2025 American Control Conference (ACC)
July 8-10, 2025. Denver, CO, USA

Exploiting Adjacent Similarity in Multi-Armed Bandit Tasks via Transfer of Reward Samples

NR Rahul[1] and Vaibhav Katewa[2]

Abstract— **We consider a sequential multi-task problem, where each task is modeled as the stochastic multi-armed bandit with K arms. We assume the bandit tasks are adjacently similar in the sense that the difference between the mean rewards of the arms for any two consecutive tasks is bounded by a parameter. We propose two algorithms (one assumes the parameter is known while the other does not) based on UCB to transfer reward samples from preceding tasks to improve the overall regret across all tasks. Our analysis shows that transferring samples reduces the regret as compared to the case of no transfer. We provide empirical results for our algorithms, which show performance improvement over the standard UCB algorithm without transfer and a naive transfer algorithm.**

I. INTRODUCTION

In sequential multi-task settings, an agent encounters a sequence of tasks to be solved. The agent can transfer information from previously solved tasks to new and similar tasks to help improve performance of the new task[1][2]. In the context of multi-armed bandits [3][4], the information from one bandit is used to make decisions in another similar bandit task[5]. This is particularly useful in scenarios such as the user cold start problem [6] in recommender systems, where good initial recommendations are made by using information gathered from similar users. Similarly, in reinforcement learning, the learned policies/model from one task is used in another similar task to help speed up learning[7][8]. Reusing information also helps to address the problem of data efficiency in reinforcement learning.

In this paper, we consider a sequential multi-task setting, where the agent interacts with each task sequentially, one after the other. Each task is modeled by a stochastic multi-armed bandit problem, where the agent interacts by pulling one of the arms at any given time and, in return, gets a random reward. We assume the tasks are adjacently similar and introduce a parameter ϵ in Section II to capture this similarity between tasks. The parameter ϵ captures many interesting scenarios like the similarity between different users based on region, age, gender, etc, and the changing user preferences in recommender systems, changing market trends in online advertising, etc. The goal is to use the information from the previously solved tasks in order to improve the performance in the current task, therefore leading to overall performance improvement. This is achieved by reusing/transferring reward samples from previously encountered tasks to the current task. Our algorithm is inspired by [9], which is based on Upper Confidence Bound (UCB) algorithm [10].

Related Work. Several works in transfer learning for bandits have focused on linear[11], [12] or contextual bandits [13], [14], where an explicit form of the reward function is assumed. This makes the problem simpler and allows the derivation of mathematical results. In contrast, we consider transfer learning in the most generic case of stochastic multi-armed bandits, where no such assumption of the reward function is made. In [6], the authors study sequential transfer in stochastic multi-armed bandits. However, they consider a fixed number of MAB tasks. In contrast, we study sequential transfer in stochastic multi-armed bandits for infinite tasks. The authors in [15] have extended this framework to infinite linear bandits tasks that are close in l_2 distance. In our previous paper on transfer in MAB [9], we considered the notion of universal similarity, where all tasks are similar. In contrast, we study adjacent similarity in this paper. Therefore, we transfer reward samples from the preceding task and not from all previous tasks. Additionally, the number of reward samples to transfer is controlled through a novel parameter, which effectively transfers more samples if the tasks are close and fewer samples if the tasks are not. On the other hand, in [9], all previous reward samples are transferred. Thus, this paper generalizes the setting of [9] in a non-trivial manner.

Another similar set of problems are non-stationary bandits [16], [17], which are different from our setting in the sense that the task-switching instants are unknown. Although, in our setting, the task-switching instants are known, we provide transfer algorithms to improve the performance over the no-transfer UCB-based algorithm (NT-UCB). Note that the algorithms in the literature of non-stationary bandits use NT-UCB with known switching task instants as the oracle algorithm. Therefore, we believe our approach of transferring reward samples to non-stationary bandits will achieve better performance (which we defer to future work).

Main Contributions. The main contributions of the paper are:

1) We propose Tr-UCB algorithm to transfer information using the reward samples from the preceeding task to the current task in a sequential multi-task bandit setting. We extend Tr-UCB to Tr-UCB2 algorithm to handle the case of unknown parameter ϵ.
2) We provide the regret analysis for Tr-UCB and Tr-UCB2 and show that there is no negative transfer. Our

NR Rahul is with the Department of Electrical Communication Engineering (ECE) at the Indian Institute of Science, Bengaluru, India. Email: rahulnr@iisc.ac.in.

Vaibhav Katewa is with the Robert Bosch Center for Cyber-Physical Systems and the Department of ECE at the Indian Institute of Science, Bengaluru, India. Email:vkatewa@iisc.ac.in

This work is supported in part by SERB Grant MTR/2022/000522.

regret upper bound clearly captures the performance improvement due to transfer.

3) We provide empirical evaluation of the algorithms Tr-UCB and Tr-UCB2 and show effectiveness of transferring information from previous tasks.

