Multi-Agent Bandit Learning through Heterogeneous Action Erasure Channels

Osama Hanna* UCLA
Merve Karakas* UCLA
Lin F. Yang UCLA
Christina Fragouli UCLA

Abstract

Multi-Armed Bandit (MAB) systems are witnessing an upswing in applications within multi-agent distributed environments, leading to the advancement of collaborative MAB algorithms. In such settings, communication between agents executing actions and the primary learner making decisions can hinder the learning process. A prevalent challenge in distributed learning is action erasure, often induced by communication delays and/or channel noise. This results in agents possibly not receiving the intended action from the learner, subsequently leading to misguided feedback. In this paper, we introduce novel algorithms that enable learners to interact concurrently with distributed agents across heterogeneous action erasure channels with different action erasure probabilities. We illustrate that, in contrast to existing bandit algorithms, which experience linear regret, our algorithms assure sub-linear regret guarantees. Our proposed solutions are founded on a meticulously crafted repetition protocol and scheduling of learning across heterogeneous channels. To our knowledge, these are the first algorithms capable of effectively learning through heterogeneous action erasure channels. We substantiate the superior performance of our algorithm through numerical experiments, emphasizing their practical significance in addressing issues related to communication constraints and delays in multi-agent environments.

* indicates equal contribution. Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

1 INTRODUCTION

Multi-armed bandits, a well-established and effective online learning model, are increasingly finding applications in multi-agent distributed environments. One notable use-case involves leveraging a central learner, with actions (arms) communicated to remote agents to collect rewards, as discussed in Hanna et al. (2022b,a, 2023b). However, a noteworthy gap in the existing literature pertains to scenarios where the communicated actions may be lost due to communication channel issues such as delays or noise interference. This challenge becomes even more pronounced when various agents possess communication channels with varying capabilities, and these channels do not provide feedback regarding the receipt of actions. Throughout this paper, we will refer to "feedback" to denote receipt acknowledgments from the channel, distinguishing this from the rewards, which represent feedback on the learned actions.

This challenge has not been well explored in the literature, as most works assume that agents will acknowledge whether an action request has been received or not. Yet the assumption of feedback availability can have high cost or simply not be possible. For instance, distributed recommendation systems may send content (action requests) over wireless channels that are notoriously subject to delays due to varying channel conditions and lost packets(Kurose and Ross, 2012); even wired networks are subject to significant delay variability due to factors such as network topology, queuing delay and prioritization within cloud databases (Yeh et al., 2014; Dehghan et al., 2019; Kurose and Ross, 2012). Meanwhile, even if the content is delivered and displayed to the agents (e.g., an app recommending to follow a route, visit a restaurant, etc.), we cannot be sure when exactly a human user sees it, if at all.

Another motivating case is when the agents are devices with very limited communication capabilities. One such application is fleets of medical micro-robots (which today can be even of nanometre-scale) that pro-

pel themselves through biological media, such as the veins and the gastrointestinal tract (Liu et al., 2023; Zou et al., 2022; Amir et al., 2020). Multi-agent MAB algorithms can facilitate personalizing the robot's actions to different patients, for instance, to release tailored amounts of substances or to attack specific particles. The rewards (capturing action outcomes) are usually observed through external medical equipment, such as ultrasound or other imaging; however, conveying what action to play to the robots, can be communication challenging. A third case is that of military operations, where a central commander may want to communicate actions to agents (such as small robots), who do not wish to communicate back so as not to reveal their position in a hostile territory, yet their actions may have impact observable through satellite imaging or sensors.

In this work, we dispense with the need for feedback. We ask, what performance can we achieve if the learner action requests are delivered according to a known probabilistic model, but we have no additional information on whether each specific request is delivered or not. In particular, we assume operation over T rounds, where at each round a central learner sends commands (which action to play) to M agents through erasure channels with erasure probability ϵ_i, $i = 1 \dots M$, where these probabilities can be arbitrarily different across channels. This induces a Geometric distribution on the reception time of each action request, different for each agent[1]. The agents send no feedback (thus the learner does not know which action request the agents are following); the agents play at each round the last action command they received. The learner observes the reward for the (uknown) action played through an error-free channel - which can lead to erroneous action-reward associations. Indeed, energy and space limitations of micro-robots, security constraints in military applications, or simply the structure of the communication protocol can prevent transmitting feedback back to the learner; and agents are required to perform an action at each time, since even no action (for instance staying still) is also an action, see Table 1 for a small example.

Our objective is to design multi-agent MAB schemes that minimize the impact of action erasures on the regret, while also leveraging the multi-agent setting to expedite the learning process. Our main contributions are as follows:

- We propose BatchSP2 , a Successive Arm Elimination based repetition algorithm with a crafted scheduling part for multi-agent MAB setup with erasures, and prove sub-linear regret guarantees on the proposed algorithm.
- We provide numerical comparisons with a number of baseline algorithms, show the superiority of our algorithm to the benchmarks and that simply applying existing MAB algorithms in a manner oblivious to action erasures can lead to linear regret.

[1]We note that our schemes extend to more general such probability distributions; see Section 3.1 for a detailed discussion.

Related Work. Various MAB algorithms achieving optimal or near-optimal regret bounds under different assumptions have been proposed over the years (Lattimore and Szepesvári, 2020). Previous studies designed optimal algorithms for the simple MAB setting through providing gap dependent regret of $\tilde{O}(\sum_{a:\Delta_a>0} \frac{1}{\Delta_a})$ and worst-case regret of $O(\sqrt{KT \log T})$ (Thompson, 1933; Auer et al., 2002; Lai, 1987). However, these algorithms are not resilient to action erasures and they are not optimized for multi-agent settings. Building upon MAB algorithms, there has been a considerable amount of recent work on multi-agent MABs in various settings (Shahrampour et al., 2017; Dubey et al., 2020; Agarwal et al., 2022; Xu and Klabjan, 2023). However, these works predominantly consider connected agents with some form of communication between neighbors or improve the regret bounds through feedback mechanisms, primarily involving collision sensing for agents (Wang et al., 2020; Shi et al., 2021). Consequently, they fail to effectively handle action erasures, particularly when connections between agents and feedback mechanism are missing.

Taking a step back, our work fits within the framework of heterogeneous distributed agents supporting central learning. The heterogeneity in our setup comes from the diversity in the communication channels (different erasure probabilities). Although as we discuss next several works have considered heterogeneous setups, communication channel diversity and how it can affect MAB learning is we believe a natural setup that has not been widely explored.

Multi-armed bandits with delayed feedback has been studied in recent years under different settings due to its practical applications. For the stochastic setting, Joulani et al. (2013) shows that bounded unknown i.i.d. delays cause an additive increase in the regret, i.e., $O(\sqrt{KT \log T} + K\mathbb{E}[D])$ where the first term is the regret of stochastic MAB problem with no delays at round T, K is the number of actions, and $\mathbb{E}[D]$ is the expected delay. Following the work in Joulani et al. (2013), Mandel et al. (2015) proposes a queue-based MAB algorithm to handle delays. Later, Pike-Burke et al. (2018) achieves the same additive increase

in regret as in Joulani et al. (2013) under delayed aggregated anonymous feedback. Vernade et al. (2017) studies Bernoulli bandits with known delay distribution where some feedback could also be censored, i.e., do not reach the learner. Relevant to these works, Grover et al. (2018) proposes an algorithm for the best arm identification problem in stochastic MABs with partial and delayed feedback where the aim is to minimize the number of samples for identifying the best action. They extend their methods to the parallel MAB setting, i.e., multiple actions are pulled at each time; however, they provide no lower bounds on the sample complexity of their problem setting. While these works incorporate delays into the stochastic MAB model, delays are associated with action pulls whereas, in our setup, delays are associated with agents and are independent on the pulled action for the same agent.

A recent work considers the single agent action erasure channel (Hanna et al., 2023a) and they provide a generic repetition scheme that works on top of any MAB algorithm and gets regret at most $O(1/\sqrt{1-\epsilon})$ away, and a specific algorithm that gets $O(\sqrt{KT} + K/\sqrt{1-\epsilon})$ regret that is near optimal; our model accepts their work as a special case; however, extending their methods to our case is highly non-trivial, as the main challenge being the variability between erasure probabilities of each channel, that induces a need for careful scheduling across agents.

Paper Organization In Section 2, we introduce the notation and system model; we explain the proposed algorithm in Section 3; analyze it in Section 4 and provide upper bounds; evaluate and compare with possible baselines in Section 5; and conclude in Section 6.

