Distributed No-Regret Learning for Multi-Stage Systems with End-to-End Bandit Feedback

I-Hong Hou
ihou@tamu.edu
Department of ECE, Texas A& M University
College Station, Texas, USA

ABSTRACT

This paper studies multi-stage systems with end-to-end bandit feedback. In such systems, each job needs to go through multiple stages, each managed by a different agent, before generating an outcome. Each agent can only control its own action and learn the final outcome of the job. It has neither knowledge nor control on actions taken by agents in the next stage. The goal of this paper is to develop distributed online learning algorithms that achieve sublinear regret in adversarial environments.

The setting of this paper significantly expands the traditional multi-armed bandit problem, which considers only one agent and one stage. In addition to the exploration-exploitation dilemma in the traditional multi-armed bandit problem, we show that the consideration of multiple stages introduces a third component, education, where an agent needs to choose its actions to facilitate the learning of agents in the next stage. To solve this newly introduced exploration-exploitation-education trilemma, we propose a simple distributed online learning algorithm, $\epsilon-$EXP3. We theoretically prove that the $\epsilon-$EXP3 algorithm is a no-regret policy that achieves sublinear regret. Simulation results show that the $\epsilon-$EXP3 algorithm significantly outperforms existing no-regret online learning algorithms for the traditional multi-armed bandit problem.

CCS CONCEPTS

• **Networks** → **Network algorithms**; • **Theory of computation** → **Online learning theory**.

KEYWORDS

Online learning, multi-armed bandit, multi-hop networks, edge computing

ACM Reference Format:
I-Hong Hou. 2024. Distributed No-Regret Learning for Multi-Stage Systems with End-to-End Bandit Feedback. In *The Twenty-fifth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MOBIHOC '24), October 14–17, 2024, Athens, Greece.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3641512.3686369

MOBIHOC '24, October 14–17, 2024, Athens, Greece

ACM ISBN 979-8-4007-0521-2/24/10.
https://doi.org/10.1145/3641512.3686369

1 INTRODUCTION

In many modern applications, a job consists of multiple stages that need to be performed by different agents, and the decision made in each stage can impact the performance of the job. For example, consider a mobile edge computing application where a mobile user offloads video analytic jobs to nearby edge servers, and each edge server is equipped with multiple video analytic neural networks with different precision and latency. To process a video analytic job, the mobile user needs to first decide which edge server to forward this job to. After the mobile user forwards the job to an edge server, the edge server needs to decide which neural network to employ for this job. The performance of the job depends on the accuracy of the result and the end-to-end latency, which includes both communication and computation delays. As another example, consider packet deliveries in multi-hop networks consisting of multiple routers. Upon receiving a packet, a router needs to decide which router to forward the packet to. The performance of the packet depends on the end-to-end latency.

This paper studies the problem of designing distributed online learning algorithms under which all agents jointly learn the optimal decisions with minimum coordination, even when the outcomes of decisions are determined by an adversary. Developing such algorithms are challenging due to three major reasons. First, in most computer and network systems, it is desirable to employ distributed algorithms where each agent can only make decisions of its own action and has neither knowledge nor control on the actions taken by agents in the next stages. Second, in many systems, an agent can only observe the end-to-end outcome of the joint effects of all stages, but cannot know how actions taken in each individual stage contribute to the end-to-end outcome. Finally, an agent can only learn the outcome of its chosen action, which is typically referred to as the *bandit feedback* in the literature.

We note that the traditional multi-armed bandit problem is a special case of multi-stage systems when there is only one stage. The main challenge of the traditional multi-armed bandit problem is to balance between learning the outcomes of each possible action (exploration) and choosing the action with the best historic outcomes (exploitation). The general problem of multi-stage systems is even more challenging because agents in the next stage are also learning agents and their ability to learn depends on actions taken by the agent in the previous stage. In the example of mobile edge computing, an edge server can only process a job and learn the outcome when it receives a job from the mobile user. When a mobile user receives a poor outcome from an edge server, it may be because the edge server has yet to learn the optimal action and chooses a bad neural network, rather than because the edge server has no good options. To ensure that all edge servers can learn the optimal

actions, the mobile user needs to *educate* edge servers by forwarding a sufficient number of jobs to each of them. Thus, the mobile user is facing an exploration-exploitation-education trilemma.

To study the online learning problem in multi-stage systems, we propose an analytical model that captures both the distributed decision making and the end-to-end bandit feedback. We first consider the simplified case when each agent can observe the outcomes of all its actions, including those not taken. We show that we can achieve sublinear regret by making all agents employ the Normalized Exponential Gradient (normalized-EG) algorithm independently in a distributed fashion.

Next, we study the multi-stage system with only end-to-end bandit feedback, that is, an agent can only observe an outcome if it receives a job and it can only observe the outcome of its chosen action. To address the exploration-exploitation-education trilemma, we propose a simple distributed online learning algorithm called $\epsilon-$EXP3. The $\epsilon-$EXP3 algorithm has two operation modes, a uniform selection mode in which the agent chooses actions uniformly at random to provide equal education to agents in the next stage, and an EXP3 mode where the agent employs a variation of the EXP3 algorithm to balance the tradeoff between exploration and exploitation. By randomly alternating between these two modes, the $\epsilon-$EXP3 algorithm explicitly address all three of exploration, exploitation, and education. We theoretically prove that, when applying $\epsilon-$EXP3 on a system with L stages, the regret accumulated over T rounds is at most $O(T^{\frac{L}{L+1}}) = o(T)$.

To understand the fundamental regret lower bounds and the role of education in multi-stage systems, we study a class of time-homogeneous oracle policies. These policies assume that each node can know the outcome of all actions *before* making a decision. Therefore, there is no need to explore and each node only faces an education-exploitation dilemma. We show that the regret of these policies is at least $\Theta(T^{\frac{L-1}{L}})$, which is only slightly better than the regret of $\epsilon-$EXP3.

The utility of the $\epsilon-$EXP3 algorithm is further evaluated by simulations. The simulation results show that the regret of the $\epsilon-$EXP3 algorithm indeed scales as $O(T^{\frac{L}{L+1}})$. We also evaluate two other policies that are no-regret policies for the traditional one-stage bandit problem. Surprisingly, we show that their regrets scale as $\theta(T)$ even when the system has only two stages. The simulation results demonstrate that the education component is indeed critical in multi-stage systems.

The rest of the paper is organized as follows. Section 2 surveys existing studies on adversarial bandit problems, mobile edge computing, and multi-hop networks. Section 3 introduces our system model and problem definition. Section 4 studies the simplified case with complete one-hop feedback. Section 5 introduces and analyzes the $\epsilon-$EXP3 algorithm for systems with end-to-end bandit feedback. Section 6 establishes a regret lower bound for time-homogeneous oracle policies. Section 7 presents our simulation results. Finally, Section 8 concludes the paper.

2 RELATED WORK

No-regret bandit learning. The multi-armed bandit problem has attracted significant research interests because it elegantly captures the trade-off between exploration and exploitation. In adversarial environments, the celebrated EXP3 algorithm has been proved to achieve a regret bound of $O(T^{\frac{1}{2}})$ [5]. This bound has later been shown to be tight [3]. There have been many studies on variations and improvements of the EXP3 algorithm [4, 22, 28, 29]. All these studies only consider systems with one agent.

There have been considerable recent efforts on cooperative learning [1, 8, 16, 18, 19, 21, 23] where agents help each other find the optimal action. These studies assume that the reward of an agent only depends on the action of that agent. In contrast, our work allows different agents to have different sets of actions and considers that the reward in each round depends on the actions of all agents. Singla, Hassani, and Krause [27] has studied a distributed learning problem in two-stage systems. It is limited to the special case of two stages and requires the root node to have the ability to block feedback information.

Mobile edge computing. One emerging application of mobile edge computing is cloud/edge robotics where a robot offloads its computation tasks to nearby edge servers. An important challenge for cloud/edge robotics is that the performance of a job depends on both the quality of the outcome and the end-to-end delay. To enable flexible trade-off between quality and latency, Jiang et al. [15] has proposed a controller that dynamically select the suitable neural network configuration. Wu et al. [30] has modeled the problem of adaptive configuration as an integer programming problem and proposed a heuristic for it. He et al. [13] has employed a reinforcement learning approach for adaptive configuration. Zhang et al. [32] has employed Lyapunov optimization to learn the optimal configuration over time. These studies only study the decisions of edge servers and they only consider stationary systems. Chinchali et al. [9] has proposed using deep reinforcement learning for the offloading decisions of robots, but it does not consider the adaptive configuration of edge servers. To the best of our knowledge, no existing work has jointly optimized the offloading decision of robots and the adaptive configuration of edge servers in unknown and time-varying environments.