Notations: $\mathbb{1}\{E\}$ denotes the indicator function whose value is 1 if the event (condition) E is true, and 0 otherwise. Similarly, for n events E_1, E_2, $\cdots$, E_n, where $n \in \mathbb{N}$, we define $\mathbb{1}\{E_1, E_2, \cdots, E_n\}$ as the indicator function whose value is 1 if all the events are true, and 0 otherwise. Further, let $\emptyset$ denote the null set, and let $[l]$ denote the set $\{1, 2, \cdots, l\}$ for some $l \in \mathbb{N}$.

II. Preliminaries and Problem Statement

We consider a sequential multi-task problem, where each task is modeled as a stochastic multi-armed bandit with K arms. Let J denote the total number of tasks, and n_j denote the task length/total steps in task j. Further, let the total number of steps in the J tasks be denoted by T and is given by $T = \sum_{j=1}^{J} n_j$. In task j, at each time step t (denotes the number of steps from the beginning of the task j), the agent makes a decision denoted by $I_t^j \in [K]$ to pull one of the K arms, and in turn, receives a random reward $r_{I_t^j} \in [0, 1]$. Let $\mathbf{r}_t^j = \{r_{I_1^j}, r_{I_2^j}, ..., r_{I_t^j}\}$ denote the corresponding rewards received from steps 1 to t in task j. The reward samples are independent across time and across arms, and their probability distributions are unknown. Let μ_k^j be the mean reward of arm k in task j. We define k_*^j and μ_*^j to be an optimal arm in task j and its mean reward, respectively, and are given by

$$k_*^j \in \mathcal{A}^j = \arg\max_{k \in [K]} \{\mu_k^j\} \quad \text{and} \quad \mu_*^j = \max_{k \in [K]} \{\mu_k^j\}.$$

Define $\Delta_k^j = \mu_*^j - \mu_k^j > 0$ as the sub-optimality gap of arm $k \notin \mathcal{A}^j$ in task j. In our setting, the agent encounters a sequence of multi-armed bandit tasks (refer to Figure 1). We assume the tasks are adjacently similar in the sense that the mean rewards of consecutive tasks do not change considerably. The following assumption captures the similarity between any two consecutive tasks.

Assumption 1. *We assume that* $|\mu_k^j - \mu_k^{j+1}| \leq \epsilon_k$ *for all* $j \in [J-1]$*, and the parameter* $\epsilon_k \in [0, 1), \forall k \in [K]$.

This assumption implies that for each arm $k \in [K]$, the mean rewards between any two consecutive tasks do not differ by more than ϵ_k. One application where this assumption is relevant is online advertising and recommender systems, wherein user preferences do not change drastically over time. Note that for any given task, Assumption 1 can be leveraged to have a better inference about the optimal mean reward (and its corresponding optimal arm) of that particular task by using additional reward samples from the previous similar tasks.

The goal of the agent in the sequential multi-armed bandit setting in any given task j and time t is to make decisions I_t^j based on the reward samples $\{\{\mathbf{r}_{n_l}^l\}_{l=1}^{j-1}, \mathbf{r}_{t-1}^j\}$ to maximize the expected total reward over all the bandit tasks. This is captured in terms of the total pseudo-regret as

$$R_J = \sum_{j=1}^{J} R_{n_j} = \sum_{j=1}^{J} \left[n_j \mu_*^j - \mathbb{E}\left[\sum_{t=1}^{n_j} \mu_{I_t^j}^j \right] \right]. \quad (1)$$

Equivalently, the goal is to make decisions $\{I_t^j : 1 \leq t \leq n_j, \forall j \in [J]\}$ to minimize the regret in (1).

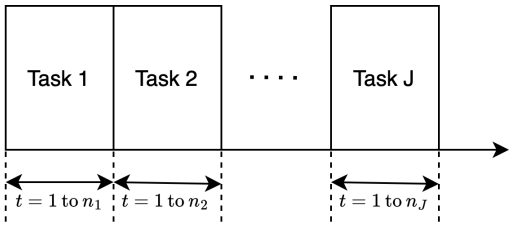


Fig. 1: Sequential multi-task bandit setting

In this paper, we leverage the relation between the mean rewards of any two consecutive tasks (c.f. Assumption 1) in order to minimize the regret R_J. This is accomplished by reusing/transferring reward samples from previous tasks to make decisions in the current task. In the next section, we present two algorithms - one assumes that the parameter ϵ_k in Assumption 1 is known, and the other for the case when ϵ_k is unknown. We provide the regret analysis of these algorithms and compare them with the baseline algorithm, which works without reusing/transferring reward samples from previous tasks.

III. Algorithms and Regret Analysis

Our algorithms are based on Upper Confidence Bound algorithm (UCB) [10] for bandits. UCB is a popular strategy for balancing exploration and exploitation by selecting arms based on the upper confidence bounds of their estimated mean rewards. A simple extension of UCB to the sequential bandit setting is to use UCB separately for each task. In particular, this approach uses reward samples only from the current task to compute the upper confidence bounds of the estimated mean rewards. We call this approach as No Transfer-UCB (NT-UCB) algorithm. However, NT-UCB does not leverage the relation between any two consecutive tasks (c.f. Assumption 1). In contrast, we leverage Assumption 1 by transferring reward samples from the previous task to the current task. We call this algorithm Transfer-UCB (Tr-UCB). Next, we discuss these algorithms in detail.

