Competing Bandits in Non-Stationary Matching Markets

Avishek Ghosh [1] **Abishek Sankararaman** [2] **Kannan Ramchandran** [3] **Tara Javidi** [4 5] **Arya Mazumdar** [4]

Abstract

Understanding complex dynamics of two-sided online matching markets, where the demand-side agents compete to match with the supply-side (arms), has recently received substantial interest. To that end, in this paper, we introduce the framework of decentralized two-sided matching market under non stationary (dynamic) environments. We adhere to the serial dictatorship setting, where the demand-side agents have unknown and different preferences over the supply-side (arms), but the arms have fixed and known preference over the agents. We propose and analyze an asynchronous and decentralized learning algorithm, namely Non-Stationary Competing Bandits (`NSCB`), where the agents play (restrictive) successive elimination type learning algorithms to learn their preference over the arms. The complexity in understanding such a system stems from the fact that the competing bandits choose their actions in an asynchronous fashion, and the lower ranked agents only get to learn from a set of arms, not *dominated* by the higher ranked agents, which leads to *forced exploration*. With carefully defined complexity parameters, we characterize this *forced exploration* and obtain sub-linear (logarithmic) regret of `NSCB`. Furthermore, we validate our theoretical findings via experiments.

1. Introduction

Repeated decision making by multiple agents in a competitive and uncertain environment is a key characteristic of modern day, two sided markets, e.g., TaskRabbit, UpWork, DoorDash, etc. Agents often act in a decentralized fashion on these platforms, and understanding the induced dynamics is an important step before designing policies around how to operate such platforms to maximize various system objectives such as revenue, efficiency and equity of allocations (Johari et al., 2021; Liu et al., 2020). A body of recent work is aimed at understanding the decentralized learning dynamics in such matching markets (Sankararaman et al., 2021; Liu et al., 2020; 2021; Dai & Jordan, 2021a;b; Basu et al., 2021). This line of work studies the matching markets introduced first by the seminal work of (Gale & Shapley, 1962), under the assumption where the participants are not aware of their preference and learn it over time by participating in the market. A key assumption made in these studies is that the true preferences of the market participants are static over time, and thus can be learnt with repeated interactions.

Markets, however are seldom stationary and continuously evolving. Indeed, an active area of research in management sciences and operations research revolve around understanding the equilibrium properties in such evolving markets (Damiano & Lam, 2005; Akbarpour et al., 2020; Kurino, 2020; Johari et al., 2021). However, a central premise in this line of work is that the participants have exact knowledge over their preferences, and only need to optimize over other agents' competitive behaviour and future changes that may occur. In this work, we take a step towards bridging the two aforementioned lines of work. To be precise, we study the learning dynamics in markets where both the participants do not know their exact preferences and the unknown preferences are themselves *smoothly varying* over time.

Conceptually, the seemingly simple addition of varying preferences invalidates the core premise of learning algorithms in a stationary environment (such as those in (Liu et al., 2021; Sankararaman et al., 2021)) where learning is guaranteed to get better with time as more samples can potentially be collected. In a dynamic environment, agents need to additionally trade-off collecting more samples by competing with other agents to have a refined estimate, with the possibility that the quantity to be estimated being stale and thus not meaningful.

Model Overview: The model we study consists of N agents and $k \geq N$ resources or arms, where the agents repeatedly make decisions of which arm to match with over a time horizon of T. The agents are globally ranked from 1 through N. The agents are initially assumed to not

[1]Systems and Control Engg. (SysCon) and Centre for Machine Intelligence and Data Science (CMInDS), IIT Bombay [2]AWS AI, Santa Clara, USA (Part of the work was done while affiliated with UC Berekely) [3]Electrical Engg. and Computer Sciences, UC Berkeley [4]Halıcıoğlu Data Science Institute (HDSI), UC San Diego [5]Electrical and Computer Engineering, UC San Diego. Correspondence to: Avishek Ghosh <avishek_ghosh@iitb.ac.in>.

Submitted to the ICML 2023 workshop on 'The Many Facets of Preference-based Learning'

know their rank. In each round, every agent chooses one of the k arms to match with. Every arm that has one or more agents requesting for a match, *allocates* itself to the highest ranking agent requesting a match[1], while *blocking* all other requesting agents. If at time t, agent j is matched to arm ℓ, then agent i sees a random reward independent of everything else with mean $\mu_{j,\ell,t}$. The agents that are blocked are notified of being blocked and receive 0 reward. Moreover the agents are decentralized, i.e., make decisions on which arm to match is a function of the history of the arms chosen, arms matched and rewards obtained at that agent.

The serial dictatorship model is as an important special case of the general matching market, with applications in pareto-optimal allocations that occur in cloud computing (Dickerson et al., 2019), (Even et al., 2009), crowd-sourcing (Massoulié & Xu, 2016) and question-answering platforms (Shah et al., 2020). In fact, in the classical application of matching universities to applicants (Gale and Shapley, 1962), a serial dictatorship model is reasonable since the university ranking is same for all the applicants. For stationary markets, decentralized algorithms was first studied in the serial dictatorship setting (Sankararaman et al., 2021; Basu et al., 2021), before being generalized to arbitrary markets (Liu et al., 2021). Thus in this context, our work contributes to the growing body of work on decentralized learning in matching markets, whereby we give first algorithms and results in the non-stationary setting under the serial dictatorship model.

The key departure from prior works of (Liu et al., 2020; 2021; Sankararaman et al., 2021) is that the unknown arm-means between any agent j and arm ℓ is time-varying, i.e., the mean is dependent on time t. We call our model *smoothly varying*, because we impose the constraint that for all agents j and arms ℓ, and time t, $|\mu_{j,\ell,t} - \mu_{j,\ell,t+1}| \leq \delta$, for some known parameter δ. However, we make no assumptions on the synchronicity of the markets, i.e., the environments of different agents can change arbitrarily with the only constraint that any arm-agent pair means does not change by more than δ in one time-step.

From a practical perspective, slow variation is a reasonable model in many settings of cloud computing ((Dickerson et al., 2019)) and financial applications, where the number of decisions are typically made in the order of seconds, while distribution changes/shifts typically occur at a larger time-scale of minutes or even hours. We show later that this model even with known δ, highlights the key tension in the multi-agent setting since the environments across agents varies asynchronously and designing decentralized algorithms is challenging and requires several technical novelty. Algorithms adaptive to general non-stationarity are unknown even for the single agent bandit problem (Krishnamurthy & Gopalan, 2021) and thus, we leave the general non-stationary market bandits to future work.

Decentralized decision making: The decision making by agents in our models are decentralized - i.e., at each time the decision of which arm to pull is based only on the information collected by that agent thus far. At each time after the pull of an arm, every agent either observes that they collided and thus get no reward, or receive a stochastic reward. In addition to rewards, we also assume that agents can write limited information on a 'black-board', which act as a public broadcast. In our model, we only assume that at each time-step, each agent can write one arm-id in $\{1, \cdots, k\}$ on the board. Such models of information sharing across agents is standard in decentralized matching markets - for ex. (Liu et al., 2021) assume that at the end of every time-step, every agent can observe the arms played by all agents in that time-step. In this regard, the model of information sharing we consider is weaker than that of (Liu et al., 2021). Furthermore, in Section D, we remove access to this black-board and show that low regret is achievable even if the only information agents have are the arms they play and the rewards they achieve.

Key technical challenges: Even in the single agent case without competitions, algorithms such as UCB (Auer et al., 2002) perform poorly compared to non-stationary bandit algorithms such as `SnoozeIT` (Krishnamurthy & Gopalan, 2021) that adapts to the varying arm means (c.f. Figure 2(a)) in smoothly varying environments. The reason is that stationary algorithms such as UCB weighs all the samples collected thus far equally in identifying which arm to pull, while adaptive algorithms such as `SnoozeIT` weighs recent samples more than older samples in order to estimate the arm-mean at the current time point. This is exacerbated in a multi-agent competitive setup where agents need to decide whether to pull an arm that yielded good results in the past, but is facing higher competition at the present.

We circumvent this problem by introducing the idea of *forced exploration* in the algorithm. Since the environments are time-varying possibly asynchronously across agents, a lower ranked agent may be forced to explore and obtain linear regret, if any of the higher ranked agents are exploring. To build intuition, consider a 2 agent system in which the higher ranked agent is called Agent 1, and the other agent is Agent 2. Suppose, Agent 1's environment (i.e., arm-means) are volatile where the gap between the best and second best arm is small, while Agent 2 has a more benign environment, where all arm-means are well separated and not varying with time. In this case, Agent 1 will be forced to explore arms a lot as its environment is fluctuating with no clear best arm emerging. Since any collision implies that Agent 2 will not receive a reward, Agent 2 is also forced to explore and play sub-optimal arms a linear number to times to evade collision, even when it knows its own best arms. This phenomenon indeed also occurs in the stationary setting, albeit in the

[1] If $i < j$, then agent with rank i is said to be higher ranked than agent j

stationary setting, every agent knows that after an initial exploration time, all agents will "settle" down and find their best arm. This is the concept of freezing time introduced in (Sankararaman et al., 2021; Basu et al., 2021), that plays a critical role in both the design and analysis of algorithms in the stationary case. In the non-stationary setting however, there is no single freezing time for the agents —rather agents must continuously switch between exploring and exploiting, as their environment varies with time.

2. Problem Setup

We consider the standard setup with N agents and k arms, with $k \geq N$. At time t, every agent $j \in [N]$ has a ranking of the arms, which is dictated by the arm means $\{\mu_{j,\ell,t}\}_{j\in[N],\ell\in[k]}$, and this ranking is not known to the agents. On the other hand, it is assumed that the agents are ranked homogeneously for all the arms, and the ranking is known to the arms. This is called the *serial dictatorship* model, is a well studied model in the market economy (see (Abdulkadiroğlu & Sönmez, 1998; Sankararaman et al., 2021)).

Without loss of generality, it is assumed that the rank of agent $j \in [N]$ is j. We say agent j is matched to arm ℓ at time t, if agent j pulls and receives (non zero) reward from arm ℓ. Our goal here is to find the unique stable matching (uniqueness ensured by the serial dictatorship model) between the agents and the arm side in a non-stationary (dynamic) environment. Our definition of stability is identical to that of the classical Gale-Shapley matching ((Gale & Shapley, 1962)); i.e., a matching is termed stable, if there exists no pair of agent-arm, who would mutually prefer each other as opposed to their current partners in the matching.

As explained in the introduction, we consider the smooth varying framework of (Wei & Srivatsva, 2018; Krishnamurthy & Gopalan, 2021) to model the non-stationary, which assumes $|\mu_{j,\ell,t+1} - \mu_{j,\ell,t}| \leq \delta$ for all t, j, k, and the maximum drift is δ. We dub this as a δ-shifted system.

We write $\ell_*^{(1,t)}$ as the arm preferred by the the Agent ranked 1 at time t, i.e., $\ell_*^{(1,t)} = \text{argmax}_{\ell\in[k]}\mu_{1,\ell,t}$. Similarly, for Agent ranked j, the preferred arm is given by $\ell_*^{(j,t)} = \text{argmax}_{\ell\in[k]\setminus\{\ell_*^{(1,t)},..,\ell_*^{(j-1,t)}\}}\mu_{j,\ell,t}$. So, we see that $(1, \ell_*^{(1,t)})$ forms a stable match, and so does $(j, \ell_*^{(j,t)})$ for $2 \leq j \leq N$. Let $L^{(j)}(t)$ be the arm played by an algorithm $\mathbb{A}$. The regret of agent j playing algorithm $\mathbb{A}$ upto time T is given by $R_j = \sum_{t=1}^{T} \mathbb{E}[\mu_{j,\ell_*^{(j,t)},t} - \mu_{j,L^{(j)}(t),t}\mathbf{1}_{M_{L^{(j)}(t)}=j}]$, where $M(.)$ indicates whether arm $L^{(j)}$ is matched.

3. NSCB with 2 agents

We now propose and analyze the algorithm, Non-Stationary Competing Bandits (NSCB) to handle the competitive nature of a market framework under a smoothly varying non-stationary environment. To understand the algorithm better, we first present the setup with 2 agents and k arms, and then in Section C, we generalize this to N agents.

We consider $N = 2$, since it is the simplest non-trivial setup to gain intuition about the complexity of the competitive nature of NSCB algorithm. Without loss of generality, assume that agent r has rank r, where $r \in \{1,2\}$. So, in the above setup, Agent 1 is the highest ranked agent.

3.1. Black Board model:

Moreover, to begin with and for simplicity, we assume a black-board model, and later in Section D, remove the necessity of this black board. We emphasize that black-board model of communication is quite standard in centralized multi-agent systems, with applications in game theory, distributed learning and auction applications (Awerbuch & Kleinberg, 2008; Buccapatnam et al., 2015; Agarwal et al., 2012). Through this black-board, the agents can communicate to one other. As we will see subsequently, Agents write critical information like the arm preferred by it, the duration of commitment to a particular arm etc to the blackboard, so that all the other agents can read the information and change their learning algorithm accordingly. Recall that, at each time-step, each agent can only write at-most one arm-id in $\{1,\cdots,k\}$ on the board. Such models of information sharing across agents is standard in decentralized matching markets - for ex. (Liu et al., 2021) assume that at the end of every time-step, every agent can observe the matched arms played by all agents in that time-step. Furthermore, in Section D, we remove access to this black-board and show that low regret is achievable even if the only information agents have are the arms they play and the rewards they achieve.