Multi-hop networks. There have been significant interests in employing online learning or reinforcement learning techniques for multi-hop networks, but few of them have been able to characterize end-to-end delay and enforcing end-to-end deadline. Bhorkar and Javidi [6] has proposed a no-regret learning policy for minimizing end-to-end transmission cost. Park, Kang, and Joo [24] has proposed a UCB-based algorithm for throughput-optimality in multi-hop wireless networks. Al Islam et al. [2] has considered the problem of end-to-end congestion control problem in multi-hop networks as a multi-armed bandit problem. Zhang, Tang, and Wang [31] has studied the problem of relay selection to minimize energy consumption in two-hop networks. None of the aforementioned studies consider end-to-end delay or end-to-end deadline.

Mao, Koksal, and Shroff [20], Deng, Zhao, and Hou [10], and Gu, Liu, Shen [11] have all studied online scheduling and routing algorithms for multi-hop networks with end-to-end deadlines, but they require precise knowledge on the capacity and latency of each link. HasanzadeZonuzy, Kalathil, and Shakkottai [12] has proposed a model-based reinforcement learning algorithm for real-time multi-hop networks but it only works for stationary systems. Both Lin and van der Schaar [17] and Shiang, and van der Schaar [26] employ

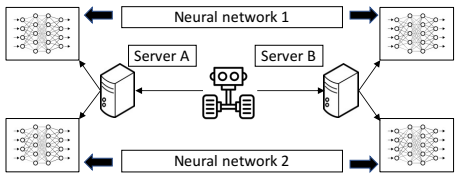
reinforcement learning to serve delay-sensitive traffic by modeling multi-hop networks as stationary MDPs with unknown kernels. To the best of our knowledge, no existing work has studied the regret of delay-sensitive multi-hop networks in adversarial environments.

3 SYSTEM MODEL

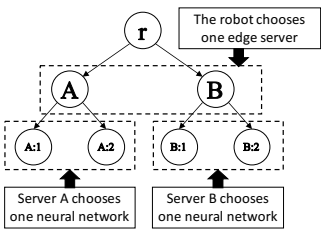
We represent a multi-stage system as a tree with depth $L+1$. We denote the root node by r and the set of leaf nodes by $\mathcal{L}$. We use C_i to denote the set of children of a non-leaf node i. In each round t, the root node r receives a job. It selects a child node $f[r,t] \in C_r$, possibly at random, and forwards the job to it. Likewise, every non-leaf node i randomly selects a child node $f[i,t] \in C_i$ and, if i receives a job in round t, forwards the job to $f[i,t]$. When the job reaches a leaf node j, it generates a cost of $c[j,t] \in [0,1]$. The value of $c[j,t]$ is revealed to all nodes between the root and the leaf node j through an end-to-end feedback message.

We note that each node only has limited feedback information in this setting. In particular, if a node receives a job in round t, then it will only know its own choice and the final cost. It has neither knowledge nor control on the choices made by its children. This is to reduce coordination overhead and to protect privacy. If a node does not receive a job in a round, then it will not receive any feedback information.

To see how our model can be used to capture mobile edge computing, we can consider the example shown in Fig. 1(a). In this system, a robot chooses one of two edge servers to offload its video analytic jobs. Each edge server has two neural networks to choose from. This system can be modeled as a tree with $L=2$ as shown in Fig. 1(b). In Fig. 1(b), the robot is the root that chooses between child A and child B. Each of these child nodes corresponds to an edge server, and each child node chooses between two leaf nodes. Each leaf node is labeled by $X:n$, where X indicates the edge server chosen by the robot, and n indicates the neural network chosen by the edge server. The cost of a leaf node is chosen to reflect the delay and the quality of the outcome of the video analytic job.



(a) System illustration



(b) Tree model

Figure 1: A mobile edge computing system and its tree model

This model can also be used to capture multi-hop networks. In multi-hop networks, the root is the source that generates packets to be delivered. Each non-leaf node corresponds to the path used to transfer a packet to an intermediate router. The choice of that non-leaf node corresponds to choosing the next-hop by the intermediate router. Each leaf node is a complete path from the source to the destination and its cost can be chosen to reflect end-to-end delay. We note that we do not require the topology of the multi-hop networks to be trees. Even when the topology of a network is not a tree, the set of all loop-free paths from the source to the destination can still be represented as a tree.

Each non-leaf node i employs a distributed online policy that determines the probability of forwarding a job to a child node j in round t, denoted by $x[i,j,t] := Prob(f[i,t]=j)$, in the event that i receives a job. We have $x[i,j,t] \geq 0$ and $\sum_{j\in C_i} x[i,j,t] = 1$. Node i needs to determine the values of $x[i,j,t]$ using only the information available up to round $t-1$.

We now characterize the performance of a distributed online policy after it determines the values of $x[i,j,t]$ and selects $f[i,t]$ accordingly. Let $y[i,t]$ be the random variable indicating the amount of cost that would be incurred if node i receives a job in round t, under the probability distribution of $f[i,t]$. By definition, we have $y[j,t] = c[j,t]$ for each leaf node $j \in \mathcal{L}$. For each non-leaf node i, $y[i,t]$ can be calculated recursively through $y[i,t] = y[f[i,t],t]$. Also, let $w[i,t] := E\Big[y[i,t]\Big|\mathcal{H}_{t-1}\Big]$ be the conditional expected amount of cost incurred if node i receives a job in round t, given all events up to round $t-1$, denoted by $\mathcal{H}_{t-1}$. The value of $w[i,t]$ can then be calculated recursively by $w[j,t] = c[j,t]$ for each leaf node j and $w[i,t] = \sum_{j\in C_i} x[i,j,t]w[j,t]$ for each non-leaf node i. The total expected cost incurred by the distributed online policy over a time horizon of T rounds can then be written as $\sum_{t=1}^{T} E\Big[y[r,t]\Big] = \sum_{t=1}^{T} E\Big[w[r,t]\Big]$.

We compare the cost of a distributed online policy against a stationary policy where each node selects the same child node in each round t, i.e., $f[i,t] \equiv f_i, \forall t$. Under a stationary policy, all jobs will reach the same leaf node j^*, and hence the total cost incurred by the stationary policy is $\sum_{t=1}^{T} c[j^*,t]$. The optimal stationary policy is the one that has a minimum cost among all stationary policies and its cost is $\min_{j\in\mathcal{L}} \sum_{t=1}^{T} c[j,t]$. We therefore define the regret of a distributed online policy as $\sum_{t=1}^{T} E\Big[y[r,t]\Big] - \min_{j\in\mathcal{L}} \sum_{t=1}^{T} c[j,t]$. Our goal is to design a *no-regret* policy whose regret is sublinear in T under all possible vectors of $c[j,t]$:

Definition 1. *A distributed online policy is said to be a no-regret policy if* $\sum_{t=1}^{T} E\Big[y[r,t]\Big] - \min_{j\in\mathcal{L}} \sum_{t=1}^{T} c[j,t] = o(T)$.

4 PRELIMINARY: POLICY WITH COMPLETE ONE-HOP FEEDBACK

In this section, we first study the simplified case where each non-leaf node has complete one-hop feedback from its children. Specifically, each non-leaf node i, regardless whether it receives a job or not, will be able to learn the values of $y[j,t]$ for each of its children $j \in C_i$ after i chooses $f[i,t]$. Node i can then use these values to update the values of $x[i,j,t+1]$. We emphasize that the communication overhead between a child node j and its parent node contains

only one single scalar $y[j,t]$ in each round. Hence, the feedback information that a non-leaf node has is still limited. For example, a non-leaf node has neither knowledge nor control over actions taken by its children.

