Delta(\tilde{x}^*, \Lambda),$$

which contradicts that $\tilde{\boldsymbol{x}}^*$ is optimal for $\tilde{\mathcal{P}}$. Therefore, $\tilde{\boldsymbol{x}}^* \in \Lambda$, which implies that $\Delta(\tilde{\boldsymbol{x}}^*, \Lambda) = 0$. It follows that $F(\tilde{\boldsymbol{x}}^*) = \sum_{k=1}^{K} \lambda_k f_k(\tilde{x}_k^*)$. We now claim that $\tilde{\boldsymbol{x}}^*$ is also optimal for $\mathcal{P}$. Indeed, if not, then there exists $\boldsymbol{x} \in \Lambda$ such that $\sum_{k=1}^{K} \lambda_k f_k(x_k) > \sum_{k=1}^{K} \lambda_k f_k(\tilde{x}_k^*)$. As $\Delta(\boldsymbol{x}, \Lambda)$, this implies that $F(\boldsymbol{x}) > F(\tilde{\boldsymbol{x}}^*)$, which again contradicts that $\tilde{\boldsymbol{x}}^*$ is optimal for $\tilde{\mathcal{P}}$.

On the other hand, as we have established that any optimal solution $\tilde{\boldsymbol{x}}^*$ to $\tilde{\mathcal{P}}$ must satisfy $\Delta(\tilde{\boldsymbol{x}}^*, \Lambda) = 0$, which means that it is also feasible for $\mathcal{P}$. Therefore, an optimal solution to $\mathcal{P}$ must also be optimal for $\tilde{\mathcal{P}}$. We thereby establish the equivalence between optimal solutions of $\tilde{\mathcal{P}}$ and $\mathcal{P}$. □

B.4. Proof of Lemma 4

Proof. We need to show that

$$UB_{\boldsymbol{x}} - LB_{\boldsymbol{x}} = \sum_{k=1}^{K} [\lambda_k^U f_k^U(x_k) - \lambda_k^L f_k^L(x_k)] + C(L+1)\left(\Delta(\boldsymbol{x}, \boldsymbol{\lambda}^U, \boldsymbol{\mu}^L) - \Delta(\boldsymbol{x}, \boldsymbol{\lambda}^L, \boldsymbol{\mu}^U)\right) \leq D\gamma.$$

For the first component, we have

$$\begin{aligned}
&\lambda_k^U f_k^U(x_k) - \lambda_k^L f_k^L(x_k) \\
=&(\bar{\lambda}_k + \frac{\gamma}{2})(\bar{f}_k(x_k) + \frac{\gamma}{2}) - (\bar{\lambda}_k - \frac{\gamma}{2})(\bar{f}_k(x_k) - \frac{\gamma}{2}) \\
=&\gamma(\bar{f}_k(x_k) + \bar{\lambda}_k + \frac{\gamma}{4}) \leq 3C\gamma,
\end{aligned}$$

where in the last inequality we have used that all the observations lie in the interval $[0, C]$ and $\gamma \leq C$, which is an implicit upper bound on γ since we have restricted the lower and upper estimates to be in $[0, C]$.

For the second component, for fixed $\boldsymbol{\mu}$, consider $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K)$ and $\boldsymbol{\lambda}' = (\lambda_1', \ldots, \lambda_K')$ where $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$ only differ at the kth component with $\lambda_k' > \lambda_k$. Then for any $\boldsymbol{x}$ in the capacity region corresponding to $\boldsymbol{\lambda}$, we construct $\boldsymbol{x}'$ such that $\boldsymbol{x}'$ only differ with $\boldsymbol{x}$ at the kth component and $x_k' = [x_k - B(\lambda_k' - \lambda_k)/\lambda_k]^+$. From the structure of the capacity region, it is easy to see that $\boldsymbol{x}' \in \Lambda$ and $\|\boldsymbol{x} - \boldsymbol{x}'\|_1 \leq B(\lambda_k' - \lambda_k)/\lambda_k \leq B(\lambda_k' - \lambda_k)$. It follows that for fixed $\boldsymbol{x}$, $\boldsymbol{\mu}$, $\Delta(\boldsymbol{x}, \boldsymbol{\lambda}', \boldsymbol{\mu}) - \Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq B(\lambda_k' - \lambda_k)$. Using a similar argument, we can show that for fixed $\boldsymbol{\lambda}$, consider two different service rate vectors $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$ and $\boldsymbol{\mu}' = (\mu_1', \ldots, \mu_K')$ that only differ at the kth component with $\mu_k' > \mu_k$, $\Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) - \Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}') \leq B(\mu_k' - \mu_k)$. It follows that $\Delta(\boldsymbol{x}, \boldsymbol{\lambda}^U, \boldsymbol{\mu}^L) - \Delta(\boldsymbol{x}, \boldsymbol{\lambda}^L, \boldsymbol{\mu}^U) \leq (KB + M)\gamma$. Combine the two parts, we have