2 PROBLEM FORMULATION

2.1 Multi-armed Bandits

We consider a stochastic multi-armed bandit problem in which a learner plays an action $a_t \in \mathcal{A}$ at each round t from the set of possible actions $\mathcal{A}$ and receives a reward r_t associated with the played action. This interaction is repeated over a horizon T, i.e., $t \in \{1, 2, ..., T\}$ and the learner aims to maximize the cumulative reward at the end of T rounds. The set of possible actions $\mathcal{A}$ are the same throughout the horizon and have K elements, i.e., $|\mathcal{A}| = K$. The decision of the learner on which action to play may depend on the history $\mathcal{H}_t = \{a_1, r_1, a_2, r_2, ..., a_{t-1}, r_{t-1}\}$. Additionally, in a stochastic setting, the reward for each action a is generated from an unknown reward distribution with an unknown mean μ_a. In our analysis, we assume that the rewards are in the interval $[0, 1]$; however, our results directly extend to sub-Gaussian distributions. The objective of the learner is minimize the regret over a time horizon T defined as

$$R_T = T \max_{a \in \mathcal{A}} \mu_a - \mathbb{E}\left[\sum_{t=1}^{T} r_t\right]$$

2.2 Multi-Agent Multi-Armed Bandits with Action Erasures

Consider a central learner connected to M distributed agents, indexed by $[M]$, over heterogeneous erasure channels. The learner faces a stochastic K-armed bandit problem, i.e., $|\mathcal{A}| = K$. At each round t during a time horizon T, the learner selects an action $a_t^{(m)} \in \mathcal{A}$ for each agent $m \in [M]$ to play. That is, M actions are played per round (unlike the traditional setting described above). When an action is chosen for agent m, it is communicated through an i.i.d. action erasure channel characterized by erasure probability ϵ_m, and may or may not be received. That is, independently from other rounds and agents, each agent m receives $a_t^{(m)}$ with probability $1 - \epsilon_m$ and does not receive an action with probability ϵ_m. The learner does not know which action requests get erased, but has knowledge of upper bounds on the erasure probabilities. The agents, on the other hand, perceive their own erasures but they do not have a feedback mechanism to inform the learner, i.e., there is no uplink between the agents and the learner. Furthermore, as motivated from applications discussed in Section 1, we assume that the agents cannot (or do not wish to) run the algorithm themselves and and continues to play the same action (last successfully received action), denoted as $\tilde{a}_t^{(m)} \in \mathcal{A}$ to play in the case of an erasure. $\tilde{a}_0^{(m)} \in \mathcal{A}$ denotes the action performed by the agent m if the action in first round is erased, it is chosen uniformly at random. An example of multi-agent multi-armed bandit learning with action erasures is provided in Table 1.

Observation. Although we focus on channels with erasures, our model can also apply over action delays: an agent receives an action not at the timeslot sent, but at a later time, based on a (known) delay probability distribution, and only changes the action she plays once she receives a new action. Our algorithms naturally extend and apply to this setting as well.

Design Objective. Our objective is to formulate a distributed learning policy composed of two key elements: a decision strategy that directs the selection of actions $a_t^{(m)}$ for each agent m at each time t, and a coping mechanism for the possible mismatch between the selected actions and received rewards due to erasures. The performance metric we want to optimize is

Table 1: Example of a MA-MAB Learning Over Action Erasure Channels. At each time t the learner sends action requests to each agent; an agent that does not receive the request, simply continues to play the last received action.

	t=1	**t=2**	**t=3**	**t=4**	**t=5**	**...**
Learner	$\{a_1^{(m)}\}_{m=1}^M$	$\{a_2^{(m)}\}_{m=1}^M$	$\{a_3^{(m)}\}_{m=1}^M$	$\{a_4^{(m)}\}_{m=1}^M$	$\{a_5^{(m)}\}_{m=1}^M$	$\cdots$
Erasure ($\epsilon_1 = 0.1$)				X		$\cdots$
Agent 1 ($\tilde{a}_t^{(1)}$)	$a_1^{(1)}$	$a_2^{(1)}$	$a_3^{(1)}$	$a_3^{(1)}$	$a_5^{(1)}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Erasure ($\epsilon_M = 0.9$)	X	X		X	X	$\cdots$
Agent M ($\tilde{a}_t^{(M)}$)	$\tilde{a}_0^{(M)}$	$\tilde{a}_0^{(M)}$	$a_3^{(M)}$	$a_3^{(M)}$	$a_3^{(M)}$	$\cdots$

X denotes the erasure of the action for the given round and agent.

the total cumulative regret incurred by the policy over time T and over all M agents:

$$R_T = \sum_{m=1}^{M} \left(T \max_{a \in \mathcal{A}} \mu_a - \mathbb{E}\left[\sum_{t=1}^{T} r_t^{(m)} \right] \right)$$

We note that the cumulative regret in a perfect communication setting (no action erasures) is lower bounded by $\Omega(\sqrt{KMT})$. This bound corresponds to the optimal regret order in a centralized K-armed bandit setup, where a total of MT reward observations are centrally accessible for learning.

3 PROPOSED ALGORITHM

In this section, we introduce Batched Scheduled Persistent Pulls (BatchSP2), a Successive Arm Elimination (SAE) based multi-agent multi-armed bandit algorithm with a crafted scheduling part. The pseudocode can be found in Algorithm 1.

For the problem we consider, misinformation (associating rewards with the wrong action) can create shifts in the action means. For instance, in the erasures example in Table 1, at time $t = 3$ the learner observes the reward of the action agent M plays, but does not know whether this reward is associated with action $\tilde{a}_0^{(M)}$, $a_1^{(M)}$, $a_2^{(M)}$, or $a_3^{(M)}$. Intuitively, to minimize this shift, it is meaningful to study an algorithm where the same action pulls are repeated several times; in the example in Table 1, if we had selected $a_1^{(M)} = a_2^{(M)} = a_3^{(M)} = a$, then, we could correctly associate reward at time $t = 3$ with action a. Moreover, the fact that we need to play in parallel across M agents, implies that we need to use a batched algorithm. Accordingly, we base our proposed algorithm on SAE (Auer and Ortner, 2010), described next, with modifications that enable robustness to misinformation.

SAE is a batched algorithm, i.e., it divides the horizon into batches of exponentially increasing length and eliminates actions based on a shrinking confidence region defined by the number of pulls (Auer and Ortner, 2010). In each batch i, all remaining actions, included in a set $\mathcal{A}_i$, are pulled 4^i times, and after all pulls of the batch are completed, actions are retained if:

$$\mathcal{A}_{i+1} \leftarrow \{a \in \mathcal{A}_i | \max_{\tilde{a} \in \mathcal{A}_i} \hat{\mu}_{\tilde{a}}^{(i)} - \hat{\mu}_a^{(i)} \leq 4\sqrt{\log(KT)/2 \cdot 4^i}\}$$

where $\hat{\mu}_a^{(i)}$ indicates the empirical mean of the reward of action a in batch i.

Note that applying SAE directly in our setup does not perform well, due to two issues that need attention: (1) it may eliminate the best arm in early batches due to wrong feedback resulting in linear regret, (2) allocate an unnecessary amount of resource to bad channels. We have to modify SAE to address these two issues, as otherwise, as Examples 1 and 2 later in this section illustrate, we may accrue large regret.

Addressing the first issue is straightforward: the learner simply repeats each action[2] until the probability the correct action has been successfully received by the agent is sufficiently high. Only after this point the learner starts associating rewards with actions, thus minimizing the probability of misinformation. More specifically, if the learner decides to receive p number of rewards for an action a through agent m, the learner first asks the agent m to repeat the action $\alpha_m = \lceil 4 \log T / \log(1/\epsilon_m) \rceil - 1$ additional times to ensure a success probability of at least $1 - 1/poly(T)$. A total of $\alpha_m + p$ rewards are generated in the environment, but only the last p are taken into account by the learner to update the mean estimate of action a. This ensures with high probability that the rewards considered (effective rewards) are generated from the distribution associated with the selected action. We

[2] A similar scheme was proposed in Hanna et al. (2023a) for the case of a single agent system.

note however that all $\alpha_m + p$ rewards generated are counted in our regret, and thus, large α_m values can affect the regret values we get, as we will also see in Section 4.

Addressing the second issue, scheduling how action pulls are allocated across agents, is significantly more challenging. One issue is that, we need to wait for all pulls of batch i to finish (agents that finish their tasks earlier will simply play random actions, potentially accumulating regret) before starting the next batch. Thus, the total regret we will accrue at batch i, is mainly determined[3] by $T^{(i)}$, the time at which all 4^i pulls of the actions in $\mathcal{A}_i$ are completed; and $T^{(i)}$ highly depends on the schedule, as simple examples can illustrate.

The following examples illustrate that two (natural to consider) scheduling algorithms (one playing all 4^i pulls of an action at only one agent, and the other splitting the pulls of each action across all agents) can lead to larger than needed $T^{(i)}$ and thus suboptimal regret.

Example 1 Assume that we order the agents so that $\alpha_1 \leq \alpha_2 \leq \dots \alpha_M$ (where α_m is the number of repetitions in each channel to ensure high probability action delivery). One intuitive schedule could be, to assign $\lfloor K^{(i)}/M \rfloor$ actions to each agent and place the remaining $\hat{K} = K - \lfloor K^{(i)}/M \rfloor M$ actions to the first (fastest) $\hat{K}$ agents, where $K^{(i)}$ is the number of active actions in batch i, i.e., $|\mathcal{A}_i| = K^{(i)}$. This scheduling has end time $T^{(i)} = \max\left(\lfloor \frac{K^{(i)}}{M} \rfloor (\alpha_M + 4^i), (\lfloor \frac{K^{(i)}}{M} \rfloor + 1)(\alpha_{\hat{K}} + 4^i)\right)$ - and although for some α_i values it can perform well, it also fails in many scenarios. For instance, if $\epsilon_m = 0\ \forall m \in [M]$, the end time becomes $4^i K^{(i)}$ whereas the optimal end time is $\lceil 4^i K^{(i)}/M \rceil$ (which is smaller by a factor of M).