The learning algorithm is presented in Algorithm 1 and 2 for Agents 1 and 2 respectively. The algorithms run over several episodes indexed by i_1 and i_2 for Agents 1 and 2 respectively.

3.2. Algorithm for Agent 1

RANK ESTIMATION (): We let both agents pull arm 1 in the first time slot. Agent 1, will see a (non-zero) reward, and hence estimates its rank to be 1. The other agent, will see a 0 reward, so it estimates its rank as 2.

Since Agent 1 is highest ranked agent, it does not face any collision. It plays the well-known and standard Successive Elimination (SE) type algorithm (see (Slivkins, 2019)). The learning algorithm for Agent 1 bears resemblance with the SnoozeIT algorithm of (Krishnamurthy & Gopalan, 2021), except the fact that we have k arms in the system as opposed to 2 arms of SnoozeIT [2]. In a nutshell, the agent (a) first explores to identify if there is a *best* arm and (b) if it finds a best arm, it commits to that for some amount of time.

[2] In fact we obtain theoretical guarantees for SnoozeIT with k arms in Appendix G.1 which may be of independent interest.

Algorithm 1 NSCB with $N=2$; for Agent 1

Input: Horizon T, drift limit δ
Initialize set of tuples $S_1=\phi$, episode index $i_1 \leftarrow 1$
RANK ESTIMATION()
for $t=1,2,...,T$ **do**
  Pull-Arm by Agent 1:
  if $S_1=\phi$ **then**
    Play round-robbin (i.e., pull arm $t \% k$); set $z_t(1) \leftarrow$ Explore
  else
    Play arm x, such that $(x,s) \in S_1$ s.t. $s > t$; set $z_t(1) \leftarrow$ Exploit
  end if
  Test by Agent 1:
  if Arm a = Lambda-Opt $(\tilde{\lambda}, [k])$ (see Definition 1) **then**
    $\Lambda_{i_1} \leftarrow t - s_{i_1}$, buffer$_1 = \frac{8}{\delta}\sqrt{\frac{k \log T}{\Lambda_{i_1}}} - 2(k-1)$
    if buf$_1 > \Lambda_{i_1}$ **then**
      $S_1 \leftarrow S_1 \cup \{(a,\, s_{i_1} + \text{buffer}_1)\}$, Updates black-board with $(a, s_{i_1} + \text{buffer}_1)$
    end if
  else
    $i_1 \leftarrow i_1+1, s_{i_1} \leftarrow t$ (next epoch starts)
  end if
  Release arm by Agent 1:
  if $\exists (x,s) \in S_1 : s \le t$ **then**
    $S_1 \leftarrow S_1 \setminus (x,s)$, release arm x, $i_1 \leftarrow i_1+1, s_{i_1} \leftarrow t$
  end if
end for

Note that with non-stationary environment, Agent 1 needs to repeat this procedure over time.

Let us set up a few notation. We denote $z_t(1)$ to denote the phase of Agent 1, and $z_t(1) \in \{\texttt{Explore}, \texttt{Exploit}\}$. In Algorithm 1, we we use $\{s_{i_1}\}_{i_1=1,2,..}$ to denote the starting of epochs, and i_1 as the index count. At time t, Agent 1 checks whether there exists an *optimal* arm by the following test.

3.2.1. Test for Optimality

Let $\hat{\mu}_{a,t}(\tilde{w})$ denote the empirical reward mean of arm a at time t, based on its last $\tilde{w}$ pulls. We now define the rest bases on which our learning algorithms decide whether to commit on an arm or not.

Definition 3.1. (Lambda-Opt $(\tilde{\lambda}, \mathcal{A})$) At time t, an arm a is said to be Lambda-Opt $(\tilde{\lambda}, \mathcal{A})$ with respect to set $\mathcal{A}$, if there exists $\tilde{\lambda} \in (0,1)$ such that $\hat{\mu}_{a,t}(\tilde{w}) > \hat{\mu}_{b,t}(\tilde{w}) + 4r(\tilde{w}) - (k-1)\delta$, for all $b \in \mathcal{A} \setminus \{a\}$, where $\tilde{w} = \frac{c_1 \log T}{\tilde{\lambda}^2}$, and $r(\tilde{w}) = \sqrt{\frac{2 \log T}{\tilde{w}}}$.

Intuitively, the above test decides whether the lower confidence interval of arm a is bigger than that of the upper confidence interval of arm b with some additional slack. Since the mean rewards are bounded in $[0,1]$, if the above test succeeds, it can be shown (see Lemma G.3) that learning algorithms can commit on arm a and incur 0 regret for some time, determined by the buffer length.

Since Agent 1 faces no competition, $\mathcal{A} = [k]$ (the set of all arms), but $\mathcal{A}$ will be different for Agent 2, as we will see shortly. In Algorithm 1, we denote Λ_{i_1} as the duration of the exploration period before the test succeeds at episode i_1. After the test, the agent exploits the obtained best arm for (buffer$_1 - \Lambda_{i_1}$) time, and then releases it. We define the set S_1 to determine whether Agent 1 should commit or continue exploring.

In Figure 1, we consider one episode of Agent 1, where the yellow segments indicate the exploration time, and at the end of that, the purple segment indicates the commit (exploitation) (to say arm i^*) time. Furthermore, when Agent 1 commits, it writes the arm on which it is committing and the duration of the commit to the black-board, so that Agent 2 can accordingly choose actions from a restricted set of arms to avoid collision. Note that, there is no competition here, and the (interesting) market aspect is absent.

3.3. Algorithm for Agent 2

The actions of Agent 2 borne out the competition (market) aspect of the problem, and is written formally in Algorithm 2. To fix notation, we denote $\{t_{i_2}\}_{i_2=1,2,..}$ as the time instances where an epoch starts for Agent 2, and i_2 denotes the episode index. Moreover, we use the notation $z_t(2)$ to denote the state of Agent 2, and as explained in Algorithm 2, $z_t(2) \in \{\texttt{Explore ALL}, \texttt{Explore} - j, \texttt{Exploit}(x)\}$, where the terms are explained shortly.

We observe from Algorithm 2 ensures that Agent 2 won't collide with Agent 1. This is because, in case of collision, Agent 2 will receive a deterministic zero reward, and thus will encounter maximum instantaneous regret.

Hence, Agent 2 gets to explore all the k arms, and plays in a round robbin fashion only when Agent is also exploring (and has not committed yet). This is called the Explore ALL phase. This is shown in light green in Figure 1, and line 5 of Algorithm 2.

On the other hand, if Agent 1 has committed to an arm, say the j-th arm, it it of best interest for Agent 2 to explore within the set $[k] \setminus \{j\}$. Otherwise, it will encounter collision once in ever k rounds. This is called the Explore-j phase of Agent 2. This is shown in dark-green in Figure 1 and lines 7-8 of Algorithm 2. In the figure, $j = i^*$.

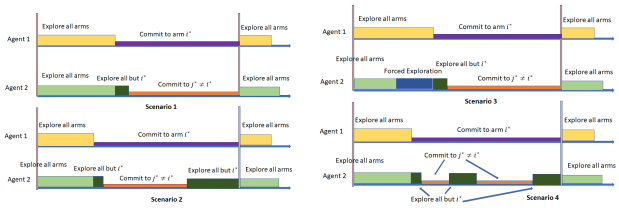
Apart from Explore ALL and Explore-j, Agent 2 also gets to commit or Exploit an arm. Observe that Agent 2 gets only gets to commit when Agent 1 has committed already (to an arm j)–otherwise it will face collision. Agent

Algorithm 2 NSCB with $N=2$; for Agent 2

1: Initialize set of tuples $S_2^{(j)}=\phi, \forall\, j\in[k]$, episode index $i_2\leftarrow 1$
2: **for** $t=1,2,...,T$ **do**
3: **Pull-Arm by Agent 2:**
4: **if** Agent 1 is not committed **then**
5: Play round robbin on $[k]$ (pull arm $t+1\,\%\,k$), set $z_t(2)\leftarrow$ Explore ALL
6: **else if** Agent 1 is committed to arm j and $S_2^{(j)}=\phi$ **then**
7: Play round robbin on $[k]\setminus\{j\}$ (i.e., pull arm index $t\%(k-1)$-th smallest arm id in $[k]\setminus\{j\}$)
8: $z_t(2)\leftarrow$ Explore-j
9: **else if** Agent 1 is committed to arm j, and $\exists(x,s)\in S_2^{(j)}$ s.t. $s>t$ **then**
10: Play arm x; set $z_t(2)\leftarrow$ Exploit
11: **end if**
12: **if** $\{z_t(1)\neq z_{t-1}(1)\}$ OR $\{z_t(2)\neq z_{t-1}(2)\}$ **then**
13: $i_2\leftarrow i_2+1$, $t_{i_2}\leftarrow t$
14: **end if**
15: **Test by Agent 2:**
16: **for** $j\in[k]$ s.t. $z_t(2)\in$ {Explore-j, Explore ALL} **do**
17: **if** Arm a = Lambda-Opt $(\tilde{\lambda},[k]\setminus\{j\})$ **then**
18: $\tau_{i_2}^{(j)}\leftarrow t-t_{i_2}$, buffer$_2=\frac{8}{\delta}\sqrt{(k-1)\log T/\tau_{i_2}^{(j)}}-2(k-2)$
19: **if** buffer$_2>\tau_{i_2}^{(j)}$ **then**
20: $S_2^{(j)}\leftarrow S_2^{(j)}\cup\{(a,\min\{t_{i_2}+\text{buffer}_2, s_{i_1+1}\})\}$
21: **end if**
22: **end if**
23: **end for**
24: **Release arms for Agent 2:**
25: **for** $j\in[k]$ **do**
26: **if** $\exists(x,s)\in S_2^{(j)}: s\leq t$ **then**, $S_2^{(j)}\leftarrow S_2^{(j)}\setminus(x,s)$
27: **end for**
28: **end for**

2 tests whether an arm is best for it by the Lambda-Opt $(\tilde{\lambda},\mathcal{A})$ test with $\mathcal{A}=[k]\setminus\{j\}$. If such an arm exists, then Agent 2 gets to commit to it. Similar to Agent 1, we define t_{i_2} as the time instance when the Lambda-Opt $(\tilde{\lambda},\mathcal{A})$ test succeeds for Agent 2 and define $\tau_{i_2}^{(j)}$ as the duration of the test. Since Agent 2 tests with $k-1$ arms, the buffer$_2$ is set accordingly.

From Line 20 of Algorithm 2, note that Agent 2 only gets to commit to the 'optimal' arm until $\min\{t_{i_2}+\text{buffer}_2, s_{i_1+1}\}$. Hence, we restrict Agent 2 to end its exploitation as soon as the exploitation of Agent 1 ends. The reasoning is same—Agent 1 starts exploring right after its exploitation and Agent 2 must release the arm it was exploiting to avoid collision. Also, observe that since Agent 2 never updates the black board. This is because Agent 2 is the lowest rank agent in the system, and the action of Agent 1 is not influenced by Agent 2.



((a)) ((b))

Figure 1: Action of Agents 1 and 2 in a matching markets

From line 20 of Algorithm 2, Agent 2 constructs the sets $S_2^{(j)}$, which denote the exploitation period of Agent 2, without arm j in the system. This is used to represent the different phases of Agent 2 in Algorithm 2 in a compact form.

Saving extra exploration: Note that Agent 2 continues to test for an optimal arm even when Agent 1 is exploring. It might seem to be wasteful at first since it cannot commit immediately. This is useful because, Agent 2 constructs the sets $S_2^{(j)}$, and as soon as Agent 1 commits to arm j with $S_2^{(j)}$ being non-empty, Agent 2 gets to commit leveraging this test. This saves extra exploration for Agent 2 and hence reduces regret.

From the above description, we characterize the fundamental reason, for which Agent 2 might suffer additional regret, which we now discuss.

Forced Exploration: Consider Scenario 3 of Figure 1(b). Here, Agent 2 has decided to commit on an arm before Agent 1. However, it cannot start to exploit since Agent 1 is still exploring. Otherwise, it will periodically face collisions (and get 0 reward, hence incurring linear regret in this duration). This is the additional exploration faced by Agent 2, which we term as *forced exploration* (shown in blue in Figure 1(b)). In Theorem 4.5, we characterize the regret stems from this forced exploration.

Furthermore, as shown in Figure 1 and discussed in Algorithm 2 (line 20), Agent 2 is forced to terminate its exploitation and start exploration, as soon as the exploitation of Agent 1 ends. Note that this also results in higher regret of Agent 2, as it does not get to *fully exploit* the arm it was committed to.

In the subsequent section, we characterize the price paid by Agent 2 because of the above-mentioned restrictions.

4. Theoretical Guarantees for NSCB with $N=2$

We first define the dynamic gap of the problem.

4.1. Problem Complexity—Dynamic Gap

We define the (dynamic) gap, denoted by $\{\lambda_t^{\mathcal{C}}[r]\}_{t=1,2,..}$ for agent r, which determines how complex the problem is.