$$UB_{\boldsymbol{x}} - LB_{\boldsymbol{x}} \leq 3CK\gamma + C(L+1)(KB+M)\gamma = D\gamma. \quad \square$$

B.5. Proof of Lemma 5

Proof. We note that $\sum_k^K \lambda_k f_k(x_k)$ increases component-wise with $\boldsymbol{\lambda}$. Also, from the definition of Λ, for a fixed $\boldsymbol{x}$, the function $\Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ decreases component-wise with $\boldsymbol{\lambda}$ while increases component-wise with $\boldsymbol{\mu}$. Therefore, we have if for all k, $f_k(x_k) \in [f_k^L(x_k), f_k^U(x_k)]$, $\lambda_k \in [\lambda_k^L, \lambda_k^U]$ and for all m, $\mu_m \in [\mu_m^L, \mu_m^U]$,

$$\begin{aligned}
LB_{\boldsymbol{x}} &= \sum_{k=1}^{K} \lambda_k^L f_k^L(x_k) - C(L+1)\Delta(\boldsymbol{x}, \boldsymbol{\lambda}^U, \boldsymbol{\mu}^L) \\
\leq F(\boldsymbol{x}) &= \sum_{k=1}^{K} \lambda_k f_k(x_k) - C(L+1)\Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \\
\leq UB_{\boldsymbol{x}} &= \sum_{k=1}^{K} \lambda_k^U f_k^U(x_k) - C(L+1)\Delta(\boldsymbol{x}, \boldsymbol{\lambda}^L, \boldsymbol{\mu}^U). \quad \square
\end{aligned}$$

B.6. Proof of Lemma 7

Proof. For each t, let $\boldsymbol{x}'(t)$ be the projection of $\boldsymbol{x}(t)$ onto Λ with respect to l_1-norm, i.e., $\|\boldsymbol{x}(t)-\boldsymbol{x}'(t)\|_1 = \Delta(\boldsymbol{x}(t),\Lambda)$ and $\boldsymbol{x}'(t)\in\Lambda$. Since $\boldsymbol{x}'(t)$ is feasible to $\mathcal{P}$ while $\boldsymbol{x}^*$ is the optimal solution to $\mathcal{P}$, we have for each t, $\sum_{k=1}^K \lambda_k f_k(x_k^*) \geq \sum_{k=1}^K \lambda_k f_k(x_k'(t))$. Then, again, starting from (7), we have

$$\sum_{t=1}^T F(\boldsymbol{x}^*) - F(\boldsymbol{x}(t)) \tag{B.4}$$

$$= \sum_{t=1}^T \sum_{k=1}^K \lambda_k [f_k(x_k^*) - f_k(x_k(t))] + C(L+1)\sum_{t=1}^T \Delta(\boldsymbol{x}(t),\Lambda)$$

$$\geq \sum_{t=1}^T \sum_{k=1}^K \lambda_k [f_k(x_k'(t)) - f_k(x_k(t))] + C(L+1)\sum_{t=1}^T \Delta(\boldsymbol{x}(t),\Lambda)$$

$$\geq \sum_{t=1}^T \sum_{k=1}^K \lambda_k \nabla f_k(x_k'(t))[x_k'(t) - x_k(t)] + C(L+1)\sum_{t=1}^T \Delta(\boldsymbol{x}(t),\Lambda) \tag{B.5}$$

$$\geq -\sum_{t=1}^T \sum_{k=1}^K \lambda_k L |x_k'(t) - x_k(t)| + C(L+1)\sum_{t=1}^T \Delta(\boldsymbol{x}(t),\Lambda) \tag{B.6}$$

$$\geq -CL\sum_{t=1}^T \Delta(\boldsymbol{x}(t),\Lambda) + C(L+1)\sum_{t=1}^T \Delta(\boldsymbol{x}(t),\Lambda) \tag{B.7}$$

$$= C\sum_{t=1}^T \Delta(\boldsymbol{x}(t),\Lambda), \tag{B.8}$$

where in (B.5), $\nabla f_k(x_k'(t))$ is a subgradient of f_k at $x_k'(t)$ and it follows from the concavity of f_k, (B.6) follows from the Lipschitz-continuity of f_k (which implies that $|\nabla f_k(x_k'(t))| \leq L$), and (B.7) follows from the definition of $\boldsymbol{x}'$ and that $\lambda_k \leq C$. Combining (B.8) with Lemma 6, we conclude the proof of the lemma. □