We consider that each non-leaf node i independently employs the Normalized Exponential Gradient (normalized-EG) algorithm, a special case of the Online Mirror Descent algorithm and the Follow-the-Regularized-Leader algorithm. Under the normalized-EG algorithm, each non-leaf node i maintains a variable $\theta[i,j,t]$ for each $j \in C_i$ by setting $\theta[i,j,1] = 0$ and $\theta[i,j,t] = \theta[i,j,t-1] - y[j,t-1]$ for all $t > 1$. It then chooses $x[i,j,t] = \frac{e^{\eta_i \theta[i,j,t]}}{\sum_{k \in C_i} e^{\eta_i \theta[i,k,t]}}$ in each round t, where η_i is a constant whose value will be determined later. The normalized-EG algorithm is an online policy because $\theta[i,j,t]$ can be calculated only based on $y[j,1], y[j,2], \ldots, y[j,t-1]$. A formal description of the normalized-EG algorithm is presented in Alg. 1.

noend 1 Distributed Normalized Exponential Gradient

$\eta_i \leftarrow$ a pre-determined constant
$\theta[i,j] \leftarrow 0, \forall j \in C_i$
for each round t **do**
  $x[i,j] \leftarrow \frac{e^{\eta_i \theta[i,j]}}{\sum_{k \in C_i} e^{\eta_i \theta[i,k]}}, \forall j \in C_i$
  Select a child $f[i]$ with $Prob(f[i] = j) = x[i,j]$
  for each $j \in C_i$ **do**
    Obtain $y[j]$ from child j
    $\theta[i,j] \leftarrow \theta[i,j] - y[j]$
  $y[i] \leftarrow y[f[i]]$
  Report $y[i]$ to the parent node

The regret of the normalized-EG algorithm has been extensively studied for the special case when $L = 1$. We will further show that the normalized-EG algorithm is a no-regret policy for the general case $L > 1$. It is important to note that the values of $y[j,t]$ observed by i under the normalized-EG algorithm can be different from those under the optimal stationary policy. This is because the values of $y[j,t]$ depend on the decisions made by children nodes j. To distinguish between these two policies, we let $y_n[j,t]$ be the values of $y[j,t]$ under the normalized-EG algorithm and let $y_*[j,t]$ be those under the optimal stationary policy.

Since the normalized-EG algorithm is updated with respect to $y_n[j,t]$, we let $\mathcal{Y}_n[i,t] := \{y_n[j,\tau], \forall j \in C_i, \tau \in [1,t]\}$ be the sequences of costs of all children of i up to round t and have the following from existing studies:

Lemma 1 ([25], Theorem 2.22). *If $y_n[j,\tau] \geq 0$ for all $j \in C_i$ and $\tau \in [1,T]$, then the expected total cost incurred by i given $\mathcal{Y}_n[i]$ is upper-bounded by:*

$$\sum_{t=1}^{T} E\Big[y_n[i,t]\Big|\mathcal{Y}_n[i,t]\Big] \leq \min_{j \in C_i} \sum_{t=1}^{T} y_n[j,t] + \frac{\log|C_i|}{\eta_i} + \eta_i \sum_{t=1}^{T} \sum_{j \in C_i} x[i,j,t] y_n[j,t]^2.$$

Moreover, if $y_n[j,\tau] \in [0,1], \forall j \in C_i, \tau \in [1,T]$, then setting $\eta_i = \sqrt{\frac{\log|C_i|}{T}}$ yields:

$$\sum_{t=1}^{T} E\Big[y_n[i,t]\Big|\mathcal{Y}_n[i,t]\Big] \leq \min_{j \in C_i} \sum_{t=1}^{T} y_n[j,t] + 2\sqrt{T \log|C_i|}.$$

□

Under the optimal stationary policy, each node will choose to forward the job to the child that incurs the least cost through all T rounds. Hence, we have $\sum_{t=1}^{T} y_*[i,t] = \min_{j \in C_i} \sum_{t=1}^{T} y_*[j,t]$. We now prove that the normalized-EG algorithm is still a no-regret policy for the multi-stage system:

Theorem 1. *If each non-leaf node has as most D children, then, by setting $\eta_i = \sqrt{\frac{\log|C_i|}{T}}, \forall i$, the expected cost incurred by the root node r is upper-bounded by:*

$$\sum_{t=1}^{T} E\Big[y_n[r,t]\Big] \leq \min_{j \in \mathcal{L}} \sum_{t=1}^{T} c[j,t] + 2L\sqrt{T \log D}. \quad (1)$$

Proof. Please see the technical report [14]. □

5 POLICY WITH END-TO-END BANDIT FEEDBACK

In this section, we consider the case where each non-leaf node only has bandit feedback. Specifically, if a node does not receive a job in round t, then it will not get any feedback. If a node receives a job and forwards it to a child node $j = f[i,t]$, then it will only learn the value of $y[j,t]$. As discussed in earlier sections, online policies with end-to-end bandit feedback faces a trilemma between exploration, i.e., choosing a child to learn its cost, exploitation, i.e., choosing a child to incur low cost, and education, i.e., choosing a child so that it has a chance to learn and improve its policy.

We propose a simple distributed online learning policy to address the exploration-exploitation-education trilemma called the ϵ−EXP3 algorithm. Under the ϵ−EXP3 algorithm, each non-leaf node i maintains a variable $\theta[i,j,t]$ for each $j \in C_i$, which it will use to determine $x[i,j,t]$. When a node i sends a job to a child node $j = f[i,t]$, node i also includes a variable $v[j,t]$ indicating the probability that the child node j receives a job in round t. Since a node j will receive a job if its parent node i receives a job and node i chooses j, the value of $v[j,t]$ can be calculated by $v[j,t] = v[i,t]x[i,j,t]$.

We now discuss how a non-leaf node i decides $f[i,t]$ in each round t. There are two modes for choosing $f[i,t]$ and node i randomly decides which mode to operate in in each t. Each node i is assigned two pre-determined constants ϵ_i and η_i. With probability ϵ_i, node i is in the *uniform selection mode* and it chooses $f[i,t]$ uniformly at random from its children, that is, $Prob(f[i,t] = j) = 1/|C_i|, \forall j \in C_i$. With probability $1 - \epsilon_i$, node i is in the *EXP3 mode* and it chooses $f[i,t] = j$ with probability $\frac{e^{\eta_i \theta[i,j,t]}}{\sum_{k \in C_i} e^{\eta_i \theta[i,k,t]}}$. We use $m[i,t] \in \{U, E\}$ to denote the mode of node i, where U is the uniform selection mode and E is the EXP3 mode. Combining these two modes and we have $x[i,j,t] = \epsilon_i \frac{1}{|C_i|} + (1 - \epsilon_i) \frac{e^{\eta_i \theta[i,j,t]}}{\sum_{k \in C_i} e^{\eta_i \theta[i,k,t]}}$.

After choosing $f[i,t]$ for each node i, we can set $y_\epsilon[i,t] = c[i,t]$ for each leaf node and set $y_\epsilon[i,t] = y_\epsilon[f[i,t],t]$ for each non-leaf node, where the subscript ϵ is to highlight that this corresponds

to the values of $y[j,t]$ under the $\epsilon-$EXP3 algorithm. We note that, even if node i does not receive a job in round t, the value of $y_\epsilon[i,t]$ is still well-defined, but node i does not know its value.

Finally, we discuss how node i determines $\theta[i,j,t]$. Node i initializes $\theta[i,j,1] = 0$ for all children j. If node i receives a job in round t, then it learns the value of $y_\epsilon[f[i,t],t]$. Node i sets $z[f[i,t],t] = \frac{y_\epsilon[f[i,t],t]|C_i|}{v[i,t]}$, if $m[i,t] = U$, and sets $z[f[i,t],t] = \frac{y_\epsilon[f[i,t],t]\sum_{k\in C_i} e^{\eta_i\theta[i,k,t]}}{v[i,t]e^{\eta_i\theta[i,j,t]}}$, if $m[i,t] = E$. Node i sets $z[j,t] = 0$ for all $j \neq f[i,t]$. On the other hand, if node i does not receive a job in round t, then it sets $z[j,t] = 0, \forall j \in C_i$. Finally, it sets $\theta[i,j,t+1] = \theta[i,j,t] - z[j,t], \forall j \in C_i$.