Definition 4.1. For $\mathcal{C} \subseteq [k]$, we define the dynamic gap of Agent r on a dominated set $\mathcal{C}$ as,

$$\lambda_t^{\mathcal{C}}[r] = \max_{\lambda \in [0,1]} \{ \min_{\substack{a,b \in [k] \setminus \mathcal{C} \\ a \neq b}} \frac{1}{w(\lambda)} |\sum_{t'=s}^{t} \mu_{r,a,t'} - \mu_{r,b,t'}| \geq \lambda \},$$

and if such a λ does not exist, we set $\lambda_t^{\mathcal{C}}[r] = c_1 \sqrt{\frac{\log T}{t}}$. Here, $s = t - w(\lambda) + 1$, and $w(\lambda) = \frac{c_0 (k - |\mathcal{C}|) \log T}{\lambda^2}$. For shorthand, if $\mathcal{C} = \phi$ (null set), we denote $\lambda_t^{\phi}[r] = \lambda_t[r]$. Here c_1 and c_0 are universal constants.

Remark 4.2. We first note that when $\delta = 0$ (stationary system) and $\mathcal{C} = \phi$, the dynamic gap reduces to the gap between the best and the second best arm. Note that the latter is the usual definition of problem complexity in multi-armed bandit problems ((Lattimore & Szepesvári, 2020)).

Remark 4.3. When $\mathcal{C} = \phi$ and $k = 2$, the above definition matches exactly to that of 'detectable gap' of (Krishnamurthy & Gopalan, 2021).

Remark 4.4. The *dominated* dynamic gap is a strict generalization of the usual window based average gap used in non-stationary bandits. We introduce a dominated set $\mathcal{C}$, for the competitive market setting, since the actions of lower ranked agents are dominated by that of higher ranked ones.

4.2. Regret Guarantee

We now characterize the regret of Agent 1 and 2 playing Algorithms 1 and 2 respectively.

Theorem 4.5 (2 Agent `NSCB`)**.** *Suppose we run Algorithm 2 with 2 Agents upto horizon T with drift δ. Then the expected regret for Agent 1 is $R_1(\delta) \leq C \sum_{\ell=1}^{m} \frac{1}{\lambda_{\min,\ell}[1]} k \log T$, and for Agent 2 is*

$$R_2(\delta) \leq C_1 \sum_{\ell=1}^{m} \Bigg\{ \left(\frac{1}{\lambda_{\min,\ell}[2]} + \frac{1}{(\lambda_{\min,\ell}[1])^2} \right) k \log T + \lceil (\frac{k}{k-1})^{1/3} \rceil \left(\frac{1}{\min_{a \in [k]} \lambda_{\min,\ell}^{\{a\}}[2]} \right) (k-1) \log T \Bigg\},$$

where horizon T is divided into m blocks, each having length at most $\min\{c\delta^{-2/3}k^{1/3}\log^{1/3} T, T\}$. Here

$$\lambda_{\min,\ell}[r] = \min_{t \in \ell\text{-th block}} \lambda_t[r] \text{ and } \lambda_{\min,\ell}^{\{a\}}[r] = \min_{t \in \ell\text{-th block}} \lambda_t^{\{a\}}[r]$$

denote the dynamic gap of Agent r over entire ℓ-th block.

Remark 4.6. Similar to (Krishnamurthy & Gopalan, 2021), we define the quantity $\lambda_{\min,\ell}^{\mathcal{C}}[r]$, which is the minimum dynamic gap of agent r over the ℓ-th block. When $\mathcal{C} = \phi$, we denote this by $\lambda_{\min,\ell}[r]$.

4.3. Discussion

Regret of Agent 1: The regret of Agent 1 is given by $\frac{k \log T}{\lambda_{\min,\ell}[1]}$, summed over blocks, where $\lambda_{\min,\ell}[1]$ is the complexity parameter (dynamic gap) for Agent 1. In the special case, where $\delta = 0$ (stationary system), the number of blocks is 1, and the complexity parameter coincides with the the definition of classical gap (difference between the best and the second best arm) of the stationary setup. In this case, this regret bound matches to that of the classical instance dependent regret of Upper Confidence Bound (UCB) algorithm of (Auer et al., 2002).

Regret of Agent 1 matches (Krishnamurthy & Gopalan, 2021): Observe that the regret of Agent 1 matches exactly to `Snooze-IT` of (Krishnamurthy & Gopalan, 2021). Since Agent 1 faces no collision, we were able to recover this.

Regret of Agent 2: The regret of Agent 2 has 3 components. The first term, $\frac{k \log T}{\lambda_{\min,\ell}[2]}$ comes from the `Explore-ALL`, where Agent 2 explores all arms and the regret is similar to Agent 1. This also depends on the complexity of Agent 2.

The second term in the regret expression, $[\frac{1}{\lambda_{\min,\ell}[1]}]^2 k \log T$ originates from the `Forced Exploration` of Agent 2. Note that this depends on the complexity (gap) of Agent 1. This validates our intuition, since, when Agent 1's environment is complex, it takes more exploration for Agent 1, and as a result Agent 2 faces additional forced exploration. This is a manifestation of the competitive structure of the market framework, since the regret of Agent 2 is influenced by that of higher ranked agent.

The third term in the regret expression comes from `Explore-j` phase, where Agent 1 is committed on arm j. Observe that here, the dominated gap naturally comes into the picture. The pre-factor of $\lceil k/(k-1) \rceil^{1/3}$ appears for the following reason. We design the blocks in such a way that each block contains at most 2 phases of Agent 1. Moreover, we show that the number of epochs for Agent 2 in one exploitation phase of Agent 1 is at most $2\lceil \lceil k/(k-1) \rceil^{1/3} \rceil$.

Regret matches to UCB-D3 of (Sankararaman et al., 2021) in stationary setup: We compare the regret of `NSCB` with that of the non-stationary UCB-D3 of (Sankararaman et al., 2021). In the stationary environment ($\delta = 0$), the definition of gap is invariant with time. For Agent 2, from (Sankararaman et al., 2021, Corollary 2), we obtain the regret to be $\mathcal{O}[\frac{1}{\rho^2}(k-1)\log T]$, where ρ is the stationary dominated gap. Note that this is exactly same as Theorem 4.5 (except for a mildly worse dependence on k). Hence, we recover the order-wise optimal regret in the stationary setting.

5. `NSCB` with N agents, Blackboard Removal and Experiments

Owing to space crunch, we defer these sections to the Appendices C,I and E respectively. We extend `NSCB` to N agents and obtain regret bound for r-th ranked agent. We also discuss the market setup without blackboard and characterize the price (in terms of regret) for it.

References

Abdulkadiroğlu, A. and Sönmez, T. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–701, 1998.

Agarwal, A., Bartlett, P. L., Ravikumar, P., and Wainwright, M. J. Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization. *IEEE Transactions on Information Theory*, 58(5):3235–3249, 2012.

Akbarpour, M., Li, S., and Gharan, S. O. Thickness and information in dynamic matching markets. *Journal of Political Economy*, 128(3):783–815, 2020.

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

Auer, P., Gajane, P., and Ortner, R. Adaptively tracking the best bandit arm with an unknown number of distribution changes. In Beygelzimer, A. and Hsu, D. (eds.), *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pp. 138–158. PMLR, 25–28 Jun 2019. URL https://proceedings.mlr.press/v99/auer19a.html.

Awerbuch, B. and Kleinberg, R. Competitive collaborative learning. *Journal of Computer and System Sciences*, 74 (8):1271–1288, 2008.

Basu, S., Sankararaman, K. A., and Sankararaman, A. Beyond $log^2(t)$ regret for decentralized bandits in matching markets. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 705–715. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/basu21a.html.

Besbes, O., Gur, Y., and Zeevi, A. Stochastic multi-armed-bandit problem with non-stationary rewards. *Advances in neural information processing systems*, 27:199–207, 2014.

Buccapatnam, S., Tan, J., and Zhang, L. Information sharing in distributed stochastic bandits. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 2605–2613. IEEE, 2015.

Dai, X. and Jordan, M. I. Learning strategies in decentralized matching markets under uncertain preferences. *Journal of Machine Learning Research*, 22(260):1–50, 2021a.

Dai, X. and Jordan, M. I. Multi-stage decentralized matching markets: Uncertain preferences and strategic behaviors. *arXiv preprint arXiv:2102.06988*, 2021b.

Damiano, E. and Lam, R. Stability in dynamic matching markets. *Games and Economic Behavior*, 52(1):34–53, 2005.

Dickerson, J., Sankararaman, K., Sarpatwar, K., Srinivasan, A., Wu, K.-L., and Xu, P. Online resource allocation with matching constraints. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2019.

Even, G., Halldórsson, M. M., Kaplan, L., and Ron, D. Scheduling with conflicts: online and offline algorithms. *Journal of scheduling*, 12(2):199–224, 2009.

Gale, D. and Shapley, L. S. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

Garivier, A. and Moulines, E. On upper-confidence bound policies for switching bandit problems. In *International Conference on Algorithmic Learning Theory*, pp. 174–188. Springer, 2011.

Johari, R., Kamble, V., and Kanoria, Y. Matching while learning. *Operations Research*, 69(2):655–681, 2021.

Karnin, Z. S. and Anava, O. Multi-armed bandits: Competing with optimal sequences. *Advances in Neural Information Processing Systems*, 29:199–207, 2016.

Knuth, D. E. *Stable marriage and its relation to other combinatorial problems: An introduction to the mathematical analysis of algorithms*, volume 10. American Mathematical Soc., 1997.

Krishnamurthy, R. and Gopalan, A. On slowly-varying non-stationary bandits. *arXiv preprint arXiv:2110.12916*, 2021.

Kurino, M. Credibility, efficiency, and stability: A theory of dynamic matching markets. *The Japanese Economic Review*, 71(1):135–165, 2020.

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

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

Liu, L. T., Mania, H., and Jordan, M. Competing bandits in matching markets. In *International Conference on Artificial Intelligence and Statistics*, pp. 1618–1628. PMLR, 2020.

Liu, L. T., Ruan, F., Mania, H., and Jordan, M. I. Bandit learning in decentralized matching markets. *Journal of Machine Learning Research*, 22(211):1–34, 2021.

Luo, H., Wei, C.-Y., Agarwal, A., and Langford, J. Efficient contextual bandits in non-stationary worlds. In *Conference On Learning Theory*, pp. 1739–1776. PMLR, 2018.

Massoulié, L. and Xu, K. On the capacity of information processing systems. In *Conference on Learning Theory*, pp. 1292–1297. PMLR, 2016.

Pittel, B. The average number of stable matchings. *SIAM Journal on Discrete Mathematics*, 2(4):530–549, 1989.

Roth, A. E. and Vate, J. H. V. Random paths to stability in two-sided matching. *Econometrica: Journal of the Econometric Society*, pp. 1475–1480, 1990.

Sankararaman, A., Basu, S., and Sankararaman, K. A. Dominate or delete: Decentralized competing bandits in serial dictatorship. In *International Conference on Artificial Intelligence and Statistics*, pp. 1252–1260. PMLR, 2021.

Shah, V., Gulikers, L., Massoulié, L., and Vojnović, M. Adaptive matching for expert systems with uncertain task types. *Operations Research*, 68(5):1403–1424, 2020.

Slivkins, A. Introduction to multi-armed bandits. *arXiv preprint arXiv:1904.07272*, 2019.

Slivkins, A. and Upfal, E. Adapting to a changing environment: the brownian restless bandits. 2008.

Wei, L. and Srivatsva, V. On abruptly-changing and slowly-varying multiarmed bandit problems. In *2018 Annual American Control Conference (ACC)*, pp. 6291–6296. IEEE, 2018.

Whittle, P. Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, 25(A): 287–298, 1988.

Appendix

A. Related work

A.1. Bandits and Matching Markets

Bandits and matching markets have received a lot of attention lately, owing to both their mathematical non-triviality and the enormous practical impact they hold. Regret minimization in matching markets was first introduced in (Liu et al., 2020) which studied the much simpler problem of stationary markets under a centralized scheme, where a central entity matches agents with arms at each time. They showed that under this policy, a learning algorithm can get per-agent regret scaling as $\mathcal{O}(\log(T))$. Subsequently, (Sankararaman et al., 2021) studied the decentralized version of the problem under the serial dictatorship and proposed the `UCB-D3` algorithm that achieved $\mathcal{O}(\log(T))$ per-agent regret. Subsequently, (Liu et al., 2021) proposed `CA-UCB`, a fully decentralized algorithm that could achieve $\mathcal{O}(\log^2(T))$ per-agent regret in the general decentralized stationary markets. Matching markets has been an active area of study in combinatorics and theoretical computer science due to the algebraic structures they present (Pittel, 1989; Roth & Vate, 1990; Knuth, 1997). However, these works consider the equilibrium structure and not the learning dynamics induced when participants do not know their preferences.

A.2. Non-Stationary Bandits

The framework on non stationary bandits were introduced in (Whittle, 1988) in the framework of restless bandits, and later improved by (Slivkins & Upfal, 2008). There has been a line of interesting work in this domain–for example in (Garivier & Moulines, 2011; Auer et al., 2019; Liu et al., 2018) the abruptly changing or switching setup is analyzed, where the arm distributions are piecewise stationary and an abrupt change may happen from time to time. In particular (Liu et al., 2018) proposes a change point based detection algorithm to identify whether an arm distribution has changes of not in a piecewise stationary environment. Furthermore, in (Besbes et al., 2014), a total variation budgeted setting is considered, where the total amount of (temporal) variation is known, but the change may happen, either smoothly or abruptly.