A. No Transfer-UCB (NT-UCB)

The NT-UCB algorithm is a straightforward extension of the UCB algorithm [10] to the sequential multi-armed bandit task setting. It involves resetting the mean reward estimates and, therefore, the upper confidence bounds at the beginning of a new task. Essentially, when transitioning to a new task, NT-UCB discards reward samples from previous tasks and recomputes the confidence bounds based on the rewards obtained in the current task. In any task, the NT-UCB

algorithm (shown in Algorithm 1) begins by pulling each arm once. From $t \geq K+1$, the algorithm computes the sample average estimates of mean reward denoted by

$$\hat{\mu}_{1k}^{j}(t) = \frac{\sum_{\tau=1}^{t} r_{I_\tau}^{j} \mathbb{1}\{I_\tau = k\}}{N_k^j(t)}, \tag{2}$$

where $N_k^j(t)$ denotes the number of times arm k is pulled until time t in the task j. Let $q_{1k}^j(t)$ denote its corresponding confidence width which is computed as follows

$$q_{1k}^{j}(t) = \sqrt{\frac{\alpha \log t}{2N_k^j(t)}}, \tag{3}$$

where $\alpha > 2$. Next, the NT-UCB algorithm uses $\hat{\mu}_{1k}^j(t-1) + q_{1k}^j(t-1)$ to make decision I_t^j at time t in task j based on the "optimism in the face of uncertainty principle"

$$I_t^j = \arg\max_{k\in[K]} \left\{ \hat{\mu}_{1k}^j(t-1) + q_{1k}^j(t-1) \right\}. \tag{4}$$

Next, we provide the regret upper bound of the NT-UCB algorithm. Let $\Delta_k^{\max} = \max_{j\geq 1} \{\Delta_k^j\}$ and $\Delta_k^{\min} = \min_{j\geq 1, \Delta_k^j > 0} \{\Delta_k^j\}$ denote the universal (over all tasks) upper and lower bound on the sub-optimality gap of arm k. Let $\alpha > 2$ be a positive integer. Then, we have the following bound on the regret.

Lemma 1. *The total pseudo-regret of NT-UCB satisfies*

$$R_J \leq \sum_{k=1}^{K} \left[\sum_{\substack{j=1 \\ \Delta_k^j > 0}}^{J} \frac{2\alpha \log n_j}{\Delta_k^j} + \frac{\alpha}{\alpha - 2} \sum_{j=1}^{J} \Delta_k^j \right]. \tag{5}$$

Proof. Follows from the regret upper bound of standard UCB algorithm [3]. □

Remark 1. *When the total number of tasks satisfies $J = \mathcal{O}(T^\beta)$, where $\beta \in [0,1)$, the regret in Lemma 1 becomes $R_T = \mathcal{O}\left(\left(\sum_{k=1}^{K} \frac{1}{\Delta_k^{\min}} \right) T^\beta \log(T) \right) + \mathcal{O}\left(\left(\sum_{k=1}^{K} \Delta_k^{\max} \right) T^\beta \right) = \mathcal{O}(T^\beta \log(T))$.*

B. Transfer-UCB (Tr-UCB) with parameter ϵ_k known

The NT-UCB algorithm uses reward samples from the current task to make decisions I_t. However, by Assumption 1, the mean rewards of consecutive tasks are similar. Hence, samples from previous tasks contain information about the mean reward of the current task. To leverage this information, we construct an auxiliary estimate using the reward samples from the preceding task. Subsequently, the UCB and auxiliary estimates are combined to make decisions I_t. Next, we describe this Tr-UCB algorithm in detail (shown in Algorithm 2).

Algorithm 1 NT-UCB

Require: Total tasks J, parameter α, and number of arms K

for task $j = 1, 2, ..., J$ **do**
  for $t = 1, \cdots, K$ **do**
    $I_t^j = t$ (Pull each arm once)
  end for
  for $t = K+1, \cdots, n_j$ **do**
    compute $\hat{\mu}_{1k}^j(t-1)$ using (2), $\forall k \in [K]$
    compute $q_{1k}^j(t-1)$ using (3), $\forall k \in [K]$
    select arm $I_t^j = \arg\max_{k\in[K]} \{\hat{\mu}_{1k}^j(t-1) + q_{1k}^j(t-1)\}$
    update number of pulls $N_k^j(t)$, $\forall k \in [K]$
  end for
end for