B.7. Proof of inequality (9)

$$\frac{1}{2}\|\boldsymbol{Q}(t+1)\|^2 - \frac{1}{2}\|\boldsymbol{Q}(t)\|^2$$

$$= \sum_{m=1}^M \left(\left[Q_m(t) + \sum_{k=1}^K \sum_{j\in A_k(t)} \mathbb{1}_{\{s_j=s_m\}} \cdot x_j(t) - c_m(t) \right]^+ \right)^2 - Q_m(t)^2$$

$$\leq \sum_{m=1}^M \left(Q_m(t) + \sum_{k=1}^K \sum_{j\in A_k(t)} \mathbb{1}_{\{s_j=s_m\}} \cdot x_j(t) - c_m(t) \right)^2 - Q_m(t)^2$$

$$= \sum_{m=1}^M \left(Q_m(t) + \sum_{k=1}^K a_k(t)\alpha_{km}^{jsq}(t)x_k(t) - c_m(t) \right)^2 - Q_m(t)^2$$

$$\leq \sum_{m=1}^M Q_m(t)\cdot \left[\sum_{k=1}^K a_k(t)\alpha_{km}^{jsq}(t)x_k(t) - c_m(t) \right] + C_1.$$

B.8. Proof of Lemma 8

Let $\{\alpha_{km}\}$ be any other routing variables that satisfy the constraints in the definition of Λ. First, from Line 5 of **Algorithm 2**, we have that the routing component of CI-JSQ minimizes the upper bound of queue-length drift over all routing choices in the network scheduling and routing literature. That is, under the CI-JSQ policy, for all t and any routing variables $\{\alpha_{km}\}$,

$$\sum_{m=1}^M Q_m(t)\left[\sum_{k=1}^K a_k(t)\alpha_{km}^{jsq}(t)x_k(t) - c_m(t) \right] \leq \sum_{m=1}^M Q_m(t)\left[\sum_{k=1}^K a_k(t)\alpha_{km}x_k(t) - c_m(t) \right]. \tag{B.9}$$

Let $\tilde{\boldsymbol{x}}(t) \in \Lambda$ be the job-size vector such that $\|\boldsymbol{x}(t) - \tilde{\boldsymbol{x}}(t)\|_1 = \Delta(\boldsymbol{x}(t), \Lambda)$. As $\tilde{\boldsymbol{x}} \in \Lambda$, there exists $\{\alpha_{km}\}$ such that $\sum_{k=1}^{K} \lambda_k \alpha_{km} \tilde{x}_k(t) - \mu_m \leq 0$ for all m. Using (B.9), we have,

$$\begin{aligned}
&\sum_{m=1}^{M} Q_m(t) \cdot \left[\sum_{k=1}^{K} a_k(t) \alpha_{km}^{jsq}(t) x_k(t) - c_m(t) \right] + C_1 \\
\leq &\sum_{m=1}^{M} Q_m(t) \cdot \left[\sum_{k=1}^{K} a_k(t) \alpha_{km} x_k(t) - c_m(t) \right] + C_1. \\
= &\sum_{m=1}^{M} Q_m(t) \cdot \left[\sum_{k=1}^{K} a_k(t) \alpha_{km} \tilde{x}_k(t) - c_m(t) + a_k(t) \alpha_{km} (x_k(t) - \tilde{x}_k(t)) \right] + C_1 \\
= &\sum_{m=1}^{M} Q_m(t) \cdot \left[\sum_{k=1}^{K} a_k(t) \alpha_{km} \tilde{x}_k(t) - c_m(t) \right] + \sum_{m=1}^{M} Q_m(t) \cdot \left[\sum_{k=1}^{K} a_k(t) \alpha_{km} |x_k(t) - \tilde{x}_k(t)| \right] + C_1 \\
\leq &\sum_{m=1}^{M} Q_m(t) \cdot \left[C_2 \Delta(\boldsymbol{x}(t), \Lambda) + \sum_{k=1}^{K} a_k(t) \alpha_{km} \tilde{x}_k(t) - c_m(t) \right] + C_1,
\end{aligned} \tag{B.10}$$

where $C_2 = KC$ is another constant independent of T.