Moreover, in the above-mentioned total variation budget based non-stationary framework, an adaptive algorithm, that does not require the knowledge of the drift parameter is obtained in (Karnin & Anava, 2016) for the standard bandit problem and later extended to (Luo et al., 2018) for the contextual bandit setup.

On the other hand, there are a different line of research that focuses on the *smoothly* varying non-stationary environment, in contrast to the above mentioned abrupt or total budgeted setup, for example see (Wei & Srivatsva, 2018; Krishnamurthy & Gopalan, 2021). Note that (Wei & Srivatsva, 2018) modify the sliding window UCB algorithm of (Garivier & Moulines, 2011) and employ windows of growing size. On the other hand, very recently (Krishnamurthy & Gopalan, 2021) analyzed the *smoothly* varying framework by designing windows of dynamic length and test for optimality within a sliding window. The algorithm of (Krishnamurthy & Gopalan, 2021), namely `Snooze-IT`, is an asynchronous algorithm that works on repeated Explore and Commit (ETC) type principle where the explore and commit times are random.

In this paper, we work with the smoothly varying non-stationary framework of (Krishnamurthy & Gopalan, 2021). We choose this algorithm because of its simplicity, and the dynamics and competition that comes out of a market framework is better understood in such a sliding window based Explore and Commit type algorithm. In general, we believe that our basic principle can be adapted to any sliding window based algorithm in a non-stationary environment.

Notation: For a positive integer r, we denote the set $\{1,2,...,r\}$ by $[r]$. Moreover, For 2 integers, a,b, the notation $a\%b$ implies the remainder (modulo) operation. Throughout the paper, we use $C,C_1,C_2,...,c,c_1,c_2,...$ as universal constants, the value of which may change from instance to instance.

B. Our Contributions

B.0.1. Algorithms

We introduce a learning algorithm, `NSCB`, in which agents proceed in phases with asynchronous start and end-points, wherein in each phase, agents explore among those arms that are not currently preferred by higher-ranked agents, and subsequently exploit a good arm, for a dynamic duration of time in which the estimated best arm can remain to be optimal. The main algorithmic innovation is to identify that the static synchronous arm-deletion strategy of UCB-D3 (Sankararaman et al., 2021),

can be coupled with SnoozeIT to yield a dynamic, asynchronous explore-exploit type algorithm for non-stationary bandits.

B.0.2. TECHNICAL NOVELTY IN THE ANALYSIS

In order to analyze and prove that NSCB yields goo d regret guarantees, we introduced this notion of *forced exploration*. Roughly speaking, this is the regret incurred due to exploration of an agent, when the higher ranked agents are exploring. This extra regret is a consequence of the rank ordering via the serial-dictatorship (which we define in Section **??**) model, whereby agents can incur collision and do not get any reward. Although agents in the stationary setting also incur forced exploration, its effect is bounded since every agent can eventually guarantee that the best arm can be learnt. However, in an asynchronously varying environment, bounding this term is non-trivial. We circumvent this by decomposing the forced exploration of an agent recursively; an agent ranked r *effectively explores* if either its own environment is fluctuating and thus hard to identify its best arm, or if the agent ranked $r-1$ is *effectively exploring*. We leverage this to recursively bound the regret of agent ranked r as a function of agent ranked $r-1$. Unravelling this recursion yields the final regret. In the process, we also derive new regret bounds for the general Snooze-IT algorithm of (Krishnamurthy & Gopalan, 2021) for the general k armed bandit that generalizes the 2 armed bound obtained in (Krishnamurthy & Gopalan, 2021), which may be of independent interest.

B.0.3. SUPERIOR EMPIRICAL PERFORMANCE

We empirically validate our algorithms to demonstrate that it *(i)* is simple to implement, *(ii)* the results match the theoretical insights, such as agents incurring additional regret due to forced explorations and *(iii)* outperforms prior state of art UCB D3 (Sankararaman et al., 2021) in both stationary and non-stationary environments. The last point is compelling and demonstrates that our algorithm is strictly more general than UCB-D3 by being provably and empirically superior to UCB-D3 in non-stationary environment, and being empirically superior to UCB-D3 in stationary environments. Theoretically, we show that the regret bounds of our algorithm matches order-wise that of UCB-D3 in the stationary environment (cf. Sections 4.3 and C.2).

C. NSCB Algorithm with N competing agents

In this section, we extend NSCB for N agents. We stick to the notation where we denote the r-th ranked Agent as Agent r and focus on its the learning algorithm. Let us fix some notation first.

We denote $\mathcal{C}_t(r)$ as the set of committed arms by agents ranked higher that r (i.e., Agents $1,2,\dots,r-1$). This can be defined sequentially in the following way: let $C_t(1) \in \{\phi,1,\dots,k\}$ as the arm committed by Agent 1. We define $\mathcal{C}_t(r)=\{C_t(1),\dots,C_t(r-1)\}$ as the committed (or dominated) set of Agent r. The learning scheme is presented in Algorithm 3

Similar to Algorithms 1 and 2, we start with the rank estimation sub-routine, the end of which agents know their own rank.

RANK ESTIMATION() The rank estimation takes $N-1$ time steps. At $t=1$, all agents pull arm 1. In the subsequent steps, i.e., for $t\in[2,N-1]$, agents, the agents who were matched to an arm, continues to play the matched arm–and the rest of the agents play arm (indexed by) t. By inductive reasoning, thanks to the collision structure, it is easy to observe that, by $N-1$ time instant, all agents know their own rank.

In Algorithm 3, we denote $\{t_{i_r}\}_{i_r=1,2,..}$ as the start epochs of episodes for Agent r. Furthermore, to denote the state of Agent r we define $z_t(r)$ as the following:

$$z_t(r)=\{\texttt{Explore}-\mathcal{C}_t(r),\texttt{Exploit}(x)\},$$

where in $\texttt{Explore}-\mathcal{C}_t(r)$, the r-th agent plays in a round robbin fashion on the set of $[k]\setminus\mathcal{C}_t(r)$ arms (line 18 of Algorithm 3) , and in $\texttt{Exploit}(x)$ it pulls arm indexed by x (line 20).

At time t, Agent r first looks at the black-board, and using the information, it constructs the dominated set $\mathcal{C}_t(r)$, which contains all the committed arms from Agents 1 to $r-1$. Based on $\mathcal{C}_t(r)$, Agent r updates $z_t(r)$ to reflect whether it is in $\texttt{Explore-}\mathcal{C}_t(r)$ phase, or in the exploit phase. In particular, Agent r gets to commit on an arm in $[k]\setminus\mathcal{C}_t(r)$, if all the higher ranked agents have already committed, i.e., $|\mathcal{C}_t(r)|=r-1$. A new phase is spawned for Agent r if either the dominated set $\mathcal{C}_t(r)\neq\mathcal{C}_t(r-1)$ changes, or its own phase ends. Both this cases are captured by $z_t(r)$, and hence, based on whether $z_t(r)$ changes or not, Agent r decides to start a new phase.

The test procedure of Agent r is similar to that of Agent 2, with a difference that Agent r tests in the arms in the subset $[k]\setminus\mathcal{C}_t(r)$, and hence the buffer length is accordingly designed. Once the Agent commits on an arm (say x), it writes the triplet (x,exploit period,r) to the black board. It also updates the dominated set $\mathcal{C}_{t+1}(r)$ for the next round.

Algorithm 3 NSCB for r-th Agent

Input: Horizon T, drift limit δ
Initialize $S_r^{(\Omega)} = \phi$ for all $\Omega \subseteq [k]$, and $|\Omega| \leq r-1$, Initialize $i_r \leftarrow 1$, $\mathcal{C}_1(r) = \phi$
RANK ESTIMATION()
for $t = 1,2,...,T$ **do**
  Update State $z_t(r)$**:**
  if $|\mathcal{C}_t(r)| < r-1$ **then**
    $z_t(r) \leftarrow$ Explore$-\mathcal{C}_t(r)$
  else if $\exists (x,s) \in S_r^{(\mathcal{C}_t(r))}$ s.t. $s > t$, **then**
    $z_t(r) \leftarrow$ Exploit(x)
  else
    $z_t(r) \leftarrow$ Explore$-\mathcal{C}_t(r)$
  end if
  if $z_t(r) \neq z_{t-1}(r)$ **then**
    $i_r \leftarrow i_r + 1, t_{i_r} \leftarrow t$
  end if
  Pull-Arm by Agent r:
  if $z_t =$ Explore$-\mathcal{C}_t(r)$ **then**
    Play round robbin with $[k] \setminus \mathcal{C}_t(r)$ (i.e., pull $t + (r - |\mathcal{C}_t(r)| - 1)\%(k - |\mathcal{C}_t(r)|)$ smallest arm in $[k] \setminus \mathcal{C}_t(r)$)
  else if $z_t =$ Exploit(x) **then**
    Play arm x
  end if
  Test by Agent r**:**
  for $\Omega \subseteq [k]$ s.t. $|\Omega| = r-1$ and $z_t(r) =$ Explore-$\mathcal{C}_t(r)$ **do**
    if Arm a = Lambda-Opt $(\tilde{\lambda}, [k] \setminus \Omega)$ **then**
      $\tau_{i_r}^{(\Omega)} \leftarrow t - t_{i_r}$, buffer$_r = \frac{8}{\delta}\sqrt{(k - |\mathcal{C}_t(r)|)\frac{\log T}{\tau_{i_r}^{(\Omega)}}} - 2(k-r)$, define $\bar{t}_{i_r} = \min\{t_{i_r} + \text{buffer}_{i_r}, t_{i_{r-1}+1}\}$
        if buffer$_{i_r} > \tau_{i_r}$, **then** $S_r^{\Omega} \leftarrow S_r^{\Omega} \cup \{(a, \bar{t}_{i_r})\}$, **else** $i_{i_r} \leftarrow i_r + 1, t_{i_r} \leftarrow t$
        end if
    end if
  end for
  Updates Black Board:
  if $\exists (x,s)$ s.t. $s \geq t+1$, **then** write $(x, \bar{t}_{i_r}, r)$ on the black board **end if**
  Updates Dominated set $\mathcal{C}_{t+1}(r)$**:**
  Updates $\mathcal{C}_{t+1}(r) = \{x \in [k] : \exists s > t+1, \text{and} \exists j \leq r-1 s.t. (x,s,j) \text{exists on board}\}$
  end if
end for

Similar to Algorithm 2, we also need to ensure that Agent r ends its exploitation phase when any higher ranked agent starts exploring, otherwise it will face collision. This is ensured by defining $\bar{t}_{i_r}$ in line 25 of Algorithm 3.

We emphasize again that Agent r only gets to commit when $|\mathcal{C}_t(r)| = r-1$, i.e., Agents 1 through $r-1$ has already committed. This leads to additional exploration, which we call **forced exploration**. In the subsequent section, we characterize the additional regret incurred by this forced exploration.

Saving extra exploration: Note that Agent r constructs the sets S_r^{Ω} for all $\Omega \subseteq [k]$ with $|\Omega| = r-1$. As explained in the 2 agent case, this saves extra exploration for Agent r, because if the statistical test succeeds on an arm $j \in [k] \setminus \mathcal{C}_t(r)$, and there exists Ω, with $|\Omega| = r-1$ such that S_r^{Ω} is non-empty, Agent r immediately commits to arm j.

C.1. Regret Guarantee

We characterize the regret of r-th agent, with $r \geq 2$. Note that the regret of Agent 1 will be identical as Theorem 4.5, since it faces no competition and hence no collision.

As explained in Algorithm 3, the regret of r-th Agent will depend on the dynamic gap of Agents 1 to $r-1$. Hence, for mathematical tractability and notational convenience, we define the following constrained dynamic gap of the problem.

$$\Delta_t[r] = \min_{\mathcal{C}\in[k],|\mathcal{C}|\leq r-2} \lambda_t^{\mathcal{C}}[r], \quad \tilde{\Delta}_t[r] = \min_{\mathcal{C}\in[k],|\mathcal{C}|\leq r-1} \lambda_t^{\mathcal{C}}[r]$$

Remark C.1. Note that the definitions of $\Delta_t[r]$ and $\tilde{\Delta}_t[r]$ are generalizations of the dynamic gap $\lambda_t^{\mathcal{C}}[r]$, with further restrictions on the dominated set $\mathcal{C}$.

With this, we have the following result:

Theorem C.2 (N agent NSCB)**.** *Suppose we run Algorithm 3 for N agents with δ drift. The regret for r-th ranked agent is given by*

$$R_r(\delta) \leq C\sum_{\ell=1}^{m}[\frac{k}{k-r+2}]^{1/3}\Bigg\{\left[\frac{k\log T}{\Delta_{\min,\ell}[r]}+\frac{k\log T}{\Delta^2_{\min,\ell}[r-1]}\right] + \left\lceil\left(\frac{k-r+2}{k-r+1}\right)^{1/3}\right\rceil\left(\frac{1}{\tilde{\Delta}_{\min,\ell}[r]}\right)(k-r+1)\log T\Bigg\},$$

where we divide the horizon T in m blocks, having at most $\min\{c\delta^{-2/3}k^{1/3}\log^{1/3}T, T\}$ length. Here

$$\Delta_{\min,\ell}[r] = \min_{t\in\ell block}\Delta_t[r] \textit{and} \tilde{\Delta}_{\min,\ell} = \min_{t\in\ell block}\tilde{\Delta}_t[r]$$

denote the gap for ℓ-th block.