Let $\hat{\mu}_{2k}^j(t)$ denote the auxiliary estimate of the mean reward of arm k at time t in task j. The auxiliary estimate $\hat{\mu}_{2k}^j(t)$ is computed using the reward samples from the current task j and the preceding task $j-1$. Further, let $B_k = \frac{\eta - 4\epsilon_k^2}{4\epsilon_k^2}$ denote the maximum number of transferred samples for arm k from the preceding task, where $\eta > 8$. The term B_k is large when the parameter ϵ_k is small, which means a large number of reward samples from the preceding task are allowed to be transferred; on the other hand, fewer reward samples are transferred when ϵ_k is large. By limiting the number of transferred reward samples through B_k, the amount of bias introduced in the estimation of mean in the current task is not excessive and remains sufficient to facilitate the transfer. Then, the auxiliary estimate $\hat{\mu}_{2k}^j(t)$ is computed as:

$$\hat{\mu}_{2k}^{j}(t) = \frac{R_k^j + \sum_{\tau=1}^{t} r_{I_\tau}^{j} \mathbb{1}\{I_\tau = k\}}{N_k^j(t) + M_k^j}, \tag{6}$$

where $R_k^j = \sum_{\tau=1}^{n_{j-1}} r_{I_\tau}^j \mathbb{1}\{I_\tau = k; N_k^{j-1}(\tau) \leq B_k\}$ denotes the sum of transferred rewards, and $M_k^j = \min\{N_k^{j-1}(n_{j-1}), B_k\}$ denotes the number of transferred reward samples for the arm k. Note that for $\epsilon_k = 0$, we have $M_k^j = N_k^{j-1}(n_{j-1})$, which means all the reward samples from the preceding task are transferred. Further, observe that the reward samples from the previous tasks other than the preceding task also carry information about the mean reward of the current task and, therefore, can be used to compute the auxiliary estimate. This is particularly useful if the length of the preceding task is small, which means there are fewer reward samples to transfer, even though the term B_k is very large. However, to keep the algorithm and the analysis simple, we have considered transferring reward samples only from the preceding task[1]. Next, we compute the confidence width denoted by $q_{2k}^j(t)$ of the auxiliary estimate

[1]The extension of the proposed algorithm to the case of transferring reward samples from multiple previous tasks is similar, and we defer it to future work.

$\hat{\mu}^j_{2k}(t)$ as follows:

$$q^j_{2k}(t) = \sqrt{\frac{\eta \log\left(B_k + t\right)}{2\left(N^j_k(t) + M^j_k\right)}}. \tag{7}$$

Finally, the Tr-UCB algorithm makes decision I^j_t at time t by combining the upper confidence bounds of the mean reward computed using the sample average estimate $\hat{\mu}^j_{1k}(t)$ and the auxiliary estimate $\hat{\mu}^j_{2k}(t)$ as follows:

$$I^j_t = \arg\max_{k\in[K]} \{\min\{\hat{\mu}^j_{1k}(t-1) + q^j_{1k}(t-1), \hat{\mu}^j_{2k}(t-1) + q^j_{2k}(t-1)\}\}. \tag{8}$$

Taking the minimum of the upper confidence bounds of the two estimates gives a conservative upper bound on the true value of the mean reward, which allows the algorithm to be less optimistic. This leads to increased exploitation and decreased exploration, therefore leading to a reduction in regret.

Next, we provide the regret result of Tr-UCB. For simplicity, we introduce the following notations:

$$u^j_{1k} \triangleq \frac{2\alpha \log\left(n_j\right)}{(\Delta^j_k)^2} \text{ and } u^j_{2k} \triangleq \frac{2\eta \log\left(B_k + n_j\right)}{(\Delta^j_k)^2}. \tag{9}$$

Theorem 1. *Let u^j_{1k} and u^j_{2k} be defined as in* (9)*. The pseudo-regret of Tr-UCB satisfies*

$$R_J \leq \sum_{k=1}^{K} \Delta^{\max}_k \left(\left(\sum_{l=0}^{\lceil \frac{J-2}{2} \rceil} \min\left\{U^l_k, V^l_k\right\} \right) + W^J_k + J\left(\frac{\alpha}{\alpha - 2} + \frac{8}{\eta - 8}\right)\right), \tag{10}$$

where,

$$U^l_k = u^{2l+1}_{1k} \mathbb{1}\{\Delta^{2l+1}_k > 0\} + u^{2l+2}_{1k} \mathbb{1}\{\Delta^{2l+2}_k > 0\},$$

$$V^l_k = \left(u^{2l+1}_{2k} \mathbb{1}\{\Delta^{2l+1}_k > 0\} + u^{2l+2}_{2k} \mathbb{1}\{\Delta^{2l+2}_k > 0\}\right) - \min\left\{\max\left\{u^{2l+1}_{2k}, u^{2l+2}_{2k}\right\}, B_k\right\}$$

and $W^J_k = \mathbb{1}\{J \text{ is odd}, \Delta^J_k > 0\}\left(\min\left\{u^J_{1k}, u^J_{2k}\right\}\right)$.

Proof. Refer to the Appendix in the full version of the paper archived paper at [18]. □

Remark 2. *Observe that when the total number of tasks $J = \mathcal{O}(T^\beta)$, where $\beta \in [0, 1)$, the regret in* (10) *follows $R_T = \mathcal{O}(T^\beta \log(T))$, which is the same as for NT-UCB. Therefore, Tr-UCB ensures there is no negative transfer by using reward samples from the preceding tasks.*

Although the algorithms have the same order of growth with respect to T, we show the benefit of transfer by comparing the actual regret expressions.