Alg. 2 describes the $\epsilon-$EXP3 algorithm in detail, where we streamline some of the steps for easier implementation.

noend 2 ϵ-EXP3

$\eta_i, \epsilon_i \leftarrow$ pre-determined constants
$\theta[i,j] \leftarrow 0, \forall j \in C_i$
for each round t **do**
  if Node i receives a job and $v[i,t]$ from its parent **then**
    $x[i,j] \leftarrow \epsilon_i \frac{1}{|C_i|} + (1-\epsilon_i)\frac{e^{\eta_i\theta[i,j]}}{\sum_{k\in C_i} e^{\eta_i\theta[i,k]}}, \forall j \in C_i$
    $v[j,t] \leftarrow v[i,t]x[i,j], \forall j \in C_i$
    Randomly select $m[i] \in \{U,E\}$ with $Prob(m[i] = U) = \epsilon_i$
    if $m[i] = U$ **then**
      Select a child $f[i] \in C_i$ uniformly at random
      Forward the job and $v[f[i],t]$ to child $f[i]$ and obtain $y_\epsilon[f[i]]$ from $f[i]$
      $\theta[i,f[i]] \leftarrow \theta[i,f[i]] - \frac{y_\epsilon[f[i]]|C_i|}{v[i,t]}$
      Return $y_\epsilon[i] \leftarrow y_\epsilon[f[i]]$ to the parent
    else
      Select a child $f[i] \in C_i$ with $Prob(f[i] = j) = \frac{e^{\eta_i\theta[i,j]}}{\sum_{k\in C_i} e^{\eta_i\theta[i,k]}}$
      Forward the job and $v[f[i],t]$ to child $f[i]$ and obtain $y_\epsilon[f[i]]$ from $f[i]$
      $\theta[i,f[i]] \leftarrow \theta[i,f[i]] - \frac{y_\epsilon[f[i]]\sum_{k\in C_i} e^{\eta_i\theta[i,k]}}{v[i,t]e^{\eta_i\theta[i,j]}}$
      Return $y_\epsilon[i] \leftarrow y_\epsilon[f[i]]$ to the parent

Remark 1. *The reason that the $\epsilon-$EXP3 algorithm has two different modes to choose $f[i,t]$ is to address the exploration-exploitation-education trilemma. When node i is in the uniform selection mode, its goal is to provide equal education to all its children. Hence, it selects $f[i,t]$ uniformly at random so that each child node has the same chance of receiving a job and learning from its outcome. When node i is in the EXP3 mode, its goal is to balance the trade-off between exploration and exploitation. Hence, it employs a very similar way of choosing $f[i,t]$ as the EXP3 algorithm. The value of ϵ_i determines the portion of time that node i dedicate to education. On the other hand, the value of η_i determines the trade-off between exploration and exploitation when node i is in the EXP3 mode, where larger η_i means more emphasis on exploitation. The values of ϵ_i and η_i will be determined later.*

We now analyze the regret of $\epsilon-$EXP3. Our first step is to establish some properties of $z[j,t]$. We let $\mathcal{Y}_\epsilon[i,t] := \{y_\epsilon[j,\tau], \forall j \in C_i, \tau \in [1,t]\}$ be the sequences of costs of all children of i up to round t and let $\mathcal{Z}[i,t] := \{z[j,\tau], \forall j \in C_i, \tau \in [1,t]\}$ be all the values of $z[j,\tau]$ that has been observed by i up to round t. We then have the following:

Lemma 2. *For any non-leaf node i,*

$$E\Big[z[j,t]\Big|\mathcal{Y}_\epsilon[i,t], \mathcal{Z}[i,t-1]\Big] = y_\epsilon[j,t], \tag{2}$$

and

$$\begin{aligned}&E\Big[z[j,t]^2\Big|\mathcal{Y}_\epsilon[i,t], \mathcal{Z}[i,t-1]\Big]\\ =&\Big(\epsilon_i|C_i| + (1-\epsilon_i)\frac{\sum_{k\in C_i} e^{\eta_i\theta[i,k,t]}}{e^{\eta_i\theta[i,j,t]}}\Big)\frac{y_\epsilon[j,t]^2}{v[i,t]}.\end{aligned} \tag{3}$$

Proof. Please see the technical report [14]. □

Next, we show that, if node i is in the EXP3 mode at round t, then its expected cost is the same as the expected cost of running the normalized-EG algorithm against the sequence $z[j,t]$.

Lemma 3. *By considering a sequence $y_n[j,\tau] = z[j,\tau], \forall j \in C_i, \tau \in [1,T]$ for the normalized-EG algorithm,*

$$\begin{aligned}&E\Big[y_\epsilon[i,t]\Big|m[i,t] = E, \mathcal{Y}_\epsilon[i,t], \mathcal{Z}[i,t-1]\Big]\\ =&E\Big[E\Big[y_n[i,t]\Big|\mathcal{Y}_n[i,t] = \mathcal{Z}[i,t]\Big]\Big],\end{aligned}$$

where the outer expectation on the right hand side is taken with respect to $z[j,t]$.

Proof. Please see the technical report [14]. □

Our next step is to bound the difference between $\sum_{t=1}^T E\Big[y_\epsilon[i,t]\Big]$ and $\min_{j\in C_i}\sum_{t=1}^T y_\epsilon[j,t]$ under any given given sequence of $y_\epsilon[j,1],\ldots,y_\epsilon[j,T]$, for all $j \in C_i$.

Lemma 4. *If each non-leaf node has at most D children, then*

$$\begin{aligned}&\sum_{t=1}^T E\Big[y_\epsilon[i,t]\Big|\mathcal{Y}_\epsilon[i,t]\Big]\\ \leq&\min_{j\in C_i}\sum_{t=1}^T y_\epsilon[j,t] + \epsilon_i T + \frac{\log D}{\eta_i} + \eta_i\sum_{t=1}^T\frac{D}{v[i,t]},\end{aligned}$$

for all non-leaf node i. Moreover, if the depth of the tree is $L+1$, then setting $\eta_i = T^{-\frac{L}{L+1}}$ for all i and setting ϵ_i to be 0 if $C_i \subset \mathcal{L}$, and $DT^{-\frac{1}{L+1}}$ otherwise yields

$$\begin{aligned}&\sum_{t=1}^T E\Big[y_\epsilon[i,t]\Big|\mathcal{Y}_\epsilon[i,t]\Big] - \min_{j\in C_i}\sum_{t=1}^T y_\epsilon[j,t]\\ \leq&\begin{cases}(D+\log D)T^{\frac{L}{L+1}}, & \textbf{\textit{if}}\ C_i \subset \mathcal{L},\\ (2D+\log D)T^{\frac{L}{L+1}} & \textbf{\textit{else}}.\end{cases}\end{aligned}$$

Proof. Please see the technical report [14]. □

Remark 2. *An explanation for the choice of ϵ_i is in order. We set $\epsilon_i = 0$ if all children of node i are leaf nodes. Since leaf nodes do not have any children to choose from, they have nothing to learn and do not need education. Hence, node i can operate exclusively in the EXP3 mode. On the other hand, if node i has some children that are non-leaf*

nodes, then node i needs to educate these children. Hence, it operates in the uniform selection mode with a constant probability.

We will now prove that the ϵ–EXP3 policy is a no-regret policy.

Theorem 2. *If the depth of the tree is $L+1$ and each non-leaf node i has at most D children, then, by using the same settings of η_i and ϵ_i as in Lemma 4, the regret of ϵ-EXP3 is at most $((2L-1)D + L\log D)T^{\frac{L}{L+1}} = o(T)$.*

Proof. We will prove the theorem by establishing the following statement: If a node i is $(L-h)$-hops from the root node r, then $\sum_{t=1}^{T} E\Big[y_n[i,t]\Big] \le \sum_{t=1}^{T} y_*[i,t] + ((2h-1)D + h\log D)T^{\frac{L}{L+1}}$, where $y_*[i,t]$ is the cost under the optimal stationary policy.