Remark C.3. The terms in the regret expression depends on the dynamic gap of Agent r and $r-1$. As shown in Appendix H, it turns out that owing to the serial dictatorship nature of the problem, in order to characterize the regret of Agent r, it is sufficient to look at the behavior of Agent $r-1$. This inductive argument helps us to extend the results in a 2-agent NSCB to a general N-agent NSCB.

Remark C.4. The performance of Agent r depends crucially on Agent $r-1$, and based on whether Agent $r-1$ is exploring or exploiting, the regret depends on the higher ranked $r-2$ agents. Hence, the dynamic gap depending on both $r-1$ and $r-2$ sneaks in the regret expression via $\Delta_t[r]$ and $\tilde{\Delta}_t[r]$.

C.2. Discussion

Special case, $r=2$***:*** When $r=2$ in Theorem C.2, we exactly get back the regret of Agent 2 in Theorem 4.5. So, for Agent 2, there is no additional cost for extending NSCB to N agents.

Different terms of Theorem C.2: Similar to the 2 agent case, the first term in the regret expression presents the regret from exploration of Agent r, when Agent $r-1$ is exploring. Hence, the size of the dominated set is at most $r-2$, and the dynamic gap is given by $\Delta_{\min,\ell}[r]$.

Similarly, the second term corresponds to the *forced exploration* of Agent r. This occurs when Agent r is ready to commit, but is forced to explore since Agent $r-1$ is exploring. Note that this depends on how complex the system of Agent $r-1$ is. Since Agent 1 is exploring, the size of the dominated set is still $r-2$, and the dynamic gap is $\Delta_{\min,\ell}[r]$.

Finally, the third term corresponds to the regret when Agent r has committed. Observe that, in this case, the size of dominated set is at most $r-1$. Hence, it is characterized by $\tilde{\Delta}_t[r]$.

Note that we characterize the regret of Agent r by focusing on one phase of Agent $r-1$ (similar to the 2 agent case), and we show that the number of epochs of Agent $r-1$ in one block is at most $4\lceil[k/(k-r+2)]^{1/3}\rceil$, which causes the multiplicative pre-factor. Note that with $r=2$, the factor is absent, since the blocks are designed to contain at most 2 epochs of Agent 1.

Regret matches UCB-D3 of (Sankararaman et al., 2021) in stationary setup**:** Note that, in the stationary setup ($\delta=0$), the regret expression in Theorem C.2 matches to that of UCB-D3 (except a mildly weak dependence on k), which is shown to be order optimal. So, NSCB recovers the optimal regret in the stationary case.

D. NSCB without Black Board; Towards Complete decentralization

Upto now, we present NSCB with a black board, via which the agents communicate among themselves. In this section, we remove this, and obtain the same information via collision. We emphasize that without the black board, the learning algorithm

Algorithm 4 Black board removal for $N=2$

Initialize $Q_0 \leftarrow$ `Explore`
for $t=1,...,T$ **do**
  if Agent 2 experiences collision **then**
    $Q_t \leftarrow \mathsf{toggle}(Q_{t-1})$
  end if
end for

Algorithm 5 Black board removal for r-th Agent `NSCB`

Initialize $M_0[s] \leftarrow$ `Explore` for all $s \in [r-1]$
for $t=1,...,T$ **do**
  if Agent r experiences collision and reward goes to Agent $r' < r$ **then**
    $M_t[r'] \leftarrow \mathsf{toggle}(M_{t-1}[r'])$
  end if
end for

is *completely* decentralized.

D.1. Special case: Black board removal with $N=2$

In the presence of the black board, Agent 2 knows whether Agent 1 is exploring (on all $[k]$ arms) or committed to a particular arm. The same information can be gathered from a collision. Agent 2 maintains a latent variable Q_t, which indicates whether Agent 1 is in `Explore` or `Exploit` phase. At the beginning, $Q_0 \leftarrow$ `Explore`.

The routine is formally written in Algorithm 4. If at round t, Agent 2 faces a collision on arm j, one of two things can happen—(a) Agent 1 has ended exploring and committed to arm j or (b) Agent 1 has ended its exploitation and is exploring. This is true from the design of `NSCB`. After a collision, Agent 2 looks at Q_{t-1}. If $Q_{t-1} =$ `Explore`, then case (a) has happened and if $Q_{t-1} =$ `Exploit`, then case (b) has happened. So, just toggling the variable Q_t is enough for Agent 2 to keep track of Agent 1. It is easy to see that, from the round robbin structure of exploration, that after Agent's 1 phase changes, it may take upto k time steps for a collision to take place.

Lemma D.1 (Regret Guarantee)**.** *Suppose $\delta' = Ck\delta$, for a constant $C > 1$. Then, for a δ shifted system, `NSCB` without blackboard the regret of Agent 1 and 2 are given by*

$$R_1^{\text{No-Blackboard}}(\delta) \leq R_1(\delta') \text{and} R_2^{\text{No-Blackboard}}(\delta) \leq R_2(\delta')$$

where $R_1(.)$ and $R_2(.)$ denote the regret of Agent 1 and 2 respectively as shown in Theorem 4.5 with the presence of the black board. So, the regret guarantees of Theorem 4.5 extend in this case with δ replaced by δ'.

Remark D.2. It is easy to check from Theorem 4.5 that the regret increases with an increasing drift δ. In the framework without blackboard, we incur additional regret since $\delta' = Ck\delta$ (and $\delta' > \delta$). We upper bound the performance without black board by a worse system with drift $\delta' > \delta$ having a blackboard. The additional regret can be thought as the price of removing the blackboard.

Remark D.3. The above lemma can be shown via a reduction argument (see Appendix I). For a δ shifted system, upto time k, the maximum total shift is $k\delta$, and hence with δ', we ensure that the system remains stationary in these k time steps. We emphasize that `NSCB` is an asynchronous algorithm, and the dentition of δ' helps in reducing the reduction to a worse system with drift $\delta' > \delta$.

D.2. Black board removal with N agents

`NSCB` is an asynchronous algorithm, and hence establishing coordination between agents is quite non-trivial. In previous works, such as (Sankararaman et al., 2021), the learning includes a fixed set of time slots for communication among agents. This coordinated communication can not be done for `NSCB`, since the phases start and end at random times. Hence, to handle this problem, we consider a slightly stronger reward model. We emphasize that the stronger reward model was also used in previous works such as (Liu et al., 2021).

Reward model: To ease communication across agents, we assume that in case of collision, the reward is given to the highest ranked agent, and all the remaining agent gets zero reward, as well as the index of the agent who gets the (non-zero) reward.

We remark that this side information is not impractical in applications like college admissions, job markets etc., and this exact reward model is also studied in (Liu et al., 2021).

The routine is formally written in Algorithm 4. Under this new reward model, Agent r maintains a set of latent variables, $M_t[s]$ for all $s \in [r-1]$, where $M_t[s] \in \{\texttt{Explore}, \texttt{Exploit}\}$. If at time t, if Agent r experiences a collision, and the reward goes to an Agent r', with $r' < r$ (i.e., r' is a higher ranked agent), then Agent r toggles $M_t[r']$. In this way, after a collision on arm j, Agent r knows that either Agent r' has committed on arm j or it is exploring on a set of arms including j. Hence, based on $M_{t-1}[r']$, Agent r knows which event has happened exactly, which is the information a black board would have provided. From the round robbin nature of exploration, detecting this may take at most k steps.

Lemma D.4 (Regret Guarantee)**.** *Suppose $\delta' = Ck\delta$, for a constant $C > 1$. Then, for a δ shifted system, NSCB without blackboard the regret of Agent r is given by*

$$R_r^{\textit{No-Blackboard}}(\delta) \leq R_r(\delta')$$

where $R_r(.)$ denotes the regret of Agent r, as in Theorem C.2 with the presence of the black board. So, the regret guarantee of Theorem C.2 extends in this case with δ replaced by δ'.

Remark D.5. Similar to the 2 agent case, here also, the removal of black board results in an increase of regret,since we replace δ by δ', which is bigger than δ.

Remark D.6. The above result holds under the modified and stronger reward model. Design of an efficient coordination protocol in an asynchronous system is left as future work.

E. Simulations

In Figure 2, we show through simulations that - *(i)* Snooze-IT of (Krishnamurthy & Gopalan, 2021) outperforms vanila UCB of (Auer et al., 2002) in the case of single agent, *(ii)* NSCB multi-agent setting is effective to simulate and matches the theoretical insights, and *(iii)* in the multi-agent case, NSCB outperforms UCB-D3, especially for higher ranked agents. In all settings, we consider the arms to have gaussian distribution with variance 0.4 and means varying with time as given below. All plots are plotted after averaging over 10 runs, with the median being highlighted in bold, and the inter-quartile range between the 25^{th} and 75^{th} quantiles in the shaded region. In the single agent setting of Figure 2(a), we considered three arms, with the third arm having a fixed mean of 0.5 throughout. In the multi-agent setting in Figures 2(b), 2(c), 2(d), 2(e), we initialized the arm means randomly from the uniform distribution on $[0,1]$. In each of the 10 runs, the arm means for every agent-arm pair evolved independently according to a symmetric random walk by either adding or subtracting a value of δ as specified in the plot title. We simulated the NSCB algorithm by assuming access to a black-board, the performance on which can be translated to the setting without access to the black-board as seen in Lemma D.4. For UCB-D3, we use the standard hyper-parameters recommended in (Sankararaman et al., 2021). The plots are averaged over the randomness in the arm-mean variation across time as well.

We observe in Figure 2(a), that the performance of `Snooze-IT` is much better than that of the classical UCB algorithm of (Auer et al., 2002). In Figures 2(b) and 2(c), we validate the performance of `NSCB` for a 3 agent system with different values of δ. We first note that the regret flattens out sometimes, which indicates that the `NSCB` algorithm has chosen an 'optimal' arm, and commit to it, incurring zero regret as shown in Lemma G.3. Furthermore, matching to our intuition, the regret of agent ranked 1 is the lowest, then Agent 2 and and finally Agent 3. This is because as Theorem 4.5 and C.2 suggest, the performance of an Agent gets dominated by all the agents ranked higher. Furthermore, observe that the flattening of regret for Agents 2 and 3 are much infrequent compared to that of Agent 1.

In Figures 2(d) and 2(e), we compare the performance of NSCB with that of UCB-D3 in a dynamic environment. We find that although the performance of agent 1 is similar in the two systems, the performance of the lower ranked agents are much superior in NSCB compared to UCB-D3. This shows that NSCB is sensitive to the potential variations in arm-means and helps all agents adapt faster compared to UCB-D3 which is designed assuming the environment is stationary. Moreover, since the phase lengths grow exponentially with time in UCB-D3 where lower ranked agents assume that the best arm of the higher ranked agents will remain constant within a phase. In the dynamic case, this assumption is no longer valid, and thus the regret of lower ranked agents are much worse under UCB-D3, as compared to NSCB.

In Figure 2(f), we take a stationary environment and compare UCB D3 with `NSCB` run with $\delta = 0.025$. In this plot we observe that even in the stationary setting, the regret of NSCB is superior to UCB-D3. The reason is that the phase lengths in UCB-D3 was fixed to be exponentially growing, while the phase lengths in `NSCB` is chosen adaptively depending on the empirical arm means.