Benefit of Transfer. We show the benefit of transfer by comparing the expressions of regret in (10) and (5). The first term in the regret bound captures the benefit of transfer, and therefore we compare the first terms of (10) and (5), respectively. Define the following terms for the tasks $2l+1$ and $2l+2$, for some $l \in \{0, 1, \cdots, \lceil \frac{J-2}{2} \rceil\}$

$$A^l_k \triangleq \Delta^{\max}_k U^l_k, \qquad E^l_k \triangleq \Delta^{\max}_k V^l_k,$$

$$F^l_k \triangleq u^{2l+1}_{1k} \Delta^{2l+1}_k + u^{2l+2}_{1k} \Delta^{2l+2}_k$$

Further, we rewrite the regret upper bound of NT-UCB in (5) using F^l_k as follows,

$$R_J \leq \sum_{k=1}^{K} \left(\left(\sum_{\substack{l=0 \\ \Delta^{2l+1}_k > 0, \Delta^{2l+2}_k > 0}}^{\lceil \frac{J-2}{2} \rceil} F^l_k \right) + \frac{\alpha}{\alpha - 2} \sum_{j=1}^{J} \Delta^j_k \right), \tag{11}$$

We analyze the benefit of transfer for the consecutive tasks $2l+1$ and $2l+2$. Note that for the transfer to be useful for the tasks $2l+1$ and $2l+2$, we need $\min\{A^l_k, E^l_k\} < F^l_k$. Since $A^l_k \geq F^l_k$, this can happen only if $E^l_k < F^l_k$. Observe that when B_k is very small, the term E^l_k is relatively large, and as B_k increases, the term E^l_k decreases in comparison with F^l_k. Hence, for some large enough B_k, we get $E^l_k < F^l_k$, which leads to a decrease in the regret upper bound of Tr-UCB. Recall that B_k depends on the parameter ϵ_k through an inverse relationship, i.e. as ϵ_k decreases B_k increases. Therefore, the regret upper bound of Tr-UCB decreases in comparison to NT-UCB, when the parameter ϵ_k decreases.

Algorithm 2 Tr-UCB - Parameter ϵ_k is known

Require: Total tasks J, number of arms K, and parameters α,η, $\epsilon_k, \forall k \in [K]$

for task $j = 1, 2, ..., J$ **do**
  repeat steps 2 to 4 of Algorithm 1
  for $t = K+1, \cdots, n_j$ **do**
    compute $\hat{\mu}^j_{1k}(t-1)$ using (2), $\forall k \in [K]$
    compute $q^j_{1k}(t-1)$ using (3), $\forall k \in [K]$
    compute $\hat{\mu}_{2k}(t-1)$ using (6),$\forall k \in [K]$
    compute $q^j_{2k}(t-1)$ using (7), $\forall k \in [K]$
    select arm I^j_t using (8) at time t
    update number of pulls $N^j_k(t)$, $\forall k \in [K]$
  end for
end for

C. Transfer-UCB (Tr-UCB2) with parameter ϵ_k unknown

The Tr-UCB algorithm discussed previously assumes that the parameter values ϵ_k are known. However, access to these parameters may not be available in practice. One approach to address this problem is to estimate the value of ϵ_k using the past reward samples and then follow the Tr-UCB algorithm using the estimated value. We employ the same approach but with a minor modification to the Tr-UCB algorithm. To increase the confidence in the estimates of ϵ_k, we pull the arms uniformly for a fixed number of steps. This increases

the number of reward samples for each arm, and therefore, increases the confidence in the estimates of ϵ_k. We call this approach Tr-UCB2. Next, we describe Tr-UCB2 algorithm in detail (refer to Algorithm 3 for pseudo-code).

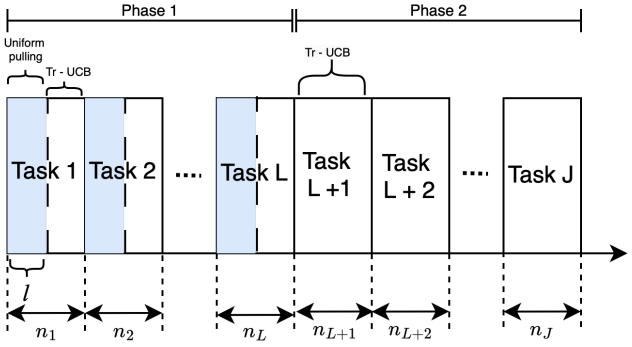


Fig. 2: Pictorial representation of Tr-UCB2

As depicted in Figure 2, the algorithm's behavior is divided into two phases. Let $L \geq 2$ denote the number of tasks in phase I. For each task in phase I, the Tr-UCB2 algorithm begins by pulling the arms uniformly for steps $1 \leq t \leq l$, where $l = aK$, for some $a \in \mathbb{N}$. This ensures enough reward samples from each arm are generated for estimating ϵ_k with good confidence. For $t > l$, the algorithm uses an estimate of ϵ_k (described below) and follows the Tr-UCB strategy. The estimate of ϵ_k is computed at the beginning of every task using the reward samples from the previous tasks. After L tasks, the algorithm enters Phase II which is similar to Phase I except that there is no uniform sampling in the tasks of Phase II. Similar to Phase I, an estimate of ϵ_k is computed at the beginning of every task in Phase II and Tr-UCB strategy is used. Algorithm 3 mentions all steps of TR-UCB2.