We prove the statement by induction. First, consider the case $h=1$, that is, the node i is $(L-1)$-hops from r. Since the tree has depth $L+1$, either i is a leaf node or all children of i are leaf nodes. If i is a leaf node, then $y_n[i,t] = y_*[i,t] = c[i,t] \in [0,1]$ and the statement holds. If all children of i are leaf nodes, then we have $y_n[j,t] = y_*[j,t] = c[j,t]$ for all $j \in C_i$. Hence, by Lemma 4,

$$\begin{aligned}\sum_{t=1}^{T} E\Big[y_\epsilon[i,t]\Big] &= \sum_{t=1}^{T} E\Big[y_\epsilon[i,t]\Big|\mathcal{Y}_\epsilon[i,t]\Big]\\ &\le \min_{j\in C_i}\sum_{t=1}^{T} y_\epsilon[j,t] + (D+\log D)T^{\frac{L}{L+1}}\\ &= \sum_{t=1}^{T} y_*[i,t] + (D+\log D)T^{\frac{L}{L+1}},\end{aligned}$$

and the statement holds.

We now assume that the statement holds when $h=g$ and consider a node i that is $(L-(g+1))$-hops from r. Either i is a leaf node or all children of i are $(L-g)$-hops from r. If i is a leaf node, then the statement clearly holds. If i is not a leaf node, then, by the induction hypothesis, we have $\sum_{t=1}^{T} E\Big[y_\epsilon[j,t]\Big] \le \sum_{t=1}^{T} y_*[j,t] + ((2g-1)D + g\log D)T^{\frac{L}{L+1}}$, for all $j \in C_i$. We can then use Lemma 4 to establish the following:

$$\begin{aligned}\sum_{t=1}^{T} E\Big[y_\epsilon[i,t]\Big] &= \sum_{t=1}^{T} E\Big[E\Big[y_\epsilon[i,t]\Big|\mathcal{Y}_\epsilon[i,t]\Big]\Big]\\ &\le E\Big[\min_{j\in C_i}\sum_{t=1}^{T} y_\epsilon[j,t]\Big] + (2D+\log D)T^{\frac{L}{L+1}}\\ &\le \min_{j\in C_i}\sum_{t=1}^{T} y_*[j,t] + ((2g-1)D + g\log D)T^{\frac{L}{L+1}}\\ &\quad + (2D+\log D)T^{\frac{L}{L+1}}\\ &= \sum_{t=1}^{T} y_*[i,t] + ((2g+1)D + (g+1)\log D)T^{\frac{L}{L+1}},\end{aligned}$$

and the statement holds. By induction, the statement holds for all h.

Since the root node r is 0-hop from itself and $\sum_{t=1}^{T} y_*[r,t] = \min_{j\in\mathcal{L}}\sum_{t=1}^{T} c_{j,t}$, the theorem holds. □

Finally, we note that the ϵ–EXP3 algorithm requires the knowledge of T to set ϵ_i and η_i. When T is not known in advance, we can employ the doubling trick to design an anytime algorithm as shown in Algorithm 3. This anytime algorithm is also a no-regret policy:

noend 3 Anytime ϵ-EXP3

1: **for** m = 0, 1, 2, … **do**
2: Set ϵ_i and η_i according to Theorem 2, but replace T with 2^m
3: Run Algorithm 2 on the 2^m rounds $t = 2^m, 2^m+1, \ldots, 2^{m+1}-1$

Theorem 3. *The regret of Algorithm 3 is at most $\frac{2^{\frac{2L}{L+1}}}{2^{\frac{L}{L+1}}-1}((2L-1)D + L\log D)T^{\frac{L}{L+1}}$.*

Proof. The proof is very similar to that in [25, Section 2.3.1], and is hence omitted. □

6 REGRET LOWER BOUND AND THE NEED FOR EDUCATION

In this section, we establish a regret lower bound of $\Omega(T^{\frac{L-1}{L}})$ for a class of *time-homogeneous oracle policies*. Under this class of policies, each node knows the outcomes of each non-leaf child, $y[i,t]$, *before* selecting a child to forward a job to. Since the outcomes of each non-leaf child is known in advance, there is no need for exploration and each node only faces an education-exploitation dilemma. As we establish a regret lower bound for this class of policies, we also establish the need for education.

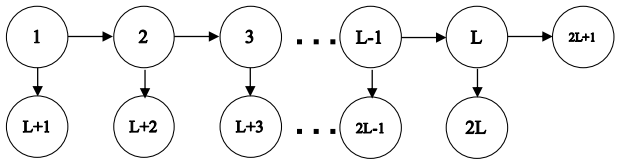


Figure 2: System illustration for establishing a lower bound

We consider a system with depth $L+1$ as shown in Fig. 2. There are L non-leaf nodes, numbered as $1, 2, \ldots, L$, and $L+1$ leaf nodes, numbered as $L+1, L+2, \ldots, 2L+1$. Each non-leaf node i has two children. For each $i \le L-1$, one child of node i is the leaf node $L+i$ and the other child is the non-leaf node $i+1$. For node L, both children, node $2L$ and node $2L+1$, are leaf nodes. When a leaf node j received a job, it generates a cost of 1 with probability p_j and a cost of 0 with probability $1-p_j$. Given a small positive constant $\delta < 1/2^L$, we set one of p_{2L} and p_{2L+1} to be $(1-2^L\delta)/2$ and the other to be $(1+2^L\delta)/2$, and then set $p_j = (1-(2^L - 2^{j-L-1})\delta)/2$ for all the other leaf nodes $j = L+1, L+2, \ldots, 2L-1$. Hence, we have $(1-2^L\delta)/2 < p_{L+1} < p_{L+2} < \cdots < p_{2L-1} < (1-2^L\delta)/2$ and $\min_{j\in\mathcal{L}}\sum_{t=1}^{T} E\big[c[j,t]\big] = (1-2^L\delta)T/2$. The regret of the system is

$$\sum_{t=1}^{T} E\big[y[1,t]\big] - (1-2^L\delta)T/2 = \sum_{t=1}^{T}\Big(E\big[y[1,t]\big] - (1-2^L\delta)/2\Big). \quad (4)$$

We now discuss the policies employed by each non-leaf node. Since both children of node L are leaf nodes, node L does not need

to consider education. We consider that node L can run an arbitrary online learning algorithm with bandit feedback. For all other non-leaf nodes $i = 1, 2, \ldots, L-1$, we assume that they employ a time-homogeneous oracle policy defined as follows:

Definition 2. *Let i_1 and i_2 be the two children of node i, then a time-homogeneous oracle policy is one that chooses a child to forward a job to at time t with the following assumptions:*

- *A1: Node i can obtain the expected cost of each child, $E\big[y[i_1,t]\big]$ and $E\big[y[i_2,t]\big]$, before making the forwarding decision.*
- *A2: Node i makes its forwarding decision solely based on $E\big[y[i_1,t]\big] - E\big[y[i_2,t]\big]$. Specifically, let $\zeta := \Big|E\big[y[i_1,t]\big] - E\big[y[i_2,t]\big]\Big|$, then node i will forward the job to the child with the higher expected cost with probability $P_i(\zeta)$, and to the other child with probability $1 - P_i(\zeta)$, where $P_i(\cdot)$ is an arbitrary decreasing function chosen by node i.*

We note that A1 provides a node with much more information than is possible in multi-stage systems with bandit feedback, where a node can only obtain the cost of a child *if* it forwards a job to the child, and only *after* it makes the forwarding decision. Thus, intuitively, the regret of policies with A1 serves as a natural lower bound for the regret of policies with end-to-end bandit feedback. The purpose of A2 is to highlight that a node i only knows the expected costs, but not the internal variables of its children.

We also note that policies with A1 do not need to explore, since it knows the expected costs of all children in advance. Hence, policies with A1 only face an education-exploitation dilemma. The only reason that a policy may select a child with a higher expected cost, by choosing $P_i(\eta) > 0$, is to educate its children.

We first establish a bound for the expected cost of node L, whose children are both leaf nodes. Let $N_L(t)$ be the number of times that node L has received a job from its parent at time t. Since node L can only learn the costs of its children when it receives a job, node L cannot determine which of its two children has the smaller p_j when $N_L(t)$ is small. The following lemma formalizes this intuition.

Lemma 5. *There exists a positive integer N_δ such that, for all t with $N_L(t) < N_\delta$, $E\big[y[L,t]\big] > (1 - (2^L - 2^{L-1})\delta)/2$.*

Proof. This is a direct result of Lemma 3.6 in [7]. □

We now establish a regret lower bound for the system in Fig. 2.