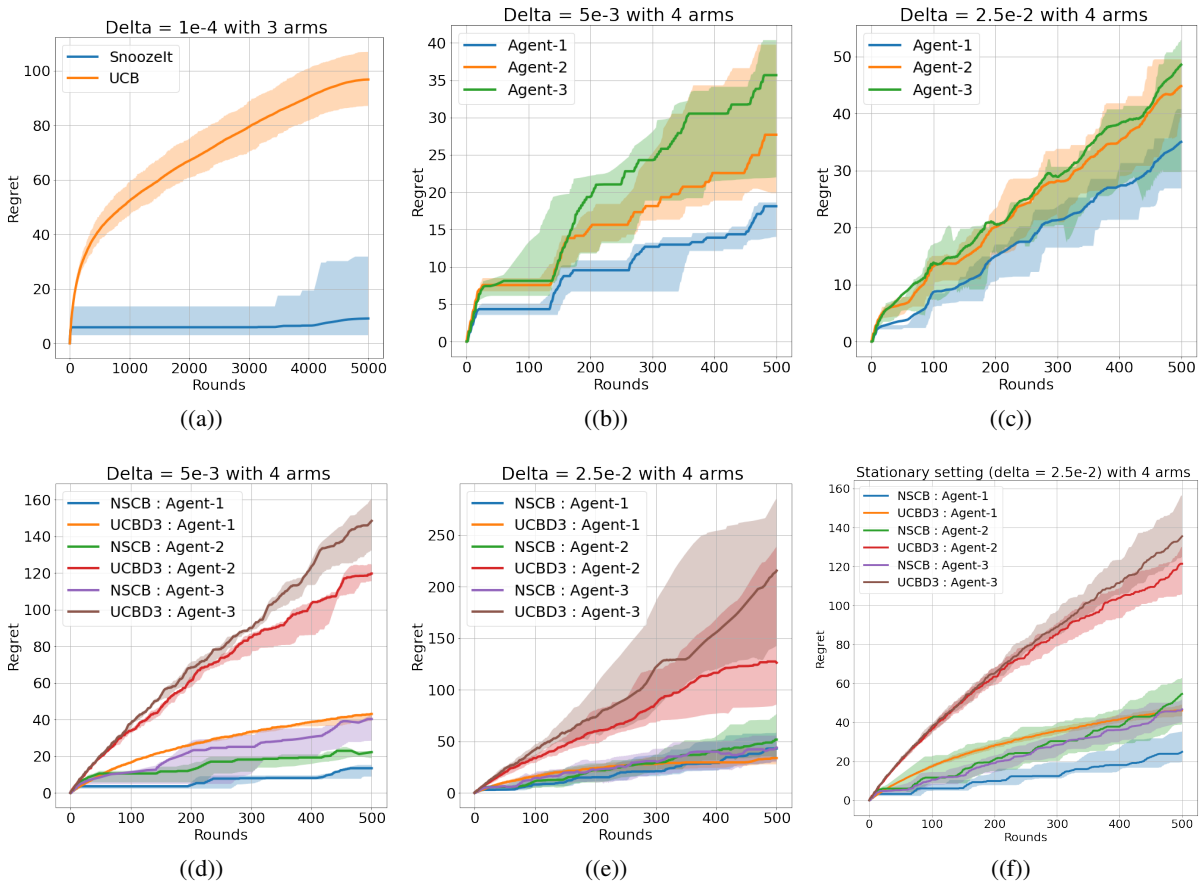


Figure 2: In (a), we compare SnoozeIT and UCB with $k=3$. In (b) and (c) NSCB on a system with 3 agents and 4 arms is simulated, and the same systems are compared with UCB-D3 in (d) and (e). In (f) we compare against a stationary system with NSCB run with $\delta=0.025$.

F. Conclusion and open problems

We introduced the problem of decentralized, online learning in two-sided markets when the underlying preferences vary smoothly over time. We propose and analyze an asynchronous algorithm, namely NSCB, first for 2 agents, and then for N agents. We provide a complete characterization of the regret of NSCB. Moreover, we verify via simulations, the theoretical insight we previously obtained. This paper however leaves several intriguing open problems: (a) we want to understand whether the assumption of known δ be relaxed, i.e., a step towards making NSCB parameter-free and problem adaptive; (b) whether it is possible to extend our analytic framework to the dynamic framework to general markets beyond serial dictatorship; and (c) to consider other forms of non-stationary such as piece-wise constant markets or variations with a total budget constraint. We keep these as future endeavors.

G. Proof of Theorem 4.5; NSCB with 2 Agents

G.1. Preliminaries: Theoretical Guarantees of SNOOZE-IT with k arms

As is standard in formalizing bandit processes (Lattimore & Szepesvári, 2020), we assume that the random process lies in a probability space endowed with a collection of independent and identically distributed random variables $(U_{i,j}[t])_{i\in[N],j\in[k]t\geq 1}$. For each $i\in[N]$ and $j\in[k]$, and $k\geq 1$, the random variables $(U_i[k])$ is distributed as the 0 mean, unit variance Gaussian random variable[3]. With this description, the realized reward by agent $i\in[N]$, when it matches with arm $j\in[k]$ for the k^{th} time at time-index

[3]Our analysis can be extended verbatim to any sub-gaussian distribution

t is given by $U_{i,j}[k]+\mu_{i,j,t}$. In this description, the set of arm-means $(\mu_{i,j,t})_{i\in[N],j\in[k],t\in[T]}$ are fixed non-random parameters.

Definition G.1 (Good Event)**.**

$$\mathcal{E} := \bigcap_{i=1}^{N} \mathcal{E}^{(i)},$$

where

$$\mathcal{E}^{(i)} := \left\{ \forall t \in [T], \forall j \in [k], \forall w \leq t, \left| \frac{1}{w} \sum_{s=0}^{w} U_{i,j}[t-s] \right| \leq r(w) \right\},$$

here $r(w) := \sqrt{\frac{8\log(T)}{w}}$.

In words, the event $\mathcal{E}$ is the one in which every contiguous sequence of i.i.d. random variables is 'well-behaved'. The event $\mathcal{E}^{(1)}$ is identical to the good-event specified for the single agent case in (Krishnamurthy & Gopalan, 2021). Standard concentration inequalities give that this occurs with high probability which we record in the proposition below.

Proposition G.2.

$$\mathbb{P}[\mathcal{E}] \geq 1 - \frac{2Nk}{T^2}.$$

Implication: The above proposition states that it is sufficient to look at the behavior of the learning algorithm under event $\mathcal{E}$. Since the rewards are bounded by 1, the additional regret incurred over T rounds will be $\mathcal{O}(\frac{Nk}{T})$, which decays with T, and hence, from now on, we only focus on events conditioned on $\mathcal{E}$.

Proof. Fix a $t \in [T], i \in [N], j \in [k]$ and $w \leq t$. Classical sub-gaussian inequality gives that

$$\begin{aligned} \mathbb{P}\left[\left| \frac{1}{w} \sum_{s=0}^{w-1} U_{i,j}[t-s] \right| > r(w) \right] &\leq 2\exp\left(-\frac{1}{2} w r(w)^2 \right), \\ &= \frac{2}{T^4}. \end{aligned}$$

Now, taking an union bound over t, i, j and w gives that

$$\mathbb{P}[\mathcal{E}^{\complement}] \leq \frac{2Nk}{T^2}.$$

□

The definition of the good event is useful due to the following result.

Lemma G.3 (No regret in the exploit-phase)**.** *If the good event $\mathcal{E}$ in Definition G.1 holds, then every agent in every exploit phase will incur* 0 *regret.*

Proof. We first prove the result for agent ranked 1. For any phase i_1 of Agent 1, denote by time $t = g_{i_1}$ to be the time-instant at which an arm a and $\lambda > 0$ is identified that satisfies $a >_\lambda b$ (this means arm a = Lambda-Opt(λ,$[k]$ from Definition 1) for all $b \in [k] \setminus \{a\}$. In words, time g_{i_1} is the time when the statistical test by Agent 1 succeeds. Recall from the notations in the algorithm that $\Lambda_{i_1} := g_{i_1} - s_{i_1}$.

Suppose in a phase i_1, agent 1 exploits an arm $a \in [k]$ one or more rounds. Notationally, this is from times $[g_{i_1}+1, s_{i_1+1}]$. We will show that *(i)* there exists a minimum gap $\lambda > 0$, such that at time g_{i_1+1}, for all arms $a^{'} \in [k] \setminus \{a\}$, the mean of arm a exceeds $a^{'}$ by a certain margin, and *(ii)* in the duration $[s_{i_1} + \tau_{i_1}, s_{i_1+1}]$ is set such that the chosen arm a continues to be optimal in the entire EXPLOIT phase.

The first claim is formalized below.

Under the good event $\mathcal{E}$, there exists a time $t^{'} \in [s_{i_1}, g_{i_1}]$, such that for all arms $b \in [k] \setminus \{a\}$, $\mu_{a,t'} - \mu_{b,t'} \geq 4\sqrt{4\frac{\log(T)}{\Lambda_{i_1}}} - 2(k-1)\delta$.

Proof. The statistical test succeeded at time g_{i_1}, i.e., there exists a $\lambda > 0$ such that $a >_\lambda b$, for all $b \in [k] \setminus \{a\}$. By Definition 3.1, the window size $w := k\lceil \frac{c_1 \log(T)}{\lambda^2} \rceil$. Since the test succeeds at time $t = g_{i_1}$, clearly $w \leq \Lambda_{i_1}$.

In order to describe the proof, we set some notations. For every arm $a \in [k]$, denote by the set of times $\mathcal{L}_{i_1}^{(a)} := \{l_1^{(a)}, \cdots, l_{w/k}^{(a)}\}$ to be the w/k times arm a was played in the time-interval $[g_{i_1} - w, g_{i_1}]$. These times are random variables —however conditioned on g_{i_1}, these are deterministic since in the Explore phase of Algorithms 2 and 3, agents explore the arms in a round-robin fashion from arms indexed the smallest to the largest. Denote by $\mu_{a,t}(w/k) := \frac{k}{w} \sum_{s \in \mathcal{L}_{i_1}^{(a)}} \mu_{a,s}$. For every arm $a \in [k]$, denote by the random index l_a to be the number of times arm a has been played in the past, before time $g_{i_1} - w$.

Since the statistical test succeeds at time $t = g_{i_1}$, we have from Definition 3.1

$$\mu_{a,t}(w/k) + \frac{k}{w} \sum_{s=l_a}^{l_a + w/k} U_a[s] > \mu_{b,t}(w/k) + \frac{k}{w} \sum_{s=l_b}^{l_b + w/k} U_b[s] + 4r(w/k) - (k-1)\delta.$$

Re-arranging and using the definition of the Good event, we have

$$\begin{aligned} \mu_{a,t}(w/k) - \mu_{b,t}(w/k) &> 4r(w/k) - (k-1)\delta + \frac{1}{w} \sum_{s=l_b}^{l_b + w/2} U_b[l] - \frac{1}{w} \sum_{s=l_a}^{l_a + w/2} U_a[l], \\ &\geq 2r(w/k) - (k-1)\delta. \end{aligned}$$

where the second inequality stems from the definition of the good event. Now, since the drift is bounded by δ, we have that

$$\mu_{a,t}(w/k) - \mu_{b,t}(w/k) \leq \frac{k}{w} \sum_{s \in \mathcal{L}_{i_1}^{(a)}} (\mu_{a,s} - \mu_{b,s}) + (k-1)\delta.$$

Combining the preceding two displays, we get that

$$\frac{k}{w} \sum_{s \in \mathcal{L}_{i_1}^{(a)}} (\mu_{a,s} - \mu_{b,s}) > 2r(w/k) - 2(k-1)\delta.$$

The second inequality follows from the fact that the window size $w \leq \Lambda_{i_1}$ is smaller than the explore duration of phase i_1. Substituting the expression for $r(w/k)$, we obtain

$$r(w/k) = \sqrt{\frac{8k \log T}{w}} = c\lambda,$$

where c is a universal constant.

Since λ is constrained with the number of samples in w, we have

$$\frac{c_1 k \log T}{\lambda^2} \leq \Lambda_{i_1} \Rightarrow \lambda \geq c_0 \sqrt{\frac{k \log T}{\Lambda_{i_1}}}$$

Substituting this, we obtain

$$\frac{k}{w} \sum_{s \in \mathcal{L}_{i_1}^{(a)}} (\mu_{a,s} - \mu_{b,s}) \geq 2\sqrt{\frac{4k \log(T)}{\Lambda_{i_1}}} - 2(k-1)\delta = 4\sqrt{\frac{k \log(T)}{\Lambda_{i_1}}} - 2(k-1)\delta$$

Now, since the average gap exceeds a bound, it implies that there exists at-least one $t' \in [s_{i_1}, g_{i_1}]$ such that $\mu_{a,t'} - \mu_{b,t'} > 4\sqrt{\frac{k \log(T)}{\Lambda_{i_1}}} - 2(k-1)\delta$.

□

Now, since the drift at each time-step in each arm is at-most δ, arm a will remain optimal compared to arm b at-least in the time-interval $[t^{'}, t^{'} + \frac{2}{\delta}(4\sqrt{\frac{k\log(T)}{\Lambda_{i_1}}} - 2(k-1))]$, i.e., arm a is optimal compared to arm b in the duration $[t^{'}, t^{'} + \frac{4}{\delta}\sqrt{\frac{4k\log(T)}{\Lambda_{i_1}}} - 2(k-1)]$. Since $t^{'} \geq s_{i_1}$, and from Algorithms 1, 2 and 3 the definition of buffer period is $buffer_1 := \frac{4}{\delta}\sqrt{\frac{4k\log(T)}{\Lambda_{i_1}}} - 2(k-1)$, arm a is superior to arm b in the exploit duration of phase i_1. Now, since arm b was arbitrary, this implies that Agent 1 will incur no regret during the exploit phase of i_1.

For the general case, we will prove by induction. Suppose the induction hypothesis that all agent ranked 1 through to $r-1$ are incurring 0 regret in an exploit phase. Notice from the description of Algorithm 3 that agent ranked r can potentially go into an exploit phase if and only if all agents ranked 1 through $r-1$ are in an exploit phase. Additionally, the base case of the induction hypothesis is what we established in the preceding paragraph where agent ranked 1 incurs 0 regret in the exploit phase. Under this induction hypothesis, we will now argue that agent ranked r will also incur 0 regret in the corresponding exploit phase.