Example 2 Another straightforward approach is to first complete 4^i pulls for one action by distributing the pulls across all agents, and then move onto the next. That is, for each action, the learner sends it to all agents, and waits until 4^i effective pulls (i.e., not counting repetitions) are received back. Note that even if an agent m needs to play one pull, we still need to wait first for α_m rounds before collecting this effective reward. This scheduling has an end time $T^{(i)} = K^{(i)} \min_{\tilde{M} \in [M]} (\frac{\sum_{m=1}^{\tilde{M}} \alpha_m + 4^i}{\tilde{M}})$, where $K^{(i)} = |\mathcal{A}_i|$. This can be suboptimal, e.g., if $\alpha_m = \alpha\ \forall m \in [M]$, then the end time of this scheduling is $T^{(i)} = K^{(i)}\alpha + K^{(i)}\lceil \frac{4^i}{M} \rceil$, whereas an end time of $\lceil \frac{K^{(i)}}{M} \rceil (\alpha + 4^i)$ can be achieved using the scheduling explained in Example 1.

[3] Recall that all actions in the set $\mathcal{A}_i$ are expected to have mean values within a bounded distance from the optimal; thus the suboptimal actions in $\mathcal{A}_i$ are expected to accumulate similar regret.

Algorithm 1 BatchSP2 (K, M, $\boldsymbol{\alpha}$)

Input: number of actions K, number of agents M, repetitions $\boldsymbol{\alpha} \in \mathbb{Z}_+^M$
Initialize batch index $i = 1$, set of active actions $\mathcal{A}_1 = [K]$
for batch i **do**
  $\boldsymbol{S}, T^{(i)} =$ Schedule$(\mathcal{A}_i, \boldsymbol{\alpha}, i)$ (see Algorithm 2)
  for t in $[T^{(i)}]$ **do**
    send action S_{mt} to agent m $\forall m \in [M]$
    receive reward r_{mt} $\forall m \in [M]$
  Update means of the actions

$$\mu_a^{(i)} = \sum_{m \in \mathbb{M}_a^{(i)}} \sum_{t=b_{am}^{(i)}}^{e_{am}^{(i)}} r_{mt}/4^i \quad \forall a \in \mathcal{A}_i$$

  ($\mathbb{M}_a^{(i)}$: set of agents that pulls action a in batch i, $b_{am}^{(i)}$ and $e_{am}^{(i)}$: start and end time of the effective pulls, respectively, of a in agent m in batch i, $b_{am}^{(i)}, e_{am}^{(i)} \in [T^{(i)}]$)
  Update active action set: $\mathcal{A}_{i+1} \leftarrow \{a \in \mathcal{A}_i | \max_{j \in \mathcal{A}_i} \mu_j^{(i)} - \mu_a^{(i)} \leq 4\sqrt{\log(KMT)/2 \cdot 4^i}\}$
  $i \leftarrow i + 1$

Our scheduling goal is, given $\alpha_1 \leq \alpha_2 \leq \dots \leq \alpha_M$, to find a schedule that minimizes $T^{(i)}$. As the previous examples illustrate, neither distributing actions across all agents, nor restricting each action to be played in one agent, is optimal. One natural approach is to express the schedule through an Integer Linear Program (provided in appendix 7). The associated LP relaxation, as also discussed in the appendix, essentially associates a cost $\frac{a_m}{4^i}$ with each action pull at agent m, and solves a cost-minimization resource allocation problem. The resulting LP solution gives us a lower bound τ on $T^{(i)}$, where:

$$\tau := 4^i K^{(i)} / \sum_{m=1}^{M} 1/(\frac{\alpha_m}{4^i} + 1). \tag{1}$$

Unfortunately, the LP solution cannot always be easily translated to an integral solution (where each agent m actually plays a_m pulls even if she needs to collect reward for an action only once) while avoiding suboptimal regrets (as compared to the ILP solution).

Instead, we develop a scheduling algorithm that is polynomial time, and carefully balances how to split the 4^i pulls of each action across agents, so as to decrease the number of required repetitions α_i, while still taking advantage as needed from the fact that we have multiple agents. The pseudocode can be found in Algorithm 2.

The algorithm works in two stages: We first round the LP solution to an integer solution, which can schedule at least $(K-M)^+$ actions. In this stage, each action is assigned to at most one agent. In the second stage, we schedule the remaining unscheduled actions by splitting each action among multiple agents. In particular, we assign to the first stage (where we do not split action pulls) a duration τ as in (1): since the LP relaxation manages to allocate the 4^i pulls for all actions before τ, keeping all pulls of an action together before that time can only decrease the total number of repetitions required by each allocated action. We prove in Section 4 that at least $(K-M)^+$[4] actions will be successfully allocated at this stage, leaving $\hat{K}$ remaining actions. In the second stage, we partition the pulls of the remaining $\hat{K}$ actions into smaller parts of size $\max(1, \lfloor M/2\hat{K} \rfloor)$ and use the first $\lfloor M/2 \rfloor$ agents to do the scheduling. Utilizing only the first $\lfloor M/2 \rfloor$ agents allows to find an end time on the scheduling in terms of $c\sum_{m=1}^{M}\alpha_m$ where $c>0$ is some constant instead of a term that depends on K or M, as will become apparent in Section 4.

Algorithm 2 Schedule ($\mathcal{A}$, $\boldsymbol{\alpha}$, i)

Input: set of actions $\mathcal{A}$ with $|\mathcal{A}|=K$, repetitions $\boldsymbol{\alpha} \in \mathbb{Z}_+^M$, batch index i
Initialize $k=0$, $T^{(i)}=\tau$ (see Eq. 1)
Shuffle the set $\mathcal{A}$ randomly
for agent $m \in [M]$ **do**
    Initialize $t_{end}=0$, $p=\alpha_m+4^i$
    while $t_{end}+p \le T^{(i)}$ **do**
        Assign next action to agent
        $k \leftarrow k+1$
        $t_{\text{end}} \leftarrow t_{\text{end}}+p$
for $\hat{K}=K-k$ unassigned actions **do**
    Divide pulls into $\max(1, \lfloor M/2\hat{K} \rfloor)$ equal parts
Assign each part to first $\lfloor M/2 \rfloor$ agents one by one
Imitate assignments of first $\lfloor M/2 \rfloor$ agents for remaining $M-\lfloor M/2 \rfloor$ agents (with their own repetitions)
Update $T^{(i)}$, the end time of the batch, to agent finishing last in first $\lfloor M/2 \rfloor$ agents
Fill remaining slots of the agents randomly
Output: $\boldsymbol{S} \in \mathbb{R}^{M \times T^{(i)}}$ the schedule of actions to agents, $T^{(i)}$ end time

3.1 Connecting to Channels with Delays

We note that our algorithm BatchSP2 (and its analysis, in the next section), directly applies to channels with delays, where an action sent by the learner to an agent m is received after t rounds with some probability $p_t^{(m)}$. Indeed, although we used erasure channels for our narrative in this paper, and motivated use of repetitions over such channels, the only fact that BatchSP2 essentially hinges on is that, agent m will receive a sent action with probability at least $\frac{1}{T}$ after α_m rounds, where α_m is known. It implies that any known (or estimated) delay/probability of successful reception can be used with our algorithms. In our case α_m was dictated from the repetition protocol, in other setups it could be dictated from delivery delay or delivery uncertainty. Datasets and models in literature, e.g., (Sagatov et al., 2020) or (Dahmouni et al., 2012), can provide empirical values for delay/probability in such setups.

4 REGRET ANALYSIS

This section provides our theoretical analysis: we first calculate an upper bound on the end time of each batch in Lemma 1, then use this to derive an upper bound on the expected regret that depends on suboptimality gaps on Theorem 1, and provide a gap-independent regret upper bound on Theorem 2.

Lemma 1 *If the scheduling algorithm outlined in Algorithm 2 is run for batch i, then the end time $T^{(i)}$ of the batch can be bounded as*

$$T^{(i)} \le K4^i\tau + 6\left(\sum_{m=1}^{M}\frac{\alpha_m}{M} + 2\frac{K4^i}{M}\right)$$

where $\tau = \frac{1}{\sum_{m=1}^{M} 1/(\alpha_m/4^i+1)}$, $\alpha_m = \lceil 4\frac{\log T}{\log(1/\epsilon_m)} \rceil - 1$, *$K$ is the number of actions, and M is the number of agents.*

Proof Sketch of Lemma 1. The upper bound on the scheduling end time, hence, total number of pulled actions in a batch, is obtained in two steps. First, because α_m+4^i rounds are sufficient to schedule 4^i effective pulls for a single arm at agent m, we prove that Algorithm 2 schedules at least $(K-M)^+$ agents in time $K4^i\tau$. This implies that at step 10 of Algorithm 2, the number of remaining arms to be scheduled is bounded by M. As the algorithm schedules these arms among the best $M/2$ agents, each agent will be assigned a constant number of arms. The final end time is bounded by noticing that from the averaging principle, the delay of all agents in the best half is bounded by the average delay $\sum_m \alpha_m/M$.

The complete proof is provided in Appendix 8.1.

[4] $x^+ = \max(x,0) \quad \forall x \in \mathbb{R}$

Theorem 1 *Consider a distributed multi-armed bandit setting with K actions and M agents connected through heterogeneous erasure channels with erasure probabilities $\{\epsilon_i\}_{i=1}^M$. If BatchSP2 is run with horizon T, then the expected regret is,*

$$\mathbb{E}[R_T] \leq c\Bigg(\sum_{a:\Delta_a>0}\Big(\frac{\log(KMT)}{\Delta_a} + \frac{M\log(MT)}{\sum_{m=1}^{M} 1/(\alpha_m + \frac{\log(KMT)}{\Delta_a})}\Big) + \sum_{m=1}^{M}\alpha_m\log(MT) + \log(MT)\Bigg)$$

where $\alpha_m = \lceil 4\log T/\log(1/\epsilon_m)\rceil - 1$ is the number of repetitions at agent m, Δ_a is the suboptimality gap for action a, and $c > 0$ a constant.