Theorem 4. *For the system in Fig. 2, the regret is $\Omega(T^{\frac{L-1}{L}})$ for any bandit learning policy employed by node L and any time-homogeneous oracle policies employed by nodes $1, 2, \ldots, L-1$.*

Proof. Let T_δ be the time at which $N_L(t) = N_\delta$. By Lemma 5, $E\big[y[L,t]\big] > (1 - (2^L - 2^{L-1})\delta)/2$ for any $t < T_\delta$.

We first study the system behavior before time T_δ. Consider the forwarding decision of node $L-1$ at any time $t < T_\delta$. Node $L-1$ has two children. One is the leaf node $2L-1$ with $E\big[y[2L-1,t]\big] = p_{2L-1} = (1 - (2^L - 2^{L-2})\delta)/2$. The other is the non-leaf node L with $E\big[y[L,t]\big] > (1 - (2^L - 2^{L-1})\delta)/2 = p_{2L-1} + 2^{L-3}\delta$. By A2, the probability that node $L-1$ selects node L is at most $q_{L-1} := P_{L-1}(2^{L-3}\delta)$. We also have $E\big[y[L-1,t]\big] \geq p_{2L-1} = (1 - (2^L - 2^{L-2})\delta)/2$.

We further analyze the forwarding decision of node $i < L-1$ at any time $t < T_\delta$. Using a simple induction argument, it can be shown that the probability that node i selects node $i+1$ is at most $q_i := P_i(2^{i-2}\delta)$, and $E\big[y[i,t]\big] \geq p_{L+i} = (1 - (2^L - 2^{i-1})\delta)/2$. Therefore, at any time $t < T_\delta$, we have

$$E\big[y[1,t]\big] - (1 - 2^L\delta)/2 \geq \delta/2. \tag{5}$$

Moreover, since node L can only receive a job if, for each $i \leq L-1$, node i selects node $i+1$, which happens with probability at most q_i, we have

$$E\big[T_\delta\big] \geq \frac{N_\delta}{\prod_{i=1}^{L-1} q_i}. \tag{6}$$

Next, we analyze the system behavior after time T_δ. For any time $t > T_\delta$, $E\big[y[L,t]\big] \geq \min\{p_{2L}, p_{2L+1}\} = (1 - 2^L\delta)/2$. Consider the forwarding decision of node $L-1$. Since $E\big[y[2L-1,t]\big] = p_{2L-1} = (1 - (2^L - 2^{L-2})\delta)/2 \leq E\big[y[L,t]\big] + 2^{L-3}\delta$, the probability that node $L-1$ selects node L is at most $1 - P_L(2^{L-3}\delta) = 1 - q_{L-1}$. Using a simple induction argument, we can further show that the probability that node i selects node $i+1$ is at most $1 - q_i$, for all $i \leq L-1$. Hence, the probability that node L receives a job is at most $\prod_{i=1}^{L-1}(1-q_i)$. If node L does not receive a job, which happens with probability at least $1 - \prod_{i=1}^{L-1}(1-q_i)$, then the expected cost is at least $\min_{j\in\{L+1,L+2,\ldots,2L-1\}} p_j = (1 - (2^L - 1)\delta)/2$. We then have, at any time $t > T_\delta$,

$$E\big[y[1,t]\big] - (1 - 2^L\delta)/2 \geq \big(1 - \prod_{i=1}^{L-1}(1-q_i)\big)\delta/2. \tag{7}$$

Combining Eq. (5), (6), and (7) and we have the following regret bound

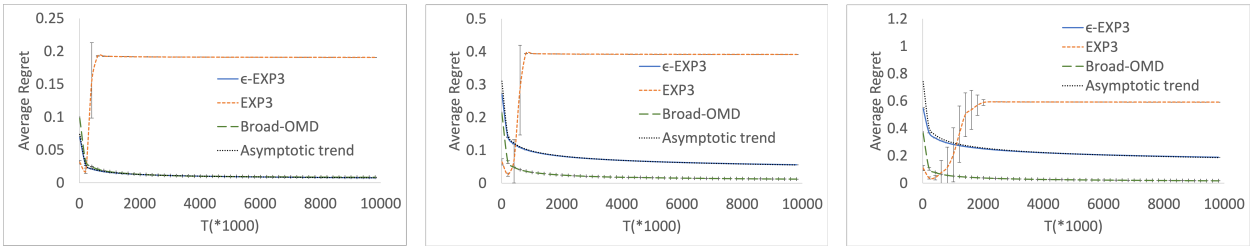
$$\begin{aligned}
&\sum_{t=1}^{T}\Big(E\big[y[1,t]\big] - (1 - 2^L\delta)/2\Big)\\
\geq& E\Big[\sum_{t=1}^{T_\delta}\delta/2 + \sum_{t=T_\delta+1}^{T}\big(1 - \prod_{i=1}^{L-1}(1-q_i)\big)\delta/2\Big]\\
\geq& \frac{\delta}{2}\Big[\frac{N_\delta}{\prod_{i=1}^{L-1} q_i} + \big(T - \frac{N_\delta}{\prod_{i=1}^{L-1} q_i}\big)\big(1 - \prod_{i=1}^{L-1}(1-q_i)\big)\Big].
\end{aligned} \tag{8}$$

It is then straightforward to show that $\frac{\delta}{2}\big[\frac{N_\delta}{\prod_{i=1}^{L-1} q_i} + (T - \frac{N_\delta}{\prod_{i=1}^{L-1} q_i})\big(1 - \prod_{i=1}^{L-1}(1-q_i)\big)\big] = \Omega(T^{\frac{L-1}{L}})$. Moreover, setting $q_i = \Theta(T^{-\frac{1}{L}})$, for all $i \leq L-1$, makes $\frac{\delta}{2}\big[\frac{N_\delta}{\prod_{i=1}^{L-1} q_i} + (T - \frac{N_\delta}{\prod_{i=1}^{L-1} q_i})\big(1 - \prod_{i=1}^{L-1}(1-q_i)\big)\big] = \Theta(T^{\frac{L-1}{L}})$. □

Before closing the section, we note that the lower-bound analysis in this section is limited to time-homogeneous policies. We make this assumption to explicitly prevent a parent node from using history to imply internal variables of its children. Extending our analysis to time-varying policies will be interesting future work.

7 SIMULATION RESULTS

We present our simulation results in this section. We simulate two different scenarios. The first scenario is based on trees whose leaf nodes generate Bernoulli costs. While this scenario is artificially constructed and may not correspond to real-world applications, its



(a) $D = 2, L = 2, p_{min} = 0.2$ (b) $D = 2, L = 3, p_{min} = 0.4$ (c) $D = 2, L = 4, p_{min} = 0.6$

(d) $D = 4, L = 2, p_{min} = 0.2$ (e) $D = 4, L = 3, p_{min} = 0.4$ (f) $D = 4, L = 4, p_{min} = 0.6$

Figure 3: Time-average regrets under various system parameters

simulation results provide important insights on how online algorithms behave in distributed multi-stage systems. The second scenario is based on mobile edge computing. We compare our ϵ−EXP3, with parameters from Lemma 4, against the standard EXP3 algorithm, where each node runs the EXP3 algorithm independently from each other, and the Broad-OMD algorithm [29]. Both EXP3 and Broad-OMD are no-regret policies for the special case when $L = 1$.

7.1 Trees with Bernoulli Costs

We consider systems that can be represented as trees with depth L+1. Each non-leaf node has D children. Each leaf node j is associated with a parameter $p_j \in [0, 1]$. Whenever a leaf node j receives a job, its cost $c[j, t]$ is 1 with probability p_j and 0 with probability $1 - p_j$. The system is run over T rounds. Initially, the values of p_j is chosen so that $\max_j p_j = 1$ and $\min_j p_j = p_{min}$. At round $t = T/100$, the leaf with $p_j = 1$ has its value of p_j changed into 0. Fig. 4 illustrates an example. For a given set of parameters D, L, T, and $[p_j]$, we simulate the system for 20 independent runs and calculate the time-average regret $\left(\sum_{t=1}^{T} y[r, t] - \min_{j \in \mathcal{L}} \sum_{t=1}^{T} c[j, t]\right)/T$ under all evaluated policies.