We make one observation based on the serial-dictatorship structure. If all agents ranked 1 through $r-1$ are in *(i)* Exploit phase and *(ii)* are incurring 0 regret, then the stable match optimal arm for agent ranked r is to play the arm with the highest mean among those arms not being exploited by agents ranked 1 through $r-1$. This is a simple consequence of the definition of stable match (c.f. Section 2). Thus, it suffices to argue that when agent ranked r commits, it commits to the optimal arm. We use identical arguments as for agent ranked 1 to show the following: If at time t, for a given $\Omega \subset [k]$ with $|\Omega| = r-1$, the statistical test succeeds with arm $a \in [k]$, then there exists a time $t^{'} \in [s_{i_r}, t]$, such that for all arms $b \in \Omega \setminus \{a\}$, $\mu_{a,t'} - \mu_{b,t'} \geq 4\sqrt{4(k-r)\frac{\log(T)}{\Lambda_{i_1}}} - 2(k-r+1)\delta$. The proof follows identical arguments as that of Claim G.1 by using the observation that $r(w)$ is a decreasing function of w. □

We now characterize the regret in the exploration phase, where SNOOZE-IT plays the round robbin algorithm. We first start with the setup where an agent gets to explore all k arms. Later, we can easily extend these results, where an agent explores a subset of arms.

Let us first recall a few notation from the previous section. For Agent 1, in episode i_1, assume that the test succeeds at g_{i_1}, and accordingly we define $\Delta_{i_1} = g_{i_1} - s_{i_1}$. We have the following result:

Lemma G.4 (Episode Regret)**.** *Under the good event $\mathcal{E}$, the Regret of Agent 1 in the exploration phase of i_1-th phase is given by*

$$R_1 \leq C_1 \left[k\sqrt{\Lambda_{i_1} \log T} \right].$$

Proof. The regret of Agent 1 in episode i_1 is given by

$$R_1 = \sum_{t=s_{i_1}+1}^{s_{i_1+1}} \mu_t^* - \mu_{SNOOZE-IT}(t).$$

The above can be further decomposed as

$$R_1 = \sum_{t=s_{i_1}+1}^{g_{i_1}} \mu_t^* - \mu_{SNOOZE-IT}(t) + \sum_{t=s_{i_1}+1}^{s_{i_1+1}} \mu_t^* - \mu_{SNOOZE-IT}(t).$$

Note that as shown in Lemma G.3, the second term in the above expression is 0, and so we need to calculate the first term only. Furthermore,

$$R_1 = 1 + \sum_{t=s_{i_1}+1}^{g_{i_1}-1} \mu_t^* - \mu_{SNOOZE-IT}(t)$$

from the fact the the mean rewards lie in $[0,1]$.

We first note that the statistical test did not pass until time $g_{i_1} - 1$. Hence, there exists no two arms a, b such that $a = $Lambda-Opt$(\tilde{\lambda}, [k])$. Using the definition of the statistical test with $t \leq g_{i_1} - 1$, under the good event $\mathcal{E}$, for τ number of

pulls between any 2 arms, we obtain

$$\mu_{a,t}(\tau)-\mu_{b,t}(\tau)\leq C_1\sqrt{\frac{\log T}{\tau}}-(k-1)\delta. \tag{1}$$

Let us now look at the regret decomposition

$$R_1=1+\sum_{t=s_{i_1}+1}^{g_{i_1}-1}\mu_t^*-\mu_{SNOOZE-IT}(t)$$

Since, we know that the `Snooze-IT` plays the round robbin algorithm in the interval $s_{i_1}+1$ to $g_{i_1}-1$, we split the said interval in k parts, $\mathcal{L}_{i_1}^{(a)}$ for all arms $a\in[k]$. Let us first look at the interval $\mathcal{L}_{i_1}^{(a)}$ where `Snooze-IT` plays arm a. We obtain

$$\sum_{t\in\mathcal{L}_{i_1}^{(a)}}\mu_t^*-\mu_a=\sum_{t\in\mathcal{L}_{i_1}^{(a)}}\max_{b\in[k]}\mu_{b,t}-\mu_{a,t}$$

Let us now break the above sum. We define the set O_b^* as the set of time indices where arm b is the optimal arm, for all $b\in[k]$. With this, we have

$$\begin{aligned}
\sum_{t\in\mathcal{L}_{i_1}^{(a)}}\mu_t^*-\mu_a&=\sum_{t\in\mathcal{L}_{i_1}^{(a)}\cap O_b^*}\sum_{b\in[k]}\mu_{b,t}-\mu_{a,t}\\
&=\sum_{b\in[k]}\sum_{t\in\mathcal{L}_{i_1}^{(a)}\cap O_b^*}\mu_{b,t}-\mu_{a,t}\\
&\overset{(i)}{\leq}C\sum_{b\in[k]}\left[|\mathcal{L}_{i_1}^{(a)}\cap O_b^*|\{\sqrt{\frac{\log T}{|\mathcal{L}_{i_1}^{(a)}\cap O_b^*|}}-(k-1)\delta+(k-1)\delta\}\right]\\
&\leq C\sum_{b\in[k]}\left[\sqrt{|\mathcal{L}_{i_1}^{(a)}\cap O_b^*|\log T}+(k-1)\delta|\mathcal{L}_{i_1}^{(a)}\cap O_b^*|\right]\\
&\leq C\sqrt{\log T}\sum_{b\in[k]}\left[\sqrt{|\mathcal{L}_{i_1}^{(a)}\cap O_b^*|}\right]\\
&\overset{(ii)}{\leq}C\sqrt{\log T}\sqrt{k}\sqrt{|\mathcal{L}_{i_1}^{(a)}|}\\
&=C\sqrt{k\log T}\sqrt{|\mathcal{L}_{i_1}^{(a)}|}
\end{aligned}$$

where (i) follows from Equation 1, and using the fact that we require at most $k\delta$ deviation to align arm a and b (owing to the round robbin nature of the play). Step (ii) follows from Cauchy Schwartz inequality. Now, the total regret we obtain is given by

$$\begin{aligned}
\sum_{a\in[k]}\sum_{t\in\mathcal{L}_{i_1}^{(a)}}\mu_t^*-\mu_a&\leq\sum_{a\in[k]}\left[C\sqrt{k\log T}\sqrt{|\mathcal{L}_{i_1}^{(a)}|}\right]\\
&\leq C_1\left[k\sqrt{\log T}\sqrt{\Lambda_{i_1}}\right]
\end{aligned}$$

Hence, under the good event, the regret of agent 1 is given by

$$R_1\leq C_1\left[k\sqrt{\log T}\sqrt{\Lambda_{i_1}}\right].$$

□

We now connect the quantity Λ_{i_1} to the complexity gap of the problem. Since we are dealing with agent ranked 1, we consider the set of dominated arms, $\mathcal{C}=\phi$ (null set). We write $\lambda_t[1]$ as the complexity gap of the problem. We have the following lemma.

Lemma G.5 (Gap dependent regret)**.** *The regret of agent 1 is given by*

$$R_1 \leq Ck\sqrt{\Lambda_{i_1}\log T} \leq \frac{C}{\lambda_{g_{i_1}-1}[1]}k\log T$$

Proof. The proof depends on the following claim: In the exploration phase of agent 1, at time $t' = s_{i_1}+\tau$, if the gap satisfies $\lambda_{t'}[1] > C\sqrt{\frac{\log T}{k\tau}}$, then the statistical test passes at time t'. In order to show the claim, we first show that for a window size of $w(\lambda_{t'}[1]) = C\frac{k\log T}{\lambda^2_{t'}[1]}$, such that all arms have $\tilde{w} = w(\lambda_{t'}[1])/k$ samples, we have

$$\mu_{a,t'}(\tilde{w}) - \mu_{b,t'}(\tilde{w}) > k(\lambda_{t'}[1] - \delta)$$

for any pair of arms a and b. The above is easy to show and follows from the definition of the problem complexity, following the lines of Lemma 5 of (Krishnamurthy & Gopalan, 2021).

Now, with the window of size $\frac{k\log T}{\lambda^2_{t'}[1]}$, such that all arms have $\tilde{w}$ samples, we apply the test on arms a and b. After some algebric manipulation, and invoking the condition of the test, we observe that as long as $\lambda_{t'} > c\sqrt{\frac{\log T}{\tau}}$, or $\tau > c\frac{\log T}{\lambda^2_{t'[1]}}$, we can use the window of size $\frac{k\log T}{\lambda^2_{t'}[1]}$ and the test succeeds at the end of the window.

Now, recall that the test did not pass at $s_{i_1} - 1$, and so from the above claim, we obtain

$$\lambda_{g_{i_1}-1}[1] \leq c\sqrt{\frac{\log T}{\Lambda_{i_1}-1}},$$

which implies

$$\sqrt{\Lambda_{i_1}} \leq \frac{C}{\lambda_{g_{i_1}-1}[1]}\sqrt{\log T},$$

thus proving the lemma. □

This concludes the regret proof where the agent gets to explore all k arms. We now explain how to extend this for agents exploring a subset of arms. For the general case, we will focus on the constrained set of arms.

Let us focus on Agent ranked r. It first constructs the set $\mathcal{C}(r)$ at time t. Then, its action will be exploring the arms in $[k]\setminus\mathcal{C}(r)$. This is given by the setup of the problem. We have the following claim, which uses identical arguments as Lemma G.4:

The regret in phase i_r of Agent r from playing round-robbin on set $[k]\setminus\mathcal{C}_r$ is given by

$$R_r \leq \frac{C}{\lambda_{g_{i_r}-1}[1]}(k - |\mathcal{C}_r|)\log T$$

When agent r explores all the arms, $\mathcal{C}_r = \phi$, and we get back to the regret of $\mathcal{O}(k\log T)$.

We finally argue about the length of an epoch. We have the following claim.

Lemma G.6 (Episode length)**.** *For k-armed* `SNOOZE-IT`*, the length of an episode is at least $c\delta^{-2/3}k^{1/3}\log^{1/3}T$.*

Proof. Recall that, the length of the buffer is set as $\frac{4}{\delta}\sqrt{\frac{4k\log(T)}{\Lambda_{i_1}}} - 2(k-1)$. We consider 2 cases:

Case I $\Lambda_{i_1} \geq buffer_1$: In this case, there is no exploit phase, and so the length of an episode is Λ_{i_1}

Case II $\Lambda_{i_1} \geq buffer_1$: In this setting, there exists an exploit phase, and the length of the episode is given by $buffer_1$ (comes from the definition of buffer)

Now considering the above two alternative, the length of an episode is given by $\max\{\Lambda_{i_1}, buffer_1\}$. In order to lower bound the episode length, we minimize $\max\{\Lambda_{i_1}, buffer_1\}$, which happens when

$$\Lambda_{i_1} = \frac{4}{\delta}\sqrt{\frac{4k\log(T)}{\Lambda_{i_1}}} - 2(k-1).$$

Now, in the standard slowly varying framework, the value of δ is typically small. For example, see (Krishnamurthy & Gopalan, 2021) Theorem 3 of (Krishnamurthy & Gopalan, 2021), we require $\delta = 1/T^{\alpha}$ for $\alpha \in (0,1)$. In the regime where T is large, the second term in the above expression becomes a minor one, and for compactness of the expression, and to ease the calculation, we ignore the second term. We obtain

$$\Lambda_{i_1} \approx \frac{4}{\delta}\sqrt{\frac{4k\log(T)}{\Lambda_{i_1}}} \Rightarrow \Lambda_{i_1} \approx c\delta^{1/3}k^{1/3}\log^{1/3}T,$$

and thus the episode length is at least $c\delta^{1/3}k^{1/3}\log^{1/3}T$, which proves the lemma. □

G.2. Continuing the proof of Theorem 4.5 with results form Section G.1

In this section, we prove the regret of both Agents 1 and 2 for Algorithms 1 and 2.

More interestingly, in this section, we provide a full characterization of the regret of Agent 2. Note that since Agent 2 plays on a restrictive or dominated set of arms, dictated by Agent 1, it encounters additional regret. In the description of Algorithm 2, we pointed out the scenarios where Agent 2 is forced to (a) either explore or (b) to stop exploiting. Here, we obtain a regret upper-bound from these forced exploration-exploitation.

Notation: To better understand the algorithm, let use focus on a particular phase of Agent 1, say the i_1-th epoch/episode. We use the same notation defined in Algorithm 2. So, s_{i_1} denotes that start-time of epoch i_1 ans s_{i_1+1} denotes the end of epoch i_1. The exploration duration before committing to an arm is Λ_{i_1}, and so the exploitation phase starts at $s_{i_1}+\Lambda_{i_1}$. Similarly, the length of exploitation is $s_{i_1+1}-\Lambda_{i_1}$. Let us also assume that the committed arm of Agent 1 in this phase is i^*.

In the calculation below, we condition on the good event $\mathcal{E}$. At the end of this section, we provide a justification of the sufficiency of conditioning on this event and provide the total regret guarantees.

Since Agent 1 plays Algorithm 1, as shown in Lemma G.3, during the exploitation phase, it incurs no regret.

During the exploration phase, using Lemma G.4, the regret of Agent 1 in i_1-th phase is

$$R_1(i_1) \leq C(k\sqrt{\Lambda_{i_1}\log T})$$

We now look at the behavior of Agent 2, while Agent 1 is in phase i_1. As shown in Figure 1, there can be multiple phases of Agent 2 inside one phase of Agent 1, and hence let us assume that at the beginning of epoch i_1, the phase number of Agent 2, given by $i_2 = n_{i_1}$, and by the end of phase i_1, the episode index is $n_{i_1}+N_{i_1}$, thus ensuring N_{i_1} number of episodes of Agent 2, within one episode of Agent 1.