Next, we explain the computation of the estimate of ϵ_k in detail. Let $\hat{\epsilon}_k^j$ denote an estimate of the parameter ϵ_k for arm k in task j. Then, $\hat{\epsilon}_k^j$ is computed as follows:

$$\hat{\epsilon}_k^j = \begin{cases} 1 & , 1 \leq j \leq 2 \\ \max\limits_{i \in [j-1], c_k^i \leq c_0} \{|\hat{\mu}_{1k}^{i+1}(n_{i+1}) - \hat{\mu}_{1k}^i(n_i) \pm c_k^i|\}, & 2 < j \leq J \end{cases} \tag{12}$$

where

$$c_k^i = \sqrt{\frac{N_k^{i+1}(n_{i+1}) + N_k^i(n_i)}{2N_k^{i+1}(n_{i+1})N_k^i(n_i)} \log\left(\frac{2}{\delta}\right)}, \text{ and } \delta \in (0,1),$$

$$c_0 = \sqrt{\frac{K}{l} \log\left(\frac{2}{\delta}\right)}.$$

We outline the motivation for the expression of the estimate $\hat{\epsilon}_k^j$ in the following three steps:

(i) From Assumption 1, we know that $|\mu_k^i - \mu_k^{i+1}| \leq \epsilon_k$. Hence, we compute a high probability upper bound on the true value of ϵ_k using the difference between the empirical averages of the mean rewards, i.e., $\hat{\mu}_{1k}^{i+1}(n_{i+1}) - \hat{\mu}_{1k}^i(n_i)$. By using Hoeffding's inequality [19], it can be shown that with probability at least $1 - \delta$, the true difference of the mean rewards $\mu_k^i - \mu_k^{i+1}$ lies in the interval $[\hat{\mu}_{1k}^{i+1}(n_{i+1}) - \hat{\mu}_{1k}^i(n_i) - c_k^i, \hat{\mu}_{1k}^{i+1}(n_{i+1}) - \hat{\mu}_{1k}^i(n_i) + c_k^i]$.

(ii) By considering $|\hat{\mu}_{1k}^{i+1}(n_{i+1}) - \hat{\mu}_{1k}^i(n_i) \pm c_k^i|$, we are essentially taking the maximum possible value of the difference in the mean rewards from the confidence interval $[\hat{\mu}_{1k}^{i+1}(n_{i+1}) - \hat{\mu}_{1k}^i(n_i) - c_k^i, \hat{\mu}_{1k}^{i+1}(n_{i+1}) - \hat{\mu}_{1k}^i(n_i) + c_k^i]$. This method is pessimistic (biased towards higher values of $\hat{\epsilon}_k$) in estimating the true value of ϵ_k, thus helping in minimizing the negative transfer, especially when the estimates are not sufficiently accurate.

(iii) Since the parameter ϵ_k upper bounds the difference in the mean rewards, we take the maximum value among all the estimates of the difference in the mean rewards. Note that the constraint $c_k^i \leq c_0$ ensures the estimates of the difference in mean rewards $\hat{\mu}_{1k}^{i+1}(n_{i+1}) - \hat{\mu}_{1k}^i(n_i)$ have good confidence. This is because, the constraint $c_k^i \leq c_0$ ensures that c_k^i is never too large, in other words, the estimates computed using fewer samples are not considered. Therefore the constraint ensures the estimates have high confidence.

In the following theorem, we provide the regret analysis of Tr-UCB2,

Theorem 2. *The pseudo-regret of Tr-UCB2 satisfies*

$$R_J \leq \sum_{k=1}^{K} \Delta_k^{\max} \left(\frac{lL}{K} + \sum_{j=1}^{J} u_{1k}^j + J\left(\frac{\alpha}{\alpha - 2} + \frac{8}{\eta - 8}\right) + TJ\delta \right). \tag{13}$$

Furthermore, if the total number of tasks satisfies $J = \mathcal{O}(T^\beta)$ and confidence parameter satisfies $\delta = \frac{1}{T}$, then the regret follows $R_T = \mathcal{O}(T^\beta \log(T))$.

Proof. Refer to the Appendix in the full version of the paper archived paper at [18]. □

Next, we analyze the regret upper bound 13 of Tr-UCB2 algorithm. The first term in (13) captures the increased regret due to the uniform pulling of arms in phase I. This means that increasing the parameters l and L, increases the regret. However, to estimate ϵ_k with high confidence, the parameters l and L need to be chosen sufficiently large. Hence, there is a tradeoff in estimating ϵ_k with high confidence and reducing the regret contributed by uniformly pulling the arms. The second term of (13) does not explicitly capture the transfer benefit, unlike the regret upper bound of Tr-UCB in (10). However, Tr-UCB2 is atleast as good as NT-UCB since both follow the same regret order and the empirical results in Section IV further demonstrate that Tr-UCB2 performs better than NT-UCB. Nevertheless, this result is somewhat loose and does not fully capture the transfer benefit, which we aim to improve in future work. The last term captures the

increased regret due to the error in the estimate of ϵ_k. By decreasing δ, the regret due to the last term decreases. However, the parameter l needs to be increased simultaneously in order to increase confidence in the estimate of ϵ_k, showing a trade-off.