Simulation results are shown in Fig. 3, with the error bars indicating standard deviations. It can be observed that the time-average regret of ϵ−EXP3 approaches 0 over time in all cases. We note that the convergence rate of ϵ−EXP3 becomes much slower as L becomes larger. This is consistent with Theorem 2, which shows that the time-average regret scales as $O(1/\sqrt[L+1]{T})$. To verify that the time-average regret of ϵ−EXP3 scales as $O(1/\sqrt[L+1]{T})$, we also plot the *asymptotic trend* in Fig. 3. The value of the asymptotic trend for a particular T is calculated as $R_{D,L}/\sqrt[L+1]{T}$, where $R_{D,L}$ is chosen so that the value of the asymptotic trend and the time-average regret

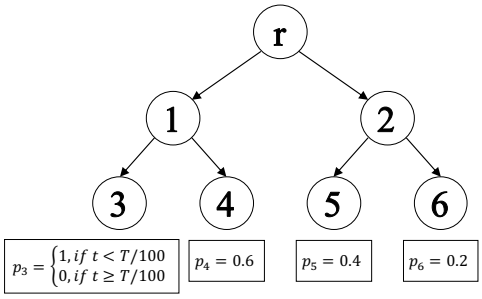
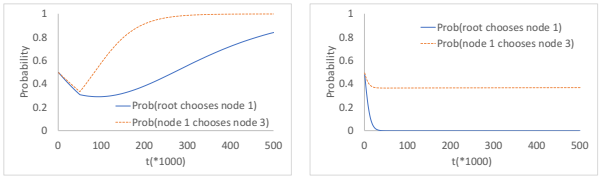


Figure 4: A system with $D = 2, L = 2$ and $p_{min} = 0.2$

of ϵ−EXP3 are the same when $T = 5 \times 10^6$, that is, at the mid-point of the x-axis in the figures. It can be observed that ϵ−EXP3 is close to the asymptotic trend. This demonstrates that the time-average regret ϵ−EXP3 indeed scales as $O(1/\sqrt[L+1]{T})$.

On the other hand, it can also be observed that the time-average regrets of both EXP3 and Broad-OMD converge to p_{min} in all settings in Fig. 3. This result shows that neither of them is a no-regret policy in multi-stage systems. To understand why the standard EXP3 algorithm is not a no-regret policy, consider the system illustrated in Fig. 4. Before round $t = \frac{T}{100}$, the optimal strategy for node 1 is to forward the job to node 4 with $p_4 = 0.6$. The optimal strategy for the root is to forward the job to node 2, who then forwards the job to node 6 with $p_6 = 0.2$. Hence, at round $t = \frac{T}{100}$ and under the EXP3 algorithm, the root will choose node 2 with a high probability and node 1 will choose node 4 with a high probability. Now, consider the first time after round $\frac{T}{100}$ when the root forwards a job to node 1. Since node 1 is unaware that p_3 has become 0, it chooses node 4 with a high probability and will likely incur a high cost. This high cost will cause the root to exponentially reduce



(a) The behavior of $\epsilon-$EXP3 (b) The behavior of EXP3

Figure 5: Transient behaviors of the system in Fig. 4 with $T = 5 \times 10^6$.

the probability of choosing node 1 in the future, making it even harder for node 1 to explore and learn the fact that p_3 has become 0. This is why the EXP3 algorithm suffers from a time-average regret of roughly $p_6 = 0.2$. In contrast, our $\epsilon-$EXP3 algorithm ensures that the root always chooses node 1 with at least a constant probability in each round. This persistent education enables node 1 to eventually discover that p_3 has become 0.

To demonstrate the behavior discussed in the above paragraph, we conduct a simulation to show the transient behaviors of the two algorithms. Specifically, we test the system shown in Fig. 4 with $T = 5 \times 10^6$. The value of p_3 is initially 1, and becomes 0 at round 5×10^4. For each algorithm, we record the probability that the root r would choose node 1 and the probability that node 1 would choose node 3. Simulation results are shown in Fig. 5, where each data point represent the average of the previous 1000 rounds. Under the EXP3 algorithm, the probability that the root would choose node 1 at round 5×10^4 is less than 0.05%. Since node 1 rarely receives any jobs, it cannot improve its performance, which, in turn, makes the root even less likely to choose node 1. At round 5×10^5, probability that the root would choose node 1 has become less than 0.02%. In contrast, the $\epsilon-$EXP3 algorithm offers persistent education to node 1. Hence, after round 5×10^4, node 1 quickly finds that p_3 has improved and increases its probability of choosing node 3. As a result, the root also starts increasing its probability of choosing node 1 after round 10^5.

7.2 Mobile Edge Computing

We consider a mobile edge computing system. In this system, there is a mobile robot that generates video analytic jobs for real-time processing. The robot is connected to D edge servers with different communication media. To process a job, each edge server has D different neural networks to choose from. Different neural networks have different precision and different processing time. There is also a communication latency of each link. The delay of transmitting over a link is an exponential function with mean $\frac{1}{\lambda}$. Some links have a constant λ while other links have a λ that increases over time. This models the time-varying congestion on these links. Fig. 6(a) illustrates the system when $D = 2$.

The robot requires a strict deadline of one time unit for each job. If the end-to-end latency, that is, the sum of communication latency and processing time, exceeds one time unit, then a deadline violation occurs and the cost is one. If the end-to-end latency is less than one time unit, then the cost is the miss rate of the employed neural network.

We have conducted 20 independent runs for each T. Simulation results are shown in Fig. 6. It can be observed that the $\epsilon-$EXP3 algorithm significantly outperforms the EXP3 algorithm when T is sufficiently large. The Broad-OMD algorithm has similar performance as $\epsilon-$EXP3 when $D = 2$, but is much worse than $\epsilon-$EXP3 when $D = 3$.

7.3 Multi-hop Networks

We consider multi-hop networks as illustrated in Fig. 7(a). In this system, the source (node S) is sending packets to the destination (node D) through a number of inter-connected relay nodes. Upon receiving a packet, a node needs to decide which node to forward the packet to. The delay of transmitting over a link is an exponential function with mean $\frac{1}{\lambda}$. Some links have a constant λ while other links have a λ that increases over time. We consider that the source requires a strict end-to-end deadline guarantee of one unit time. If the end-to-end delay of a packet is more than one unit time, then a deadline violation occurs.

Let L be the number of relay nodes that a packet needs to visit before reaching the destination. We have tested this system for different values of L. Simulation results are shown in Fig. 7. It can be observed that the $\epsilon-$EXP3 algorithm is either optimal or near-optimal in all settings.

8 CONCLUSION

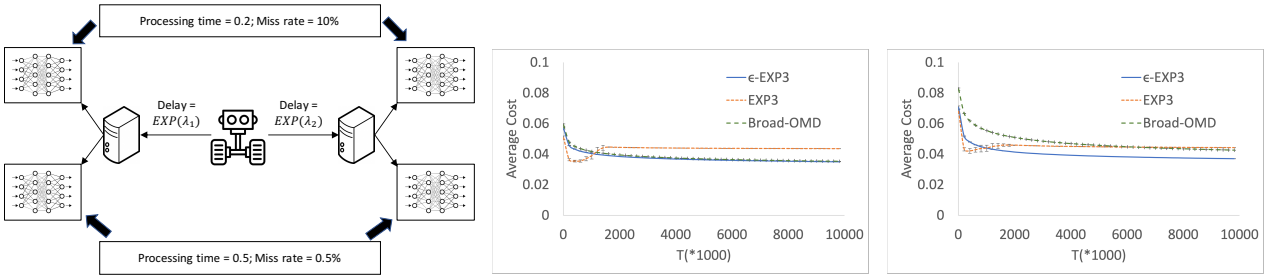
In this paper, we study multi-stage systems with end-to-end bandit feedback. The fundamental challenge of learning the optimal policy of agents in each stage is a newly introduced exploration-exploitation-education trilemma. We propose a simple distribute policy, the $\epsilon-$EXP3 algorithm, that explicitly addresses this trilemma. Moreover, we theoretically prove that the $\epsilon-$EXP3 algorithm is a no-regret policy. Simulation results show that the $\epsilon-$EXP3 algorithm significantly outperforms existing policies.

9 ACKNOWLEDGEMENT

This material is based upon work supported in part by NSF under Award Numbers ECCS-2127721 and CCF-2332800 and in part by the U.S. Army Research Laboratory and the U.S. Army Research Office under Grant Number W911NF-22-1-0151.