Proof Sketch of Theorem 1. The regret bound is achieved by decomposing the regret of each batch as $\mathbb{E}[R_T^{(i)}] = \sum_a T_{ia}\Delta_a$ and bounding the expected number of times $\mathbb{E}[T_{ia}]$ that arm a is pulled in batch i. To that end, we condition on a good event entailing that for each agent m and each consecutive $\alpha_m + x$ pulls from action a, the last x rewards are samples from the distribution of action a. This provides a concentration of the empirical means used in Algorithm 1 with high probability. As a result, we get an upper bound on the number of batches a suboptimal arm can survive. Having this, to bound T_{ia}, it only remains to bound the number of times an active action is pulled in batch i, which is highly sensitive to the scheduling of action pulls. This is proved by utilizing the upper bound on the number of pulls in Lemma 1 and leveraging the symmetry imposed by the randomization in Algorithm 2 to show that each action has an equal contribution in the total number of pulls. The final regret bound is obtained by showing that the good event has high probability.

Bounding the excess regret from the rewards not used by the algorithm is a challenging part of the regret analysis. If the schedule is designed naively, these rewards may come from the action with the largest gap in the batch. However, as we show in the proof of Theorem 1, the randomization and shuffling performed in Algorithm 2 make the contributions of the different arms in the excess regret uniform in expectation.

The complete proof is provided in Appendix 8.2.

The three components of the regret bound in Theorem 1 originate from distinct aspects of the algorithm. The initial term, $\sum_{a:\Delta_a>0}\log(KMT)/\Delta_a$, is inevitable, representing the order optimal regret achievable under perfect channels (no delay, no erasure). The second and third terms are due to the repetition and scheduling of actions (Algorithm 2). It is noteworthy that, the second term matches the regret of an optimal scheduling algorithm (see App. 7). Additionally, under perfect channels, this term simplifies to the a lower bound on the regret up to logarithmic factors. The third term, $\sum_{m=1}^{M}\alpha_m\log(MT)$, emerges at the outset of the learning process, reflecting that each agent m will repeat the first pulled (suboptimal) action for α_m iterations on average.

The regret bound in Theorem 1 is nearly constant for constant gaps and erasure probabilities. However, for small gaps, the regret bound can be large. It is important to note that this will not be the actual regret suffered when the gaps are small, as for small Δ, the regret is bounded by $TM\Delta$. The following theorem provides an instance-dependent regret bound that works for all values of the suboptimality gaps.

Theorem 2 *Consider the distributed multi-armed bandit setting with K actions and M agents connected through heterogeneous erasure channels $\{\epsilon_i\}_{i=1}^M$. If BatchSP2 is run for horizon T, then the expected regret is*

$$R_T \leq c\Bigg(M\sqrt{\frac{KT\log(MT)}{\sum_{m=1}^{M} 1/(\alpha_m\Delta_\star + \log(KMT))}} + \sum_{m=1}^{M}\alpha_m\log(KMT)\Bigg)$$

where $\Delta_\star$ is the value satisfying

$$\Delta_\star = \frac{c'K\log(MT)}{T\sum_{m=1}^{M} 1/(\alpha_m + \frac{\log(KMT)}{\Delta_\star})},$$

which can be efficiently approximated using the bisection method, $\alpha_m = \lceil 4\log T/\log(1/\epsilon_m)\rceil - 1$ number of repetitions and $c, c' > 0$ constants.

Proof Sketch of Theorem 2. The regret bound is proved by bounding the regret bound for arms with small gap (less than $\Delta_\star$, which will be determined later) by $TM\Delta_\star$ and using the bound in Theorem 1 for the remaining arms with large gap (greater than $\Delta_\star$). The value of $\Delta_\star$ is chosen to minimize the bound by balancing the regret resulting from arms with small gaps and the regret from arms with large gaps.

The complete proof is provided in Appendix 8.3.

It is worth noting that for $\alpha_m = 0 \quad \forall m$ (no erasures), the regret bound in Theorem 2 reduces to $\tilde{O}(\sqrt{KTM})$, nearly matching the lower bound on the regret for the model considered in Lattimore and Szepesvári (2020). More importantly, our bound shows that if

$\alpha_m = \tilde{O}(1/\Delta_{\max}) \quad \forall m$, where $\Delta_{\max}$ is the maximum gap, then the regret bound is still $\tilde{O}(\sqrt{KTM})$, hence, we (nearly) suffer no extra regret beyond the no erasure case.

For the single agent case $M = 1$, the regret bound in Theorem 2 reduces to $\tilde{O}(\sqrt{KT} + K\alpha)$ which is shown to be nearly optimal in Hanna et al. (2023a).

5 EXPERIMENTS

In this section, we empirically evaluate the regret performance of our proposed algorithm, BatchSP2, and compare against the following methods:

- MA-SAE: This is an extension of SAE (Auer and Ortner, 2010) to multi-agent setting. It utilizes the agents without repeating any actions and considers all the rewards generated in the environment.
- MA-LSAE-V: This is an extension of SAE to multi-agent setting, that restricts all pulls of an action to be played at the same agent, as described in Example 1 in Section 3.
- MA-LSAE-H: This is another extension of SAE to multi-agent setting, that distributes the pulls of an action across all agents, as described in Example 2 in Section 3.
- MA-UCB: This is an extension of Upper Confidence Bound (UCB) (Auer et al., 2002; Lai, 1987) algorithm to multi-agent setting. UCB is an optimal algorithm for a simple MAB setting. Compared to SAE, it makes the decision on which action to pull at each round instead of at each batch.

We have explored a number of experimental setups (in terms of number of actions, channel quality, horizons, etc)[5]; we here show results for two experiments that we believe are representative:

• **Experiment 1**, shown in Figure 1, uses $K = 10$ actions, with Gaussian reward distributions that have variance 1 and means $[0.8, 1, 0, \cdots, 0]$. The time horizon is $T = 5 \times 10^4$ and the regret in each plot is averaged over 100 experiments with arms shuffled. The channels have erasure probabilities 0.2, 0.7, 0.9, and 0.99, and there is an equal number of $M/4$ channels for each erasure probability.

• **Experiment 2**, shown in Figure 2, has all parameters the same as Experiment 1, with the difference that we have now channels with similar erasure probabilities of 0.9, 0.93, 0.95, and 0.99 (as before, there is an equal number of channels for each erasure probability).

[5]The code to our experiments is available here.

From Figure 1, it can be seen that extensions of UCB and SAE may result in linear regret under action erasures even when the suboptimality gap is large (0.2 for the instance used in this experiment). Comparing Figure 1 (a) to (c) for MA-LSAE-V, we can observe how waiting for a bad channel to finish pulling actions slows down the learning: when the number of agents with erasure probability 0.2 increases from left to right, MA-LSAE-V starts assigning actions to agents with small number of repetitions, and cumulative regret gets smaller. This supports the splitting idea behind our algorithm.

From Figure 2, it can be seen that while trends of algorithms are similar to Figure 1 (b) in terms of learning (linear versus logarithmic cumulative regret); when the channel quality gets worse, the gap between MA-LSAE-H and our algorithm widens. This indicates that while repetitions ensure learning with high probability, if we assign action pulls to agents without considering how many additional repetitions are evoked, it can significantly slow down the learning process. In some cases, it might even result in UCB or SAE to having lower regret for an extended period of time, despite their linear regret behavior.

6 CONCLUSION

In this work, we consider the case of a learner connected to multiple distributed agents through heterogeneous channels, that are subject to action erasures without action feedback (the same setup can also capture delays and uncertainty on action reception). If rewards can be externally observed, we may have misinformation, a mismatch between the action the learner requests and the agent plays. Because of this, traditional algorithms can easily fail; instead, we introduce BatchSP2, an efficient algorithm that uses repetition to achieve robustness over erasures and careful allocation of action pulls to agents to minimize regret. We provide a theoretical regret analysis of BatchSP2, which allows to recover as special cases existing bounds, as well as numerical evaluations that show Batch2SP can achieve superior performance over baseline schemes.

Acknowledgements

We thank the anonymous reviewers and the meta-reviewer for their insightful suggestions and comments. This work is supported in part by NSF grants #2007714 and #2221871, by Army Research Laboratory grant under Cooperative Agreement W911NF-17-2-0196, and by the Amazon Faculty Award.