Algorithm 3 Tr-UCB2 - Parameter ϵ_k is unknown

Require: Parameters α, l, L, K and J

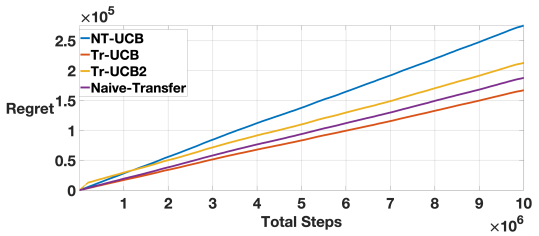
for task $j = 1, 2, ..., J$ **do**
  compute estimate $\hat{\epsilon}_k^j, \forall k \in [K]$
  if $j \leq L$ **then**
    for $t = 1, \cdots, l$ **do**
      $I_t = t \bmod K$
    end for
    for $t = l+1, \cdots, n_j$ **do**
      repeat steps 4 to 9 of Algorithm 2 using $\hat{\epsilon}_k^j$
    end for
  else
    repeat steps 2 to 10 of Algorithm 2 using $\hat{\epsilon}_k^j$
  end if
end for

IV. Numerical Simulations

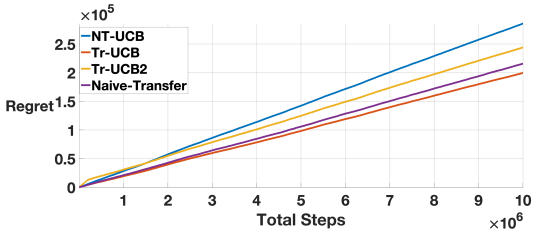
In this section, we present the simulation results showing the empirical improvement of algorithms Tr-UCB, Tr-UCB2 over NT-UCB and a naive transfer algorithm (called Naive-Transfer), indicating the benefit of transfer. Naive-Transfer is an empirical algorithm that transfers all reward samples from the preceding task, assuming that the samples come from the same distribution. We consider a sequence of tasks, where each task is a multi-armed bandit with $K = 5$ arms. The total number of tasks $J = 1000$, with task length $n_j = 10000, \forall j \in [J]$. The mean rewards $\{\mu_k^1\}_{k=1}^K$ of the first task are generated by uniformly sampling from the $[0, 1]$ interval. The mean rewards of the subsequent tasks have to satisfy Assumption 1. Towards this end, we construct uniform distributions of mean μ_k^1 and width $2\epsilon_k$ for each arm k. If the support of any distribution lies outside $[0, 1]$ interval, then we appropriately adjust the width. Then, the mean rewards μ_k^2 of task 2 are uniformly sampled from these distributions. The mean rewards of the subsequent tasks are generated from the mean rewards of the preceding task in a similar manner. The rewards for arm k in any task j are generated by sampling from uniform distributions of mean μ_k^j with width $d = 0.1$. Once again, the widths are adjusted if they fall outside the $[0, 1]$ interval.

Figure 3 shows the empirical regret over the total steps of algorithms NT-UCB, Tr-UCB, Tr-UCB2, and Naive-Transfer for different values of ϵ_k. We have considered $\epsilon_k = \epsilon, \forall k \in [K]$. The plots are generated by averaging over 20 realizations. The parameter values used in Tr-UCB2 algorithm are $l = 2000$, $L = 20$, $\delta = 0.1$ and $\alpha = \eta = 8.1$. The same value of α is used for NT-UCB algorithm.

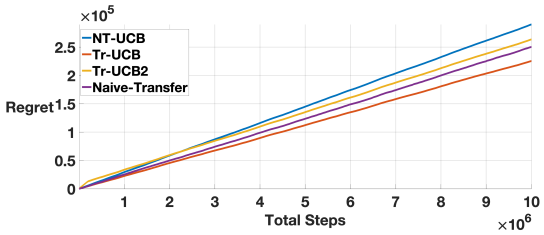
Next, we analyze the results in Figure 3. For each value of ϵ_k, the regret of the Tr-UCB algorithm is the lowest