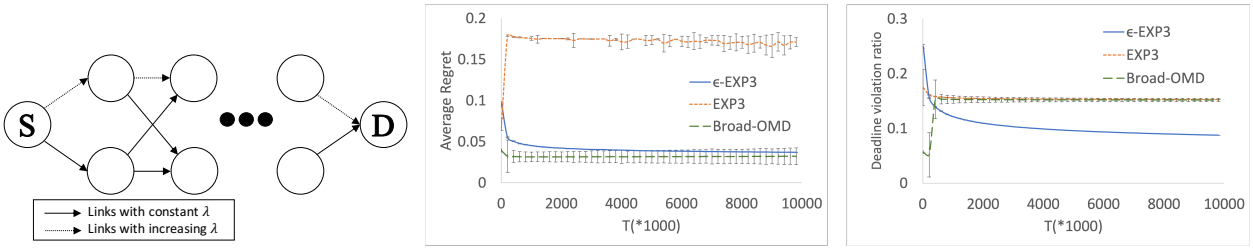
REFERENCES

[1] Mridul Agarwal, Vaneet Aggarwal, and Kamyar Azizzadenesheli. 2022. Multi-agent multi-armed bandits with limited communication. *The Journal of Machine Learning Research* 23, 1 (2022), 9529–9552.
[2] ABM Alim Al Islam, SM Iftekharul Alam, Vijay Raghunathan, and Saurabh Bagchi. 2012. Multi-armed bandit congestion control in multi-hop infrastructure wireless mesh networks. In *2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, 31–40.
[3] Jean-Yves Audibert and Sébastien Bubeck. 2009. Minimax Policies for Adversarial and Stochastic Bandits.. In *COLT*, Vol. 7. 1–122.
[4] Jean-Yves Audibert and Sébastien Bubeck. 2010. Regret bounds and minimax policies under partial monitoring. *The Journal of Machine Learning Research* 11 (2010), 2785–2836.
[5] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002. The nonstochastic multiarmed bandit problem. *SIAM journal on computing* 32, 1 (2002), 48–77.
[6] AA Bhorkar and T Javidi. 2010. No regret routing for ad-hoc wireless networks. In *2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers*. IEEE, 676–680.
[7] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning* 5, 1 (2012), 1–122.



(a) System illustration when $D = 2$ (b) Results for $D = 2$ (c) Results for $D = 3$

Figure 6: Setting and result of a mobile edge computing system



(a) System illustration (b) Results for $L = 2$ (c) Results for $L = 3$

Figure 7: Setting and result of multi-hop networks

[8] Nicolò Cesa-Bianchi, Tommaso Cesari, and Claire Monteleoni. 2020. Cooperative online learning: Keeping your neighbors updated. In *Algorithmic learning theory*. PMLR, 234–250.

[9] Sandeep Chinchali, Apoorva Sharma, James Harrison, Amine Elhafsi, Daniel Kang, Evgenya Pergament, Eyal Cidon, Sachin Katti, and Marco Pavone. 2021. Network offloading policies for cloud robotics: a learning-based approach. *Autonomous Robots* 45, 7 (2021), 997–1012.

[10] Han Deng, Tao Zhao, and I-Hong Hou. 2019. Online routing and scheduling with capacity redundancy for timely delivery guarantees in multihop networks. *IEEE/ACM Transactions on Networking* 27, 3 (2019), 1258–1271.

[11] Yan Gu, Bo Liu, and Xiaojun Shen. 2021. Asymptotically Optimal Online Scheduling With Arbitrary Hard Deadlines in Multi-Hop Communication Networks. *IEEE/ACM Transactions on Networking* 29, 4 (2021), 1452–1466.

[12] Aria HasanzadeZonuzy, Dileep Kalathil, and Srinivas Shakkottai. 2020. Reinforcement learning for multi-hop scheduling and routing of real-time flows. In *2020 18th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*. IEEE, 1–8.

[13] Zhaoliang He, Yuan Wang, Chen Tang, Zhi Wang, Wenwu Zhu, Chenyang Guo, and Zhibo Chen. 2022. AdaConfigure: Reinforcement Learning-Based Adaptive Configuration for Video Analytics Services. In *International Conference on Multimedia Modeling*. Springer, 245–257.

[14] I-Hong Hou. 2024. Distributed No-Regret Learning for Multi-Stage Systems with End-to-End Bandit Feedback. arXiv:2404.04509 [cs.LG]

[15] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 253–266.

[16] Peter Landgren, Vaibhav Srivastava, and Naomi Ehrich Leonard. 2021. Distributed cooperative decision making in multi-agent multi-armed bandits. *Automatica* 125 (2021), 109445.

[17] Zhichu Lin and Mihaela van der Schaar. 2010. Autonomic and distributed joint routing and power control for delay-sensitive applications in multi-hop wireless networks. *IEEE Transactions on Wireless Communications* 10, 1 (2010), 102–113.

[18] Keqin Liu and Qing Zhao. 2010. Distributed learning in multi-armed bandit with multiple players. *IEEE transactions on signal processing* 58, 11 (2010), 5667–5681.

[19] Udari Madhushani, Abhimanyu Dubey, Naomi Leonard, and Alex Pentland. 2021. One more step towards reality: Cooperative bandits with imperfect communication. *Advances in Neural Information Processing Systems* 34 (2021), 7813–7824.

[20] Zhoujia Mao, Can Emre Koksal, and Ness B Shroff. 2014. Optimal online scheduling with arbitrary hard deadlines in multihop communication networks. *IEEE/ACM Transactions on Networking* 24, 1 (2014), 177–189.

[21] David Martínez-Rubio, Varun Kanade, and Patrick Rebeschini. 2019. Decentralized cooperative stochastic bandits. *Advances in Neural Information Processing Systems* 32 (2019).

[22] Gergely Neu. 2015. Explore no more: Improved high-probability regret bounds for non-stochastic bandits. *Advances in Neural Information Processing Systems* 28 (2015).

[23] Conor Newton, Ayalvadi Ganesh, and Henry Reeve. 2022. Asymptotic Optimality for Decentralised Bandits. *ACM SIGMETRICS Performance Evaluation Review* 49, 2 (2022), 51–53.

[24] Daehyun Park, Sunjung Kang, and Changhee Joo. 2021. A learning-based distributed algorithm for scheduling in multi-hop wireless networks. *Journal of Communications and Networks* 24, 1 (2021), 99–110.

[25] Shai Shalev-Shwartz. 2012. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning* 4, 2 (2012), 107–194.

[26] Hsien-Po Shiang and Mihaela van der Schaar. 2010. Online learning in autonomic multi-hop wireless networks for transmitting mission-critical applications. *IEEE Journal on Selected Areas in Communications* 28, 5 (2010), 728–741.

[27] Adish Singla, Hamed Hassani, and Andreas Krause. 2018. Learning to interact with learning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[28] Taishi Uchiya, Atsuyoshi Nakamura, and Mineichi Kudo. 2010. Algorithms for adversarial bandit problems with multiple plays. In *International Conference on Algorithmic Learning Theory*. Springer, 375–389.

[29] Chen-Yu Wei and Haipeng Luo. 2018. More adaptive algorithms for adversarial bandits. In *Conference On Learning Theory*. PMLR, 1263–1291.

[30] Kun Wu, Yibo Jin, Weiwei Miao, Zeng Zeng, Zhuzhong Qian, Jingmian Wang, Mingxian Zhou, and Tuo Cao. 2021. Soudain: Online Adaptive Profile Configuration for Real-time Video Analytics. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. IEEE, 1–10.

[31] Jian Zhang, Jian Tang, and Feng Wang. 2020. Cooperative relay selection for load balancing with mobility in hierarchical WSNs: A multi-armed bandit approach. *IEEE Access* 8 (2020), 18110–18122.

[32] Sheng Zhang, Can Wang, Yibo Jin, Jie Wu, Zhuzhong Qian, Mingjun Xiao, and Sanglu Lu. 2021. Adaptive Configuration Selection and Bandwidth Allocation for Edge-Based Video Analytics. *IEEE/ACM Transactions on Networking* 30, 1 (2021), 285–298.