From (B.10), we can see that the upper bound on the drift is smaller than or equal to the product of queue length and a term of $\sum_{k=1}^{K} a_k(t) \alpha_{km} \tilde{x}_k(t) - c_m(t)$ and another involving the constraint violation $\Delta(\boldsymbol{x}(t), \Lambda)$. A cumulative upper bound on the latter term can be established from Lemma 7. Invoking Lemma 7, we have that there exists a constant (independent of T) C_3 such that $\sum_{t=1}^{T} \Delta(\boldsymbol{x}(t), \Lambda) \leq C_3 \sqrt{T} \log T$ with probability at least $1 - 1/T$. In the subsequent analysis, we can thus focus on the set of sample paths where $\sum_{t=1}^{T} \Delta(\boldsymbol{x}(t), \Lambda) \leq C_3 \sqrt{T} \log T$, as the remaining set where the condition is not satisfied has probability at most $1/T$. We next proceed to bound the former term $\sum_{k=1}^{K} a_k(t) \alpha_{km} \tilde{x}_k(t) - c_m(t)$. Let $\tilde{\Delta}(t) := \sum_{m=1}^{M} \sum_{k=1}^{K} a_k(t) \alpha_{km} \tilde{x}_k(t) - c_m(t)$. $\tilde{\Delta}(t)$ can be considered as a stochastic version of the quantity $\sum_{k=1}^{K} \lambda_k \alpha_{km} \tilde{x}_k(t) - \mu_m \leq 0$ which has been shown to be less than or equal to zero. $\{\tilde{\Delta}(t)\}_{t=1,\ldots,T}$ is a stochastic process and we use $\{\mathcal{F}_t\}$ to denote its natural filtration. Note that the job sizes $x_k(t)$ and $\tilde{x}_k(t)$ are determined by information up to $t-1$, and are thus $\mathcal{F}_{t-1}$-measurable. While $a_k(t)$'s, $c_m(t)$'s are independent of $\mathcal{F}_{t-1}$. Hence, we have that

$$\mathbb{E}[\tilde{\Delta}(t) \mid \mathcal{F}_{t-1}] = \mathbb{E}\left[\sum_{m=1}^{M} \sum_{k=1}^{K} a_k(t) \alpha_{km} \tilde{x}_k(t) - c_m(t) \mid \mathcal{F}_{t-1} \right] = \sum_{m=1}^{M} \sum_{k=1}^{K} \lambda_k \alpha_{km} \tilde{x}_k(t) - \mu_m \leq 0.$$

Also, as $a_k(t)$'s, $c_m(t)$'s, α_{km}, $\tilde{x}_k(t)$ are all bounded, $\tilde{\Delta}(t)$ is also bounded with probability 1. It follows that $S_t := \sum_{\tau=1}^{t} \tilde{\Delta}(\tau)$ is a super-martingale with bounded increment. Therefore, by Azuma–Hoeffding Inequality, we have that with probability at least $1 - 1/T$, $S_t \leq C_4 \sqrt{T} \log T$ for all $t = 1, \ldots, T$ for some constant C_4 independent of T. Thus, we can again restrict ourselves to the set of sample paths on which both $\sum_{\tau=1}^{t} \Delta(\boldsymbol{x}(\tau), \Lambda) \leq C_3 \sqrt{T} \log T$ and $\sum_{\tau=1}^{t} \tilde{\Delta}(\tau) \leq C_4 \sqrt{T} \log T$ for all $t = 1, \ldots, T$.

Appendix C. Computation aspect of CI-JSQ

In this section, we show that for given $\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}$, $\Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ can be obtained by solving the following linear program LP_Δ.

$$LP_\Delta: \max_{\{y_k, b_{km}\}} \sum_{k=1}^{K} y_k \tag{C.1}$$

$$\textbf{s.t.} \quad \sum_{m} b_{km} = \lambda_k y_k, \quad \forall k \tag{C.2}$$

$$\sum_{k} b_{km} \leq \mu_m \quad \forall m \tag{C.3}$$

$$b_{km} = 0, \quad \forall s_m \notin S_{u_k} \tag{C.4}$$

$$b_{km} \geq 0, \quad \forall k, m \tag{C.5}$$

$$0 \leq y_k \leq x_k, \quad \forall k. \tag{C.6}$$

Let $\{y_k^*\}$ (or $\boldsymbol{y}^*$) be the optimal solution to LP_Δ, the following proposition shows that we can obtain the value of $\Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ by solving LP_Δ.