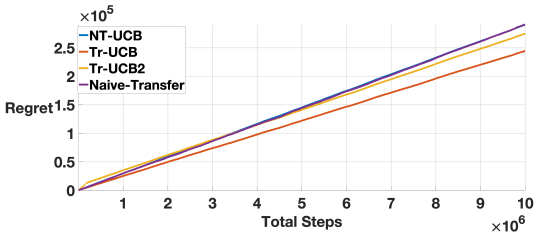
(a) $\epsilon_k = 0.05$



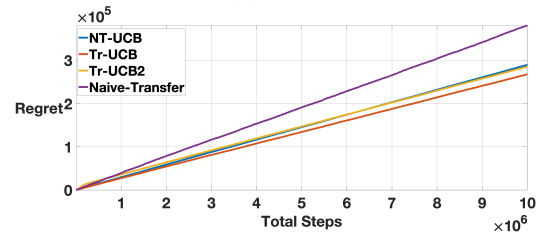
(b) $\epsilon_k = 0.1$



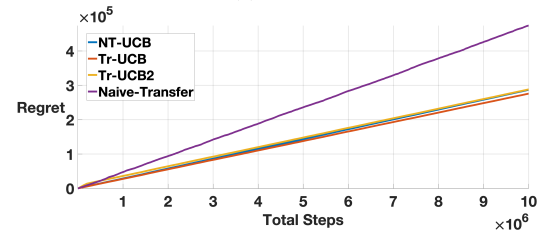
(c) $\epsilon_k = 0.15$



(d) $\epsilon_k = 0.2$



(e) $\epsilon_k = 0.3$



(f) $\epsilon_k = 0.4$

Fig. 3: Empirical Regret Vs Total Steps of NT-UCB, Tr-UCB, Tr-UCB2 and Naive-Transfer algorithms for different values of ϵ_k

among all algorithms, which means transferring the reward samples with the knowledge of ϵ_k helps reduce the regret significantly. Next, observe that Tr-UCB2 incurs more regret than Tr-UCB since ϵ_k is not known. Also, both Tr-UCB and Tr-UCB2 have significantly lower regret than NT-UCB, especially when ϵ_k values are small and lie below NT-UCB overall ϵ_k values considered (no negative transfer). This implies transferring reward samples from preceding tasks is beneficial. On the other hand, Naive-Transfer performs well when ϵ_k values are small, with increasing ϵ_k values, the performance starts degrading and performs worse than NT-UCB for larger values of ϵ_k. The reason for this behavior is that transferring a large number of reward samples naively introduces a large bias in the mean estimates of the reward in the current task. Further, Tr-UCB2 incurs high regret in the initial tasks because of uniform sampling of the arms and inaccurate estimates of ϵ_k. However, it performs better as more tasks are encountered.

V. CONCLUSION

We considered a sequential multi-task setting, where each task is a stochastic multi-armed bandit. We introduced the parameter ϵ_k to capture the adjacent similarity between tasks. We analyzed the transfer of reward samples and proposed two transfer algorithms based on UCB; one assumes the knowledge of ϵ_k, and the other estimates this parameter from data. We provided a regret analysis of the algorithms and validated our approach via numerical experiments. One of the future research directions is to address the gap between the performance of Tr-UCB and Tr-UCB2. Other research directions include computation of lower bound on the regret, and studying transfer learning in the context of varying task similarity parameters, non-stationary bandits and reinforcement learning.

REFERENCES

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[2] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[3] S. Bubeck, N. Cesa-Bianchi, *et al.*, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[4] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[5] D. Bouneffouf, I. Rish, and C. Aggarwal, "Survey on applications of multi-armed and contextual bandits," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, IEEE, 2020.

[6] N. Silva, H. Werneck, T. Silva, A. C. Pereira, and L. Rocha, "Multi-armed bandits in recommendation systems: A survey of the state-of-the-art and future directions," *Expert Systems with Applications*, vol. 197, p. 116669, 2022.

[7] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey.," *Journal of Machine Learning Research*, vol. 10, no. 7, 2009.

[8] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[9] N. Rahul and V. Katewa, "Transfer in sequential multi-armed bandits via reward samples," in *2024 European Control Conference (ECC)*, pp. 2083–2089, 2024.

[10] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, pp. 235–256, 2002.

[11] L. Cella, A. Lazaric, and M. Pontil, "Meta-learning with stochastic linear bandits," in *International Conference on Machine Learning*, pp. 1360–1370, PMLR, 2020.

[12] L. Cella, K. Lounici, G. Pacreau, and M. Pontil, "Multi-task representation learning with stochastic linear bandits," in *International Conference on Artificial Intelligence and Statistics*, pp. 4822–4847, PMLR, 2023.

[13] C. Cai, T. T. Cai, and H. Li, "Transfer learning for contextual multi-armed bandits," *The Annals of Statistics*, vol. 52, no. 1, pp. 207–232, 2024.

[14] A. A. Deshmukh, U. Dogan, and C. Scott, "Multi-task learning for contextual bandits," *Advances in neural information processing systems*, vol. 30, 2017.

[15] M. Soare, O. Alsharif, A. Lazaric, and J. Pineau, "Multi-task linear bandits," in *NIPS2014 workshop on transfer and multi-task learning: theory meets practice*, 2014.

[16] A. Garivier and E. Moulines, "On upper-confidence bound policies for switching bandit problems," in *International conference on algorithmic learning theory*, pp. 174–188, Springer, 2011.

[17] F. Liu, J. Lee, and N. Shroff, "A change-detection based framework for piecewise-stationary multi-armed bandit problem," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[18] N. Rahul and V. Katewa, "Exploiting adjacent similarity in multi-armed bandit tasks via transfer of reward samples," 2024. Available from `https://arxiv.org/abs/2409.19975`.

[19] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *The collected works of Wassily Hoeffding*, pp. 409–426, 1994.