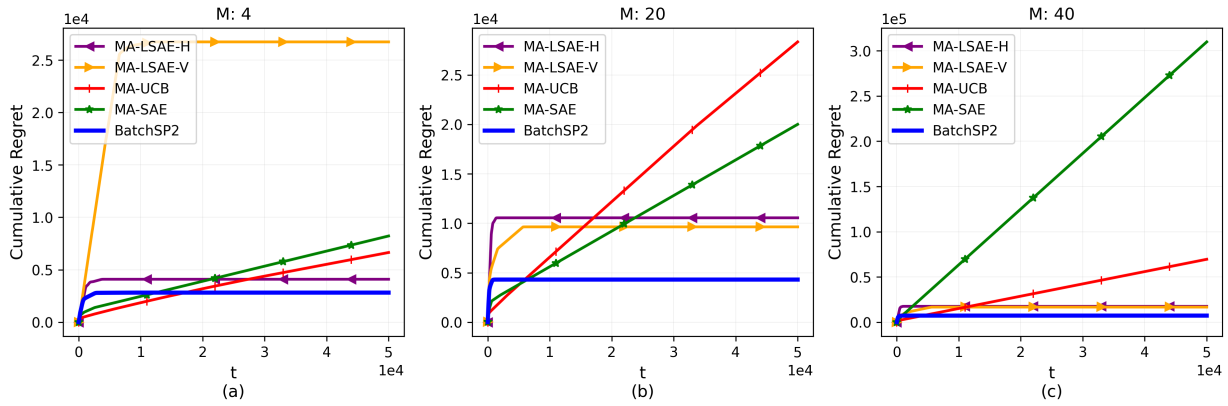

Figure 1: Comparison Results For Different Numbers Of Agents. From Left To Right, The Plots Show Cumulative Regret As A Function Of Rounds t For (a) 4 Agents, (b) 20 Agents, and (c) 40 Agents, Respectively.

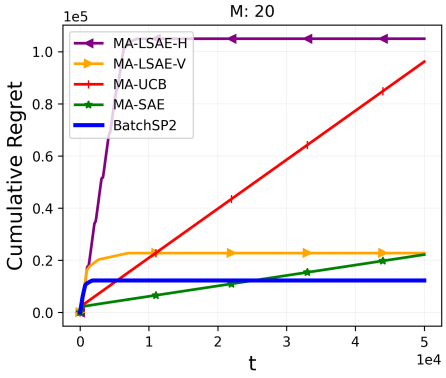

Figure 2: Same Scenario As In Figure 1 (b) With Worse Channel Quality.

References

Mridul Agarwal, Vaneet Aggarwal, and Kamyar Azizzadenesheli. Multi-agent multi-armed bandits with limited communication. *J. Mach. Learn. Res.*, 23 (1), jan 2022. ISSN 1532-4435.

Idan Amir, Idan Attias, Tomer Koren, Yishay Mansour, and Roi Livni. Prediction with corrupted expert advice. *Advances in Neural Information Processing Systems*, 33:14315–14325, 2020.

Peter Auer and Ronald Ortner. Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010.

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32 (1):48–77, 2002. doi: 10.1137/S0097539701398375.

Hamza Dahmouni, André Girard, and Brunilde Sansò. An analytical model for jitter in ip networks. *annals of telecommunications-annales des télécommunications*, 67:81–90, 2012.

Mostafa Dehghan, Weibo Chu, Philippe Nain, Don Towsley, and Zhi-Li Zhang. Sharing cache resources among content providers: A utility-based approach. *IEEE/ACM Transactions on Networking*, 27(2):477–490, 2019.

Abhimanyu Dubey et al. Cooperative multi-agent bandits with heavy tails. In *International conference on machine learning*, pages 2730–2739. PMLR, 2020.

Aditya Grover, Todor Markov, Peter Attia, Norman Jin, Nicolas Perkins, Bryan Cheong, Michael Chen, Zi Yang, Stephen Harris, William Chueh, and Stefano Ermon. Best arm identification in multi-armed bandits with delayed feedback. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 833–842. PMLR, 4 2018.

Osama Hanna, Lin Yang, and Christina Fragouli. Learning from distributed users in contextual linear bandits without sharing the context. *Advances in Neural Information Processing Systems*, 35:11049–11062, 2022a.

Osama A Hanna, Lin Yang, and Christina Fragouli. Solving multi-arm bandit using a few bits of communication. In *International Conference on Artificial Intelligence and Statistics*, pages 11215–11236. PMLR, 2022b.

Osama A Hanna, Merve Karakas, Lin F Yang, and Christina Fragouli. Multi-arm bandits over action erasure channels. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 1312–1317. IEEE, 2023a.

Osama A Hanna, Lin F Yang, and Christina Fragouli.

Compression for multi-arm bandits. *IEEE Journal on Selected Areas in Information Theory*, 2023b.

Pooria Joulani, Andras Gyorgy, and Csaba Szepesvari. Online learning under delayed feedback. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1453–1461, Atlanta, Georgia, USA, 6 2013. PMLR.

James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach*. Pearson, 6th edition edition, 2012. ISBN 978-0132856201.

Tze Leung Lai. Adaptive treatment allocation and the multi-armed bandit problem. *Annals of Statistics*, 15:1091–1114, 1987.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Yangzhe Liu, Zonghao Zou, On Shun Pak, and Alan C. H. Tsang. Learning to cooperate for low-reynolds-number swimming: a model problem for gait coordination. *Scientific Reports*, 13, 2023.

Travis Mandel, Yun-En Liu, Emma Brunskill, and Zoran Popović. The queue method: Handling delay, heuristics, prior data, and evaluation in bandits. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), 2 2015. doi: 10.1609/aaai.v29i1.9604.

Ciara Pike-Burke, Shipra Agrawal, Csaba Szepesvari, and Steffen Grunewalder. Bandits with delayed, aggregated anonymous feedback. In *International Conference on Machine Learning*, pages 4105–4113. PMLR, 2018.

Evgeny Sagatov, Samara Mayhoub, Andrei Sukhov, and Dmitrii Chernysh. Dataset of one-way delay in local and global networks, 2020. URL https://dx.doi.org/10.21227/0dmg-3r29.

Shahin Shahrampour, Alexander Rakhlin, and Ali Jadbabaie. Multi-armed bandits in multi-agent networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2786–2790, 2017. doi: 10.1109/ICASSP.2017.7952664.

Chengshuai Shi, Wei Xiong, Cong Shen, and Jing Yang. Heterogeneous multi-player multi-armed bandits: Closing the gap and generalization. *Advances in neural information processing systems*, 34:22392–22404, 2021.

William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294, 1933.

Claire Vernade, Olivier Cappé, and Vianney Perchet. Stochastic bandit models for delayed conversions. *arXiv preprint arXiv:1706.09186*, 2017.

Po-An Wang, Alexandre Proutiere, Kaito Ariu, Yassir Jedra, and Alessio Russo. Optimal algorithms for multiplayer multi-armed bandits. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 4120–4129. PMLR, 26–28 Aug 2020.

Mengfan Xu and Diego Klabjan. Decentralized randomly distributed multi-agent multi-armed bandit with heterogeneous rewards. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Edmund Yeh, Tracey Ho, Ying Cui, Michael Burd, Ran Liu, and Derek Leong. Vip: A framework for joint dynamic forwarding and caching in named data networks. In *Proceedings of the 1st ACM Conference on Information-Centric Networking*, pages 117–126, 2014.

Z. Zou, Y. Liu, Y. N. Young, and et al. Gait switching and targeted navigation of microswimmers via deep reinforcement learning. *Communications Physics*, 5: 158, 2022.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] Section 2, Section 3
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] Section 3, Section 4
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] Section 5

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [No]
   Error bars are not as it disrupted the figures' interpretability due to high variance towards the end for some algorithms.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Not Applicable]
   Simple experiments that can run on CPU or Google Colaboratory.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Multi-Agent Bandit Learning through Heterogeneous Action Erasure Channels: Supplementary Materials

7 LINEAR PROGRAM FORMULATION

We first formulate a (nonlinear integer) program that minimizes the end time to schedule action pulls in batch i with K actions across M agents as follows:

$$\min_{\boldsymbol{X}\in\mathbb{R}^{M\times K}} \quad \max_{m\in[M]} \sum_{k=1}^{K} (\alpha_m \mathbb{1}[X_{mk} > 0] + X_{mk}) \tag{2a}$$

$$\text{s.t.} \quad \sum_{m=1}^{M} X_{mk} = 4^i \quad \forall k \in [K] \tag{2b}$$

$$X_{mk} \in \{0, 1, 2, \cdots, 4^i\} \quad \forall m \in [M],\ \forall k \in [K], \tag{2c}$$

where $\boldsymbol{X} \in \mathbb{R}^{M\times K}$ captures the variables of the program, with X_{mk} indicating the number of effective pulls of action k performed by agent m. The objective function (2a) is to minimize the latest end time among agents. Constraint (2b) ensures that the total number of effective pulls for each action is 4^i; and constraint (2c) forces effective pulls assigned to each agent per action to be an integral value in $[0, 4^i]$. It is easy to see that the progam in (2) is equivalent to the following integer linear program (ILP):

$$\min_{\substack{\boldsymbol{X},\boldsymbol{W}\in\mathbb{R}^{M\times K}\\ t\in\mathbb{R}}} \quad t \tag{3a}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} (\alpha_m W_{mk} + X_{mk}) \le t, \quad \forall m \in [M] \tag{3b}$$

$$X_{mk} \le 4^i W_{mk} \quad \forall m \in [M],\ \forall k \in [K] \tag{3c}$$

$$\sum_{m=1}^{M} X_{mk} = 4^i \quad \forall k \in [K] \tag{3d}$$

$$X_{mk} \in \{0, 1, 2, \cdots, 4^i\} \quad \forall m \in [M],\ \forall k \in [K] \tag{3e}$$

$$W_{mk} \in \{0, 1\} \quad \forall m \in [M],\ \forall k \in [K], \tag{3f}$$

where the variable $t \in \mathbb{R}$ replaces the max in objective equation 2a and the variable $\boldsymbol{W} \in \mathbb{R}^{M\times K}$ replaces the indicator function. Notice that for any feasible solution $\boldsymbol{X}$, if $X_{mk} > 0$, $W_{mk} = 1$. The relaxed version of the ILP in (3) can be written as