Proposition 5. *For given $\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}$, $\Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \sum_{k=1}^{K} (x_k - y_k^*)$.*

Proof. Let Λ be the capacity region of the network with the network statistics being $\boldsymbol{\lambda}, \boldsymbol{\mu}$. First, if $\boldsymbol{x} \in \Lambda$, then from the definition, we see that the optimal solution to LP_Δ can be obtained by setting $y_k^* = x_k$, $b_{km}^* = \alpha_{km}\lambda_k x_k$, where $\{\alpha_{km}\}$ is a set of routing variables that makes the constraints of Λ satisfied with $\boldsymbol{x}$. It follows that $\sum_{k=1}^K (x_k - y_k^*) = \Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = 0$. If $\boldsymbol{x} \notin \Lambda$. Let $\boldsymbol{x}' \in \Lambda$ be a job-size vector such that $\|\boldsymbol{x} - \boldsymbol{x}'\|_1 = \Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$. Note that by definition of Δ, $\boldsymbol{x}'$ must satisfy $x_k' \le x_k$ for all k, otherwise we can decrease some x_k' that violates this and obtain a $\boldsymbol{x}'$ that still lies in Λ but with a smaller l_1-distance to $\boldsymbol{x}$. Hence, we have $\sum_{k=1}^K x_k' = \sum_{k=1}^K x_k - \Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$. Note that $\boldsymbol{x}' \in \Lambda$, so we can define variables $\{\alpha_{km}\}$ such that $\{\alpha_{km}'\}$, $\{x_k'\}$ satisfy the constraints of Λ. By setting $b_{km} = \alpha_{km}'\lambda_k x_k'$, we see that $\{x_k'\}$, $\{b_{km}\}$ are feasible to LP_Δ. It follows that

$$\sum_{k=1}^K (x_k - y_k^*) \le \sum_{k=1}^K (x_k - x_k') = \|\boldsymbol{x} - \boldsymbol{x}'\|_1 \Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}). \tag{C.7}$$

On the other hand, as $\{y_k^*\}$ satisfies the constraints of LP_Δ, $\boldsymbol{y}^* \in \Lambda$ (as manifested by setting $\alpha_{km} = b_{km}/\sum_m b_{km}$) and $\|\boldsymbol{x} - \boldsymbol{y}^*\|_1 = \sum_{k=1}^K (x_k - y_k^*)$. Thus, from the definition of $\boldsymbol{x}'$, we have

$$\|\boldsymbol{x} - \boldsymbol{y}^*\|_1 = \sum_{k=1}^K (x_k - y_k^*) \le \|\boldsymbol{x} - \boldsymbol{x}'\|_1 = \sum_{k=1}^K (x_k - x_k') = \Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}). \tag{C.8}$$

Combining (C.7) and (C.8), we have $\Delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \sum_{k=1}^K (x_k - y_k^*)$. □

Appendix D. Workflow of the optimization component of CI-JSQ

In this section, we present the workflow of the optimization component of the CI-JSQ for the elastic job scheduling algorithm using the one-dimensional case in Section 4.

Initially, at $t = 1$, CI-JSQ starts with the target region (of the zeroth epoch of SCBA) $[l_0, r_0] = [0, B]$. Then, it repeatedly executes the following steps:

1. Let $w_\tau = r_\tau - l_\tau$. Set $x_l := l_\tau + \frac{w_\tau}{4}$, $x_c := l_\tau + \frac{w_\tau}{2}$, $x_r := l_\tau + \frac{3w_\tau}{4}$.
2. For $i = 1, \ldots$; $\gamma_i = 1/2^i$:
   (a) Let t be the current time slot. Set job-size decisions $x(t+1) = \cdots = x(t + \lceil \sigma \log T^2/\gamma_i^2 \rceil)$ as x_l.
   (b) Feed the query observations to **Algorithm** 1 and obtain confidence interval $[LB_{x_l}, UB_{x_l}]$ for x_l.
   (c) Set job-size decisions $x(t + \lceil \sigma \log T^2/\gamma_i^2 \rceil + 1) = \cdots = x(t + 2\lceil \sigma \log T^2/\gamma_i^2 \rceil)$ as x_c.
   (d) Feed the query observations to **Algorithm** 1 and obtain confidence interval $[LB_{x_c}, UB_{x_c}]$ for x_c.
   (e) Set the job-size decisions $x(t + 2\lceil \sigma \log T^2/\gamma_i^2 \rceil + 1) = \cdots = x(t + 3\lceil \sigma \log T^2/\gamma_i^2 \rceil)$ as x_r.
   (f) Feed the query observations to **Algorithm** 1 and obtain confidence interval $[LB_{x_r}, UB_{x_r}]$ for x_r.
   (g) If $[LB_{x_l}, UB_{x_l}]$ is γ_i-separated with $[LB_{x_c}, UB_{x_c}]$ or $[LB_{x_r}, UB_{x_r}]$, eliminate $[l_\tau, x_l]$ from the target region (by setting l_τ to x_l) and proceed to the next epoch.
   (h) If $[LB_{x_r}, UB_{x_r}]$ is γ_i-separated with $[LB_{x_c}, UB_{x_c}]$ or $[LB_{x_l}, UB_{x_l}]$, eliminate $[x_r, r_\tau]$ from the target region (by setting r_τ to x_r) and proceed to the next epoch.
   (i) Otherwise, increment i and repeat step (2).

Appendix E. Gradient sampling max weight

We explain in more detail how to apply the GSMW policy in [20] to the elastic job scheduling problem. For simplicity of description of the policy, we assume that every time slot, there are at least two job arrivals of each class and utility values are observable immediately after the injection of jobs. The pseudo-code of the GSMW policy on our problem is shown in **Algorithm 4**.

The GSMW policy can be considered as a first-order primal dual algorithm that solves the optimization problem $\mathcal{P}$ associated with the elastic job scheduling problem. The primal variables are the job-sizes and GSMW uses (noisy) utility value observations to construct approximate gradient of the utility functions (Line 6). The dual variables are the queue lengths and are updated based on network dynamics. The authors in [20] analyze the regret of GSMW when the utility observations are noiseless and show that by setting $V = O(\sqrt{T}), \alpha = O(T), \delta = O(1/\sqrt{T})$, GSMW achieves $\tilde{O}(\sqrt{T})$-regret. However, when the utility observations are noisy, $\tilde{O}(\sqrt{T})$-regret is no longer achievable due to the variance of the approximate gradient $\hat{\nabla} f_k(\hat{x}_k(t))$. Following a similar analysis in [20], it can be shown that the optimal parameter regime under noisy observations is $\alpha = O(T), V = O(\sqrt{T}), \delta = O(T^{1/4})$ and GSMW achieves $\tilde{O}(T^{3/4})$-regret.

Algorithm 4 The Gradient Sampling Max-Weight Policy

Input: Parameters V, δ, α

Initialize: $\hat{x}_k(0) = \delta$.
for $t = 1, 2, \ldots, T$ **do**
  for $k = 1, \ldots, K$ **do**
    u_k injects one job of size $\hat{r}_k(t) + \delta$, another of size $\hat{r}_k(t) - \delta$, and all other jobs of size $\hat{r}_k$.
    The designate server of class k is chosen as $s_{k(t)} \in \arg\min_{s_m \in S_{u_k}} Q_m(t)$.
    Observe $\hat{f}_k(\hat{r}_k(t) + \delta)$ and $\hat{f}_k(\hat{r}_k(t) - \delta)$ and compute $\hat{\nabla} f_k(\hat{x}_k(t)) := \frac{f_k(\hat{x}_k(t)+\delta) - f_k(\hat{x}_k(t)-\delta)}{2\delta}$
  Update queue lengths according to $\boldsymbol{x}(t)$ and the network dynamics.
  for $k = 1, \ldots, K$ **do**
    $\hat{x}_k(t+1) := \mathcal{P}_{[\delta, B-\delta]}\left[\hat{x}_k(t) + \frac{1}{\alpha}(V \cdot \hat{\nabla} f_k(\hat{x}_k(t)) - Q_{s_{k(t)}}(t))\right]$

References

[1] D. Feitelson, L. Rudolph, U. Schwiegelshohn, K.C. Sevcik, P. Wong, Theory and practice in parallel job scheduling, in: Workshop on Job Scheduling Strategies for Parallel Processing 1997, pp. 1–34.
[2] B. Berg, J.P. Dorsman, M. Harchol-Balter, Towards optimality in parallel job scheduling. in Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems, 2018.
[3] D. Applegate, W. Cook, A computational study of the job-shop scheduling problem, ORSA J. Comput. 3 (2) (1991) 149–156.
[4] T. Cheng, C. Sin, A state-of-the-art review of parallel-machine scheduling research, European J. Oper. Res. 47 (3) (1990) 271–292.
[5] G.C. Buttazzo, G. Lipari, M. Caccamo, L. Abeni, Elastic scheduling for flexible workload management, IEEE Trans. Comput. 51 (3) (2002) 289–302.
[6] S.T. Maguluri, R. Srikant, L. Ying, Heavy traffic optimal resource allocation algorithms for cloud computing clusters, Perform. Eval. 81 (2014) 20–39.
[7] S.T. Maguluri, R. Srikant, L. Ying, Stochastic models of load balancing and scheduling in cloud computing clusters, in: Proceedings of IEEE Infocom 2012, pp. 702–710.
[8] Y. Zheng, B. Ji, N. Shroff, P. Sinha, Forget the deadline: Scheduling interactive applications in data centers, in: IEEE International Conference on Cloud Computing 2015, pp. 293–300.
[9] A. Wierman, M. Nuyens, Scheduling despite inexact job-size information. in: Proceedings of the ACM SIGMETRICS, 2008.
[10] S.T. Maguluri, R. Srikant, Scheduling jobs with unknown duration in clouds, IEEE/ACM Trans. Netw. 22 (6) (2013) 1938–1951.
[11] Z. Zheng, N. Shroff, Online multi-resource allocation for deadline sensitive jobs with partial values in the cloud, in: IEEE INFOCOM 2016, pp. 1–9.
[12] Y. Bao, Y. Peng, C. Wu, Z. Li, Online job scheduling in distributed machine learning clusters, in: IEEE INFOCOM 2018, pp. 495–503.
[13] Y. Peng, Y. Bao, Y. Chen, C. Wu, C. Guo, Optimus: an efficient dynamic resource scheduler for deep learning clusters, in: Proceedings of the Thirteenth EuroSys Conference 2018, pp. 1–14.
[14] A. Or, H. Zhang, M. Freedman, Resource elasticity in distributed deep learning, in: Proceedings of Machine Learning and Systems Vol. 2, 2020, pp. 400–411.
[15] A. Harlap, A. Tumanov, A. Chung, G.R. Ganger, P.B. Gibbons, Proteus: agile ml elasticity through tiered reliability in dynamic resource markets, in: Proceedings of the Twelfth European Conference on Computer Systems 2017, pp. 589-604.
[16] H. Zhang, L. Stafman, A. Or, M. Freedman, Slaq: quality-driven scheduling for distributed machine learning, in: Proceedings of the Symposium on Cloud Computing 2017, pp: 390-404.
[17] M. Jeon, S. Venkataraman, A. Phanishayee, J. Qian, W. Xiao, F. Yang, Analysis of large-scale multi-tenant GPU clusters for DNN training workloads, in: USENIX Annual Technical Conference 2019, pp. 947–960.
[18] M. Anzanello, F. Fogliatto, Learning curve models and applications: Literature review and research directions, Int. J. Ind. Ergon. 41 (5) (2011) 573–583.
[19] A. Klein, S. Falkner, J. Springenberg, F. Hutter, Learning Curve Prediction with Bayesian Neural Networks, 2016.
[20] X. Fu, E. Modiano, Learning-NUM: Network Utility Maximization with Unknown Utility Functions and Queueing Delay. in: Proceedings of ACM Mobihoc, 2021.
[21] A. Agarwal, D.P. Foster, D. Hsu, S.M. Kakade, A. Rakhlin, Stochastic convex optimization with bandit feedback, SIAM J. Optim. 23 (1) (2013) 213–240.
[22] A.D. Flaxman, A.T. Kalai, H.B. McMahan, Online convex optimization in the bandit setting: gradient descent without a gradient, 2004, arXiv preprint cs/0408007.
[23] A. Badanidiyuru, R. Kleinberg, A. Slivkins, Bandits with knapsacks, in: IEEE Annual Symposium on Foundations of Computer Science 2013, pp. 207–216.
[24] S. Agrawal, N.R. Devanur, Bandits with concave rewards and convex knapsacks, in: Proceedings of the ACM Conference on Economics and Computation 2014, pp: 989-1006.
[25] X. Cao, K.J. Liu, Online convex optimization with time-varying constraints and bandit feedback, IEEE Trans. Automat. Control 64 (7) (2017) 2665–2680.
[26] T. Liang, H. Narayanan, A. Rakhlin, On zeroth-order stochastic convex optimization via random walks, 2014, arXiv preprint arXiv:1402.2667.
[27] V. Dani, T.P. Hayes, S.M. Kakade, Stochastic Linear Optimization under Bandit Feedback, 2008.
[28] A. Nemirovsky, D. Yudin, Problem Complexity and Method Efficiency in Optimization, 1983.
[29] Y. Wang, S. Du, S. Balakrishnan, A. Singh, Stochastic zeroth-order optimization in high dimensions, in: International Conference on Artificial Intelligence and Statistics 2018, pp. 1356–1365.
[30] O. Shamir, On the complexity of bandit and derivative-free stochastic convex optimization, in: Conference on Learning Theory, 2013, pp. 3–24.
[31] H. Yu, Hao, M. Neely, X. Wei, Online convex optimization with stochastic constraints. in: Advances in Neural Information Processing Systems, 2017.
[32] T. Chen, G.B. Giannakis, Bandit convex optimization for scalable and dynamic IoT management, IEEE Internet Things J. 6 (1) (2018) 1276–1286.
[33] S. Yang, M. Mohri, Optimistic bandit convex optimization, Adv. Neural Inf. Process. Syst. (2016) 2297–2305.
[34] E. Hazan, K. Levy, Bandit convex optimization: Towards tight bounds, Adv. Neural Inf. Process. Syst. 27 (2014) 784–792.