$$\min_{\substack{\boldsymbol{X},\boldsymbol{W}\in\mathbb{R}^{M\times K}\\ t\in\mathbb{R}}} \quad t \tag{4a}$$

$$\text{s.t.} \quad \sum_{k=1}^{K}(\alpha_m W_{mk} + X_{mk}) \le t, \quad \forall m \in [M] \tag{4b}$$

$$\boldsymbol{X}_{mk} \le 4^i W_{mk} \quad \forall m \in [M],\ \forall k \in [K] \tag{4c}$$

$$\sum_{m=1}^{M} X_{mk} = 4^i \quad \forall k \in [K] \tag{4d}$$

$$0 \le X_{mk} \le 4^i \quad \forall m \in [M],\ \forall k \in [K] \tag{4e}$$

$$0 \le W_{mk} \le 1 \quad \forall m \in [M],\ \forall k \in [K], \tag{4f}$$

Notice that the minimum value W_{mk} can take is $X_{mk}/4^i$ due to (4c); hence, by replacing W_{mk} with its minimum value, we get the following linear program which gives a lower bound on the ILP (3):

$$\min_{\substack{\boldsymbol{X}\in\mathbb{R}^{M\times K}\\ t\in\mathbb{R}}} \quad t \tag{5a}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} X_{mk}\left(\frac{\alpha_m}{4^i} + 1\right) \le t \quad \forall m \in [M] \tag{5b}$$

$$\sum_{m=1}^{M} X_{mk} = 4^i \quad \forall k \in [K] \tag{5c}$$

$$0 \le X_{mk} \quad \forall m \in [M],\ \forall k \in [K], \tag{5d}$$

In the linear program (5), X_{mk} is the variable that indicates how many effective pulls are assigned to agent m for action k. (5c) forces each action to be pulled 4^i effective times; however, instead of an integer number of pulls, each agent is allowed to perform nonnegative fractional pulls. Furthermore, (5b) indicates that for each agent m, one effective pull has a cost of $\frac{\alpha_m}{4^i} + 1$.

Claim 1 *The optimal objective value of (5) satisfies, $t^\star = \sum_{k=1}^{K} X^\star_{mk}(\alpha_m/4^i + 1)\ \forall m \in M$, where $(t^\star, \boldsymbol{X}^\star)$ is the optimal solution of 5.*

Proof of Claim 1. First, we observe that at least one of the inequalities in (5b) holds with equality, otherwise the value of $t^\star$ can be decreased leading to a better objective. Define the set of indices $\mathcal{E}_i := \{m \in [M] : \sum_{k=1}^{K} X^\star_{mk}(\alpha_m/4^i + 1) = t^\star\}$.

Now, assume Claim 1 is not correct. And let m_s be such that

$$\sum_{k=1}^{K} X^\star_{m_s k}(\alpha_{m_s}/4^i + 1) < t^\star.$$

Then $\forall m \in \mathcal{E}_i\ \exists\{\beta_{mk}\}_{k=1}^{K} \ge 0 : \sum_k \beta_{mk} > 0$ small enough such that

$$X'_{mk} = \begin{cases} X^\star_{mk} - \beta_{mk}, & m \in \mathcal{E}_i,\ \forall k \in [K] \\ X^\star_{mk} + \sum_{k=1}^{K} \beta_{mk}, & m_s = m \\ X^\star_{mk} & \text{otherwise} \end{cases}$$

$$t' = \max_m \{\sum_{k=1}^{K} X^{'}_{mk}(\alpha_m/4^i + 1)\} < t^\star$$

forms a feasible solution in (5) with a smaller objective value $t' < t^\star$; hence, $(t^\star, \boldsymbol{X}^\star)$ cannot be optimal. Then at the optimal solution $(t^\star, \boldsymbol{X}^\star)$, $t^\star = \sum_{k=1}^{K} X^\star_{mk}(\alpha_m/4^i + 1)\ \forall m \in M$.

Using Claim 1 and the constraint (5c);

$$4^i K = \sum_{m=1}^{M}\sum_{k=1}^{K} X^\star_{mk} = \sum_{m=1}^{M} \frac{t^\star}{(\alpha_m/4^i+1)} \;=\; t^\star \sum_{m=1}^{M} \frac{1}{(\alpha_m/4^i+1)} \;\Rightarrow\; t^\star \;=\; \frac{4^i K}{\sum_{m=1}^{M} 1/(\alpha_m/4^i+1)} \tag{6}$$

which justifies equation 1.

Observation Note that the solution of the relaxed LP (5) can be directly used for scheduling of actions by adding $\max(2\alpha_{M-1}, \alpha_m)$ to the end time $t^\star$. Since the relaxation in general does not give a feasible solution for the ILP, we add $\max(2\alpha_{M-1}, \alpha_M)$ to the end time of the relaxed ILP to guarantee a feasible solution for the ILP. As the additional time slots accumulate regret across all agents, this can result in $\Omega(M\alpha_M)$ additional regret which can be large for large M. Our algorithm improves the M factor in $M\alpha_M$.

8 MISSING PROOFS

Table 2: Notation

4^i	:	Number of effective pulls in batch i for each active action
$\mathcal{A}$	:	Set of actions, $\lvert\mathcal{A}\rvert = K$
$\mathcal{A}_i$	:	Set of active actions in batch i, $\lvert\mathcal{A}_i\rvert = K^{(i)}$
α_m	:	$= \lceil 4\log T/\log(1/\epsilon_m)\rceil - 1$, number of repetitions for agent m
Δ_a	:	$= \max_{a'\in\mathcal{A}} \mu_{a'} - \mu_a$, suboptimality gap for action a
G	:	The event that at least one instruction among the times $t, t+1, \cdots, t+\alpha_m-1$ will not be erased for all agents m and all times t
G'_i	:	$= \left\{ \lvert\mu_a^{(j)} - \mu_a\rvert \;\le\; 2\sqrt{\frac{\log(KMT)}{2\cdot 4^j}} \;\; \forall a\in\mathcal{A}_j,\; j\in[i-1] \right\}$, the event that empirical means of active actions in batch j ($\forall a \in A_j$) is in confidence region for all batches until batch i
M	:	Number of agents
μ_a	:	Reward mean of action a
$\mu_a^{(i)}$	:	The empirical mean calculated for action a at batch i (as defined in step 8 in Algorithm 1)
$N_1^{(i)}$	:	$= \frac{M}{\sum_{m=1}^{M} 1/(\alpha_m+4^i)}$, a term that appears in regret
$N_2^{(i)}$	:	$= 12\cdot 4^i$, a term that appears in regret
R_T	:	Regret of K arm bandit over M channels with horizon T
$R_T^{(i)}$	:	Regret of batch i for K arm bandit over M channels with horizon T
$T^{(i)}$	:	Length of the scheduling outputted by Algorithm 2 for batch i
T_i	:	The total number of instructions played by all agents due to instructions sent in batch i
T_{ia}	:	Number of times action a is played by agents due to an instruction sent in batch i
$\hat{K}$	:	Number of actions unassigned in the first part of the scheduling (as described in Algorithm 2 line 10)

8.1 Proof of Lemma 1

In this section, we present the detailed proof of Lemma 1.

Lemma 1 *If the scheduling algorithm outlined in Algorithm 2 is run for batch i, then the end time $T^{(i)}$ of the batch can be bounded as*

$$T^{(i)} \le K4^i\tau + 6\left(\sum_{m=1}^{M}\frac{\alpha_m}{M} + 2\frac{K4^i}{M}\right)$$

where $\tau = \frac{1}{\sum_{m=1}^{M} 1/(\alpha_m/4^i+1)}$, $\alpha_m = \lceil 4\frac{\log T}{\log(1/\epsilon_m)}\rceil - 1$, K is the number of actions, and M is the number of agents.

We prove the upper bound on the end time in two steps.
Step A. First, we claim that the algorithm uses the first $4^i K\tau$ rounds to schedule all 4^i pulls of at least $(K-M)^+$ actions:
Each agent m takes $\alpha_m + 4^i$ to complete all pulls of an action; hence, it can play all pulls of at least

$$\left\lfloor \frac{4^i K\tau}{\alpha_m + 4^i} \right\rfloor$$

actions. Hence, the total number of actions scheduled across all channels during the first $4^i K\tau$ rounds is

$$\sum_{m=1}^{M} \left\lfloor \frac{4^i K\tau}{\alpha_m + 4^i} \right\rfloor \;\geq\; \sum_{m=1}^{M} \left(\frac{4^i K\tau}{\alpha_m + 4^i} - 1 \right) \;=\; K\tau \sum_{m=1}^{M} \frac{1}{\alpha_m/4^i + 1} - M \;=\; K - M.$$

A lower bound of $(K-M)^+$ follows by the non-negativity of the number of scheduled pulls.
Step B. The second step is to show that the remaining number of actions $\hat{K} \leq K - (K-M)^+ = \min(K, M)$ can be scheduled using an additional time of

$$6\left(\frac{\sum_{m=1}^{M} \alpha_m}{M} + 2\frac{4^i K}{M} \right).$$

Recall that Algorithm 2 divides the 4^i pulls of each of the remaining actions into $\max(1, \lfloor M/2\hat{K} \rfloor)$equal parts and assign each part to an agent. Hence, each part will have number of pulls

$$\frac{4^i}{\max(1, \lfloor M/2\hat{K} \rfloor)} \overset{(a)}{\leq} \min(4^i, \frac{4\hat{K}}{M} 4^i) \tag{7}$$

and there will be at most M such parts. The first $\lfloor M/2 \rfloor$ agents can be used for scheduling these parts in a way such that each agent is assigned at most three parts. It follows that each agent m needs at most $3\alpha_m + 3\min(4^i, \frac{4\hat{K}}{M} 4^i)$ time to perform the scheduled pulls. Thus the total number of rounds required to schedule the remaining pulls can be bounded by

$$3 \max_{m \in \{1, \cdots, \lfloor M/2 \rfloor\}} \alpha_m + 3\min(4^i, \frac{4\hat{K}}{M} 4^i) \overset{(i)}{=} 3\alpha_{\lfloor M/2 \rfloor} + 3\min(4^i, \frac{4\hat{K}}{M} 4^i) \overset{(ii)}{\leq} 6\frac{\sum_{m=1}^{M} \alpha_m}{M} + 3\min(4^i, \frac{4\hat{K}}{M} 4^i) \tag{8}$$

where $(i), (ii)$ follow from the fact that α_m's are ordered, i.e., $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_M$. Combining this with the result from **Step A**, we get that the end time needed to send all actions in batch i.

8.2 Proof of Theorem 1

Theorem 1 *Consider a distributed multi-armed bandit setting with K actions and M agents connected through heterogeneous erasure channels with erasure probabilities $\{\epsilon_i\}_{i=1}^{M}$. If BatchSP2 is run with horizon T, then the expected regret is,*

$$\mathbb{E}[R_T] \leq c\left(\sum_{a:\Delta_a > 0} \left(\frac{\log(KMT)}{\Delta_a} + \frac{M\log(MT)}{\sum_{m=1}^{M} 1/(\alpha_m + \frac{\log(KMT)}{\Delta_a})} \right) + \sum_{m=1}^{M} \alpha_m \log(MT) + \log(MT) \right)$$

where $\alpha_m = \lceil 4\log T / \log(1/\epsilon_m) \rceil - 1$ is the number of repetitions at agent m, Δ_a is the suboptimality gap for action a, and $c > 0$ a constant.

The regret bound is reached by bounding the number of batches a suboptimal arm can survive as a function of the suboptimality gap, conditioned on a good event that we specify later. This gives a bound on the maximum sub-optimality gap at each batch which in turn gives a bound on the regret using the bound on the batch length given in Lemma 1.

Let G be the event that for all agents m and for all times t, at least one instruction among the times $t, t+1, \cdots, t+\alpha_m - 1$ will not be erased. Hence, the event G means that for any agent m, we cannot have α_m or

more consecutive erasures. This implies that, conditioned on G, when an action a is sent $\alpha_m + 4^i$ consecutive times by the learner to agent m, each of the last 4^i pulls will generate a reward from the distribution of action a. We call these last 4^i pulls, the effective pulls. The probability of the compliment of G can be bounded as

$$\mathbb{P}[G^c] \overset{(i)}{\leq} \sum_{m=1}^{M}\sum_{t=1}^{T} {\epsilon_m}^{\alpha_m} \overset{(ii)}{\leq} \sum_{m=1}^{M}\sum_{t=1}^{T} \frac{1}{T^4} \overset{(iii)}{\leq} \frac{1}{MT}, \tag{9}$$

where (i) follows by the union bound over all agents m and times t, (ii) uses $\alpha_m = \lceil 4\log T/\log(1/\epsilon_m)\rceil - 1$, and (iii) follows from $M \leq T$.

Define an event G_i' as

$$G_i' = \left\{ |\mu_a^{(j)} - \mu_a| \ \leq \ 2\sqrt{\frac{\log(KMT)}{2\cdot 4^j}} \ \ \forall a \in \mathcal{A}_j, \ j \in [i-1] \right\},$$

where $\mu_a^{(j)}$ is the empirical mean calculated for action a at batch j, as defined in step 8 in Algorithm 1. By Hoeffding's inequality and the fact that rewards lie in $[0,1]$ almost surely, we have that $\mathbb{P}[G_i'|G] \geq 1 - 0.25/(MT)$. Consequently, events G and G_i' happening together have a probability

$$\mathbb{P}[G_i' \cap G] \ \geq \ (1 - 0.25/(MT))^2 \ \geq \ 1 - 2/(MT). \tag{10}$$

We first bound the number of batches, a suboptimal arm can survive as a function of the suboptimality gap. Conditioned on $G \cap G_{i+1}'$ and the elimination criterion in Algorithm 1, a sub-optimal action a can survive getting eliminated in batch i only if $4\sqrt{\frac{\log(KMT)}{2\cdot 4^i}} \geq \Delta_a/2$. This implies that a can be in $\mathcal{A}_{i+1}$ only when

$$i \leq \left\lceil \log_4 \left(\frac{32\log(KMT)}{\Delta_a^2} \right) \right\rceil, \tag{11}$$

i.e., whenever the batch number i is greater than the bound provided in equation 11, $a \notin \mathcal{A}_i$.

Using the result of Lemma 1, we know the number of sent instructions in each batch i is upper bounded as

$$MT^{(i)} \ \leq \ K^{(i)} M \cdot 4^i \tau + 6\sum_{m=1}^{M} \alpha_m + 12K^{(i)}4^i, \tag{12}$$

where $K^{(i)} = |\mathcal{A}_i|$ is the number of actions at the start of batch i and $T^{(i)}$ is the length of batch i. Conditioned on the event G (we cannot have α_m consecutive erasures for any agent m), the last action played by agent m in batch i will be played at most α_m times in batch $i+1$ (due to potential erasures). This implies that the total number of instructions, T_i, played by all agents due to instructions sent in batch i, can be bounded as

$$T_i \ \leq \ \sum_{m=1}^{M} (T^{(i)} + \alpha_m) \ \leq \ K^{(i)} M \cdot 4^i \tau + 7\sum_{m=1}^{M} \alpha_m + 12K^{(i)}4^i. \tag{13}$$

We utilize the following proposition, restated and proved at the end of section 8.2, to bound the expected number of times a certain action is played due to an instruction sent in batch i.

Proposition 1 *Conditioned on $(G \cap G_i', \mathcal{A}_i)$, the expected number of times arm a is played due to an instruction sent in batch i is the same for all $a \in \mathcal{A}_i$. In particular, $\mathbb{E}[T_{ia}|G \cap G_i', \mathcal{A}_i] = \mathbb{E}[T_{ia'}|G \cap G_i', \mathcal{A}_i], \quad \forall a, a' \in \mathcal{A}_i$.*

Conditioning on $\mathcal{A}_i$ in the previous proposition and in the following abbreviates conditioning on the event that the random set of surving actions in batch i takes the value $\mathcal{A}_i$.

Then, we have that

$$\mathbb{E}[T_{ia}|G \cap G_i', \mathcal{A}_i] \ = \ \frac{\mathbb{E}[T_i|G \cap G_i', \mathcal{A}_i]}{K^{(i)}} \ \leq \ M \cdot 4^i \tau + 7\frac{\sum_{m=1}^{M} \alpha_m}{K^{(i)}} + 12 \cdot 4^i \quad \forall a \in \mathcal{A}_i. \tag{14}$$

Let $R_T^{(i)}$ be the regret of batch i. The regret of the algorithm can be bounded as

$$\begin{aligned}
\mathbb{E}[R_T] &= \sum_{i=1}^{\log(MT)} \mathbb{E}[R_T^{(i)}] \leq \sum_{i=1}^{\log(MT)} \left(\mathbb{E}[R_T^{(i)}|G\cap G_i'] + MT(1-\mathbb{P}[G\cap G_i'])\right) \\
&\overset{(a)}{\leq} \sum_{i=1}^{\log(MT)} (\mathbb{E}[\mathbb{E}[R_T^{(i)}|G\cap G_i', \mathcal{A}_i]] + 1) = \sum_{i=1}^{\log(MT)} \mathbb{E}[\sum_a \mathbb{E}[T_{ia}|G\cap G_i', \mathcal{A}_i]\Delta_a] + \log(MT) \\
&\overset{(b)}{\leq} \sum_{i=1}^{\log(MT)} \Bigg(\mathbb{E}[\sum_a (N_1^{(i)} + N_2^{(i)})\mathbb{E}[\mathbb{1}[a\in\mathcal{A}_i]|G\cap G_i']\Delta_a] \\
&\qquad + \mathbb{E}[\sum_a 7\sum_{m=1}^{M} \frac{\alpha_m}{K^{(i)}}\mathbb{E}[\mathbb{1}[a\in\mathcal{A}_i]|G\cap G_i', \mathcal{A}_i]\Delta_a]\Bigg) + \log(MT) \\
&\leq \sum_{i=1}^{\log(MT)} \sum_a (N_1^{(i)} + N_2^{(i)})\mathbb{E}[\mathbb{1}[a\in\mathcal{A}_i]|G\cap G_i']\Delta_a + \mathbb{E}[7\sum_{m=1}^{M}\alpha_m] + \log(MT) \\
&\leq \sum_{i=1}^{\log(MT)} \sum_a (N_1^{(i)} + N_2^{(i)})\mathbb{E}[\mathbb{1}[a\in\mathcal{A}_i]|G\cap G_i']\Delta_a + c''\log(MT)\sum_{m=1}^{M}\alpha_m + \log(MT), \qquad (15)
\end{aligned}$$

where (a) follows from law of total expectation and equation 10, (b) follows from equation 14 and we use $N_1^{(i)} = \frac{M}{\sum_{m=1}^{M} 1/(\alpha_m + 4^i)}$ and $N_2^{(i)} = 12\cdot 4^i$ for these quantities that do not depend on $\mathcal{A}_i$. We will bound each term in equation 15 separately to get the final regret bound.