[35] S. Agrawal, N.R. Devanur, Linear contextual bandits with knapsacks, Adv. Neural Inf. Process. Syst. (2016) 3450–3458.
[36] T. Jaksch, R. Ortner, P. Auer, Near-optimal regret bounds for reinforcement learning, J. Mach. Learn. Res. 11 (4) (2010).
[37] I. Osband, D. Russo, B. Van Roy, (More) Efficient Reinforcement Learning via Posterior Sampling. in: Advances in Neural Information Processing Systems, 2013.
[38] I. Osband, B. Van Roy, Model-based reinforcement learning and the eluder dimension, in: Proceedings of the 27th International Conference on Neural Information Processing Systems Vol. 1, 2014, pp. 1466–1474.
[39] K. Asadi, D. Misra, M. Littman, Lipschitz continuity in model-based reinforcement learning, in: International Conference on Machine Learning 2018, pp. 264–273.
[40] C. Jin, Z. Allen-Zhu, S. Bubeck, M.I. Jordan, Is Q-learning provably efficient?, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems 2018, pp. 4868–4878.
[41] S.H. Low, D.E. Lapsley. Optimization flow control. I., Basic algorithm and convergence, IEEE/ACM Trans. Netw. 7 (6) (1999) 861–874.
[42] D. Palomar, M. Chiang, Alternative distributed algorithms for network utility maximization: Framework and applications, IEEE Trans. Automat. Control 52 (12) (2007) 2254–2269.
[43] X. Lin, N.B. Shroff, Utility maximization for communication networks with multipath routing, IEEE Trans. Automat. Control 51 (5) (2006) 766–781.
[44] M. Neely, E. Modiano, C.E. Rohrs, Dynamic power allocation and routing for time varying wireless networks, in: Proceedings of IEEE INFOCOM, 2003.
[45] S. Krishnasamy, R. Sen, R. Johari, S. Shakkottai, Regret of queueing bandits, Adv. Neural Inf. Process. Syst. 29 (2016) 1669–1677.
[46] S. Krishnasamy, R. Sen, R. Johari, S. Shakkottai, Learning unknown service rates in queues: A multiarmed bandit approach, Oper. Res. 69 (1) (2021) 315–330.
[47] T. Stahlbuhk, B. Shrader, E. Modiano, Learning algorithms for minimizing queue length regret, IEEE Trans. Inf. Theory, 67 (3) 1759-1781.
[48] T. Stahlbuhk, B. Shrader, E. Modiano, Learning algorithms for minimizing queue length regret, in: Proc. IEEE International Symposium on Information Theory, 2018.