We start by bounding the effect of the first term in equation 15, $N_1^{(i)} = \frac{M}{\sum_{m=1}^{M} 1/(\alpha_m+4^i)}$, on the final regret bound. We have that

$$\begin{aligned}
\sum_{i=1}^{\log(MT)} \sum_a N_1^{(i)}\mathbb{E}[\mathbb{1}[a\in\mathcal{A}_i]|G\cap G_i']\Delta_a &= \sum_a \sum_{i=1}^{\log(MT)} \Delta_a \frac{M\mathbb{E}[\mathbb{1}[a\in\mathcal{A}_i]|G\cap G_i']}{\sum_{m=1}^{M} 1/(\alpha_m+4^i)} \\
&\overset{(a)}{\leq} c\sum_{a:\Delta_a>0} \frac{M\log(MT)}{\sum_{m=1}^{M} 1/(\alpha_m + \frac{\log(KMT)}{\Delta_a})} \qquad (16)
\end{aligned}$$

where c is a universal constant, and (a) follows from equation 11 and the bound being an increasing function of i.

The effect of the second term in equation 15, $N_2^{(i)} = 12\cdot 4^i$,

$$\sum_a \sum_{i=1}^{\log(MT)} N_2^{(i)}\mathbb{E}[\mathbb{1}[a\in\mathcal{A}_i]|G\cap G_i']\Delta_a = 12\sum_a \sum_{i=1}^{\left\lceil \log_4\left(\frac{32\log(KMT)}{\Delta_a^2}\right)\right\rceil} 4^i\Delta_a \overset{(a)}{\leq} c'\sum_{a:\Delta_a>0}\frac{\log(KMT)}{\Delta_a}, \qquad (17)$$

where (a) follows from equation 11, and c' is a universal constant. The final result follows by summing the bounds in equation 16 and equation 17.

Proposition 1 *Conditioned on $(G\cap G_i', \mathcal{A}_i)$, the expected number of times arm a is played due to an instruction sent in batch i is the same for all $a\in\mathcal{A}_i$. In particular, $\mathbb{E}[T_{ia}|G\cap G_i', \mathcal{A}_i] = \mathbb{E}[T_{ia'}|G\cap G_i', \mathcal{A}_i], \quad \forall a, a'\in\mathcal{A}_i$.*

Proof. Recall that T_{ia} is the number of times an agent plays arm a due to an instruction sent in batch i. We represent the schedule by the set $S = \{\{S_{mt}\}_{m=1}^{M}\}_{t=1}^{T^{(i)}}$, where S_{mt} is the action the learner sends to agent m at time t. Let $S(a\leftrightarrow a')$ represents the schedule where actions a, a' are exchanged in the schedule S, i.e., $S(a\leftrightarrow a')_{mt} = a$ whenever $S_{mt} = a'$, $S(a\leftrightarrow a')_{mt} = a'$ whenever $S_{mt} = a$, otherwise $S(a\leftrightarrow a')_{mt} = S_{mt}$. We

notice that conditioned on the schedule S in batch i, whether an action is played in slot t due to an instruction sent in batch i is only a function of the erasures in batches $i, i+1, ...$ Hence, we have that

$$\begin{aligned}\mathbb{E}[T_{ia}|G\cap G_i', \mathcal{A}_i] = \sum_{S\in\mathbb{S}} \mathbb{P}[S|G\cap G_i', \mathcal{A}_i]\mathbb{E}[T_{ia}|G\cap G_i', \mathcal{A}_i, S] = \sum_{S\in\mathbb{S}} \mathbb{P}[S|\mathcal{A}_i]\mathbb{E}[T_{ia}|G, S] \\ \overset{(a)}{=} \sum_{S\in\mathbb{S}} \frac{1}{|\mathbb{S}|}\mathbb{E}[T_{ia}|G, S] = \sum_{S\in\mathbb{S}} \frac{1}{|\mathbb{S}|}\mathbb{E}[T_{ia'}|G, S(a\leftrightarrow a')] = \sum_{S\in\mathbb{S}} \frac{1}{|\mathbb{S}|}\mathbb{E}[T_{ia'}|G, S] = \mathbb{E}[T_{ia'}|G\cap G_i', \mathcal{A}_i],\end{aligned} \tag{18}$$

where $\mathbb{S}$ is the set of all (non-zero probability) possible schedules for batch i, and (a) follows since the randomization in Algorithm 2 makes all the schedules in $\mathbb{S}$ equally probable.

8.3 Proof of Theorem 2

Theorem 2 *Consider the distributed multi-armed bandit setting with K actions and M agents connected through heterogeneous erasure channels $\{\epsilon_i\}_{i=1}^M$. If BatchSP2 is run for horizon T, then the expected regret is*

$$R_T \leq c\left(M\sqrt{\frac{KT\log(MT)}{\sum_{m=1}^M 1/(\alpha_m\Delta_\star + \log(KMT))}} + \sum_{m=1}^M \alpha_m \log(KMT)\right)$$

where $\Delta_\star$ is the value satisfying

$$\Delta_\star = \frac{c'K\log(MT)}{T\sum\limits_{m=1}^M 1/(\alpha_m + \frac{\log(KMT)}{\Delta_\star})},$$

which can be efficiently approximated using the bisection method, $\alpha_m = \lceil 4\log T/\log(1/\epsilon_m)\rceil - 1$ number of repetitions and $c, c' > 0$ constants.

Proof of Theorem 2. From equation 15, the expected regret can be bounded as

$$\begin{aligned}\mathbb{E}[R_T] &\leq \sum_{i=1}^{\log(MT)}\sum_a \mathbb{E}[T_{ia}|G\cap G_i']\Delta_a + \log(MT) \\ &\leq MT\Delta + \sum_{i=1}^{\log(MT)}\sum_{a:\Delta_a>\Delta} \mathbb{E}[T_{ia}|G\cap G_i']\Delta_a + \log(MT) \\ &\overset{(a)}{\leq} MT\Delta + \sum_{i=1}^{\log(MT)}\sum_{a:\Delta_a>\Delta} (N_1^{(i)} + N_2^{(i)})\mathbb{1}[a\in\mathcal{A}_i]\Delta_a + c'''\log(MT)\sum_{m=1}^M \alpha_m + \log(MT) \\ &\overset{(b)}{\leq} MT\Delta + c\sum_{a:\Delta_a>\Delta}\left(\frac{M\log(MT)}{\sum\limits_{m=1}^M 1/(\alpha_m + \frac{\log(KMT)}{\Delta_a})} + \frac{\log(KMT)}{\Delta_a}\right) + c'''\log(MT)\sum_{m=1}^M \alpha_m + \log(MT) \\ &\overset{(c)}{\leq} MT\Delta + c\sum_{a:\Delta_a>\Delta}\frac{2M\log(MT)}{\sum\limits_{m=1}^M 1/(\alpha_m + \frac{\log(KMT)}{\Delta_a})} + c'''\log(MT)\sum_{m=1}^M \alpha_m + \log(MT) \\ &\overset{(d)}{=} MT\Delta + \frac{c'KM\log(MT)}{\sum\limits_{m=1}^M 1/(\alpha_m + \frac{\log(KMT)}{\Delta})} + c'''\log(MT)\sum_{m=1}^M \alpha_m + \log(MT) \\ &\leq 2\max\left\{TM\Delta, \frac{c'KM\log(MT)}{\sum\limits_{m=1}^M 1/(\alpha_m + \frac{\log(KMT)}{\Delta})}\right\} + c'''\log(MT)\sum_{m=1}^M \alpha_m + \log(MT) \quad \forall\Delta > 0\end{aligned} \tag{19}$$

where $c, c', c''' > 0$ some constants. (a) follows from equation 14 where $N_1^{(i)} = M/(\sum_{m=1}^{M} 1/(\alpha_m + 4^i))$ and $N_2^{(i)} = 12 \cdot 4^i$. (b) follows from directly substituting equation 16 and equation 17 for the terms; and (c) follows from the fact that the first term is an increasing function of α_m's; therefore,

$$\frac{M \log(MT)}{\sum_{m=1}^{M} 1/(\alpha_m + \frac{\log(KMT)}{\Delta_a})} \geq \frac{M \log(MT)}{\sum_{m=1}^{M} 1/\frac{\log(KMT)}{\Delta_a}} \geq \frac{\log(KMT)}{\Delta_a} \quad \forall \{\alpha_m\}_{m=1}^{M} \geq 0.$$

(d) follows from the term inside the summation being a decreasing function of Δ_a.

We choose Δ to be the value that minimizes the bound. Hence the optimal value $\Delta_\star$ satisfies:

$$TM\Delta_\star = \frac{c' KM \log(MT)}{\sum_{m=1}^{M} 1/(\alpha_m + \frac{\log(KMT)}{\Delta_\star})} \tag{20}$$

Substituting equation 20 to the bound in equation 19, we get that

$$\mathbb{E}[R_T] \leq 2M \sqrt{\frac{c' KT \log(MT)}{\sum_{m=1}^{M} 1/(\alpha_m \Delta_\star + \log(KMT))}} + c''' \log(MT) \sum_{m=1}^{M} \alpha_m + \log(MT). \tag{21}$$