G.3. Regret of Agent 2 during the exploration period of Agent 1 in the i_1th phase

In this phase, which lasts for Λ_{i_1} rounds, we characterize the regret of Agent 2. For this, let us define $\tau_{n_{i_1}}$ as the duration, starting from s_{i_1} it takes for Agent 2 to commit to an arm unconditionally. This means that in the absence of competition, starting from s_{i_1}, Agent 2 would take $\tau_{n_{i_1}}$ to commit to an arm by exploring all the arms.

We have 2 cases:

Case I ($\Lambda_{i_1} \leq \tau_{n_{i_1}}$): In this case, since Agent 1 commits first, the regret of Agent 2, from Lemma G.4 is given by $\mathcal{O}(k\sqrt{\Lambda_{i_1}\log T})$. In this case, Agent 2 is not forced to explore.

Case II ($\Lambda_{i_1} \geq \tau_{n_{i_1}}$): In this case, Agent 2 incurs a regret of $\mathcal{O}(k\sqrt{\tau_{n_{i_1}\log T}})$ plus some additional regret owing to force exploration. The forced exploration comes from the fact that in this case, although Agent 2 has enough information to commit, it still explores because Agent 1 has not committed yet, and the commitment of Agent 2 will cause periodic collisions for Agent 2.

G.3.1. Forced Exploration

We now characterize the regret of Agent 2 form forced exploration. Note that Agent 2 is forced to explore at time t if:

1. Agent 1 is exploring, and
2. At time t, $S_2^{(j_t)}$ is non-empty, where $j_t \in [k]$ is the arm played by Agent 1.

Let us understand this in a bit more detail. If $S_2^{(j_t)}$ is non-empty, it implies that without the presence of competition, Agent 2 would have played arm j_t. This comes from the definition of $S_2^{(.)}$. Now, when Agent 1 is playing that arm, it implies a forced exploration on Agent 2. We can write down the above forced exploration time steps as the following.

$$\text{Forced Exploration} = \sum_{t=s_{i_1}}^{s_{i_1}+\Lambda_{i_1}} \sum_{j=1}^{k} \mathbf{1}(j_t = j)\mathbf{1}\left(\tau_{n_{i_1}}^{(j)} < t\right),$$

where $\tau_{n_{i_1}}^{(j)}$ is defined as the duration of the exploration period before the $(\tilde{\lambda}, \mathcal{A})$ test succeeds with $\mathcal{A} = [k] \setminus \{j\}$, when Agent is in phase `Explore ALL`.

Since the mean rewards lie in $[0,1]$, the regret obtained from forced exploration is

$$\mathcal{O}\left[\sum_{t=s_{i_1}}^{s_{i_1}+\Lambda_{i_1}} \sum_{j=1}^{k} \mathbf{1}(j_t = j)\mathbf{1}\left(\tau_{n_{i_1}}^{(j)} < t\right)\right],$$

Combining this two, the regret of Agent 2 during the exploration phase of Agent 1 is given by

$$\mathcal{O}\left[\mathbf{1}(\text{Case-I})k\sqrt{\Lambda_{i_1}\log T} + \mathbf{1}(\text{Case -II})\left(k\sqrt{\tau_{n_{i_1}}\log T} + \sum_{t=s_{i_1}}^{s_{i_1}+\Lambda_{i_1}} \sum_{j=1}^{k} \mathbf{1}(j_t = j)\mathbf{1}\left(\tau_{n_{i_1}}^{(j)} < t\right)\right)\right]$$

G.4. Regret of Agent 2 during exploitation phase of Agent 1

Suppose Agent 1 commits to arm i^*. In this phase, Agent 2 is forced to play in a restrictive set $[k] \setminus \{i^*\}$. Note that in this phase, several cases may happen:

Agent 2 is exploiting: Note that Agent 2 keeps the set $S_2^{(j)}$ for all $j \in [k]$, and if $j \neq i^*$, Agent 2 immediately commits to j. Keeping track of such $S_2^{(j)}$ thus ensures that agent 2's exploration are not wasted.

Furthermore, if $S_2^{(j)}$ is empty, for all $j \neq i^*$, Agent 2 will keep accumulating samples, now from a restrictive set $[k] \setminus \{i^*\}$, and may commit to an arm within the set. In both the cases, we have shown in Lemma G.3 that the regret incurred is zero.

Agent 2 is exploring: Note that inside the exploit phase of Agent 1, Agent 2 basically plays the Snooze-IT algorithm over the arm-set $[k] \setminus \{i^*\}$. Hence, using Lemma G.4 on the constrained set $[k] \setminus i^*$, the regret is given by

$$\mathcal{O}\left(\sum_{p=n_{i_1}+1}^{n_{i_1}+N_{i_1}} (k-1)\sqrt{\tilde{\tau}_p^{(i^*)}\log T}\right),$$

where N_{i_1} is the number of phases of Agent 2 in the current exploitation phase of Agent 1, and $\tilde{\tau}_p^{(i^*)}$ is defined as the duration of the exploration period before the $(\tilde{\lambda}, \mathcal{A})$ test succeeds with $\mathcal{A} = [k] \setminus \{i^*\}$, when Agent 2 is in state `Explore-`i^*.

G.5. Total Regret of both agents in one phase

Putting everything together, the regret of Agent 1 and 2, denoted by $R_1(i_1)$ and $R_2(i_1)$ respectively, during the i_1-th phase of Agent 1 is given by

$$R_1(i_1) \leq \mathcal{O}(k\sqrt{\Lambda_{i_1}\log T}), \text{and}$$

$$R_2(i_1) \leq \mathcal{O}\Bigg[\underbrace{\mathbf{1}(\text{Case-I})k\sqrt{\Lambda_{i_1}\log T}}_{T_1} + \underbrace{\mathbf{1}(\text{Case -II})\left(k\sqrt{\tau_{n_{i_1}}\log T} + \sum_{t=s_{i_1}}^{s_{i_1}+\Lambda_{i_1}}\sum_{j=1}^{k}\mathbf{1}(j_t=j)\mathbf{1}\left(\tau_{n_{i_1}}^{(j)} < t\right)\right)}_{T_2}$$
$$+ \underbrace{\left(\sum_{\ell=n_{i_1}+1}^{n_{i_1}+N_{i_1}}(k-1)\sqrt{\tilde{\tau}_{\ell}^{(i^*)}\log T}\right)}_{T_3}\Bigg]$$

G.5.1. REGRET FOR AGENT 1 IN PHASE i_1:

We now bound $\sqrt{\Lambda_{i_1}}$ using Lemma G.5. In particular, we obtain

$$\sqrt{\Lambda_{i_1}} \leq \mathcal{O}\left(\frac{1}{\lambda_{g_{i_1}-1}}\right)\sqrt{\log T},$$

where $g_{i_1} = s_{i_1} + \Lambda_{i_1}$ is the time instant where the test succeeds for Agent 1, and λ_t denotes the dynamic gap. Hence, we have

$$R_1(i_1) \leq \mathcal{O}(k\sqrt{\Lambda_{i_1}\log T}) \leq \mathcal{O}\left(\frac{k\log T}{\lambda_{g_{i_1}-1}[1]}\right),$$

G.5.2. REGRET FOR AGENT 2 IN PHASE i_1

We now upper bound T_1,T_2 and T_3 separately. We first consider T_1.

We have

$$T_1 = \mathbf{1}(\text{Case-I})k\sqrt{\Lambda_{i_1}\log T} \leq k\sqrt{\Lambda_{i_1}\log T},$$

and using the same modified lemma as before, we obtain

$$T_1 \leq \mathcal{O}\left(\frac{k\log T}{\lambda_{g_{i_1}-1}[1]}\right),$$

where $\lambda_{g_{i_1}-1}[1]$ denotes the dynamic gap for player 1 at time instant $g_{i_1}-1$.

For T_2, we have

$$T_2 = \mathbf{1}(\text{Case -II})\left(k\sqrt{\tau_{n_{i_1}}\log T} + \sum_{t=s_{i_1}}^{s_{i_1}+\Lambda_{i_1}}\sum_{j=1}^{k}\mathbf{1}(j_t=j)\mathbf{1}\left(\tau_{n_{i_1}}^{(j)} < t\right)\right)$$
$$\leq \underbrace{\left(k\sqrt{\tau_{n_{i_1}}\log T}\right)}_{T_{2,1}} + \underbrace{\sum_{t=s_{i_1}}^{s_{i_1}+\Lambda_{i_1}}\sum_{j=1}^{k}\mathbf{1}(j_t=j)\mathbf{1}\left(\tau_{n_{i_1}}^{(j)} < t\right)}_{T_{2,2}}$$

The term $T_{2,1}$ can be bounded similar to Λ_{i_1}. This is the exploitation time of Agent 2 in the Explore all phase. Hence, it can be upper bounded as

$$T_{2,1} \leq \mathcal{O}\left(\frac{k\log T}{\lambda_{\tilde{g}_{i_1}-1}[2]}\right),$$

where $\tilde{g}_{i_1} = s_{i_1} + \tau_{n_{i_1}}$ is the time instant where the test succeeds for Agent 2, and$\lambda_{\tilde{g}_{i_1}-1}[2]$ denotes the dynamic gap for player 2 at time instant $\tilde{g}_{i_1}-1$.

Note that during the exploration phase of Agent 1, the arms are being played in a round robbin fashion, and hence

$$\begin{aligned}T_{2,2} &\leq \sum_{t=s_{i_1}}^{s_{i_1}+\Lambda_{i_1}}\sum_{j=1}^{k}\mathbf{1}(j_t=j)\mathbf{1}\left(\tau_{n_{i_1}}^{(j)}<t\right) \leq \sum_{t=s_{i_1}}^{s_{i_1}+\Lambda_{i_1}}\sum_{j=1}^{k}\mathbf{1}(j_t=j)\\ &\leq \sum_{j=1}^{k}\sum_{t=s_{i_1}}^{s_{i_1}+\Lambda_{i_1}}\mathbf{1}(j_t=j)\mathbf{1}\left(\tau_{n_{i_1}}^{(j)}<t\right)\\ &\leq \sum_{j=1}^{k}\frac{\Lambda_{i_1}}{k}=\Lambda_{i_1}.\end{aligned}$$

Hence, we have

$$T_{2,2}\leq \mathcal{O}\left[\left(\frac{1}{\lambda_{g_{i_1}-1}[1]}\right)^2\log T\right].$$

Combining $T_{2,1}$ and $T_{2,2}$, we have

$$T_2\leq \mathcal{O}\left[\left(\frac{k\log T}{\lambda_{\tilde{g}_{i_1}-1}[2]}\right)+\left(\frac{1}{\lambda_{g_{i_1}-1}[1]}\right)^2\log T\right]$$

Let us now control T_3. Note that during the exploitation phase of Agent 1, Agent 2 only incurs regret while exploring within the set of $[k]\setminus\{i^*\}$. So, using the Lemma G.5 now using on arm set $[k]\setminus\{i^*\}$ with cardinality $k-1$ is given by

$$\sqrt{\tilde{\tau}_p^{(i^*)}}\leq \mathcal{O}\left(\frac{(k-1)\sqrt{\log T}}{\lambda^{(i^*)}_{(g_{n_{i_1},p})-1}[2]}\right),$$

where $g_{n_{i_1},p}$ is the time instant where the test succeeds when Agent 2 is in p-th phase. Furthermore, since Agent 2 is not playing arm i^*, this regret depends on the dynamic gap excluding arm i^*, denoted by $\lambda^{(i^*)}_{(.)}$. Using this, we have

$$T_3=\sum_{p=n_{i_1}+1}^{n_{i_1}+N_{i_1}}(k-1)\sqrt{\tilde{\tau}_p^{(i^*)}\log T}\leq \sum_{p=n_{i_1}+1}^{n_{i_1}+N_{i_1}}\left(\frac{(k-1)\log T}{\lambda^{(i^*)}_{(g_{n_{i_1},p})-1}[2]}\right).$$

Combining T_1,T_2 and T_3, we obtain

$$\begin{aligned}R_2(i_1)&\leq \mathcal{O}\left[\left(\frac{k\log T}{\lambda_{g_{i_1}-1}[1]}\right)+\left(\frac{k\log T}{\lambda_{\tilde{g}_{i_1}-1}[2]}\right)+\left(\frac{1}{\lambda_{g_{i_1}-1}[1]}\right)^2\log T+\sum_{p=n_{i_1}+1}^{n_{i_1}+N_{i_1}}\left(\frac{(k-1)\log T}{\lambda^{(i^*)}_{(g_{n_{i_1},p})-1}[2]}\right)\right]\\ &\leq \mathcal{O}\left[\left(\frac{k\log T}{\lambda_{\tilde{g}_{i_1}-1}[2]}\right)+\left(\frac{1}{\lambda_{g_{i_1}-1}[1]}\right)^2 k\log T+\sum_{p=n_{i_1}+1}^{n_{i_1}+N_{i_1}}\left(\frac{(k-1)\log T}{\lambda^{(i^*)}_{(g_{n_{i_1},p})-1}[2]}\right)\right],\end{aligned}$$

since $\lambda_t\in[0,1]$. What remains is a bound on N_{i_1}.

G.6. Total Regret upto time T

In the above calculations, we have the regret for the i_1-th phase of Agent 1 only. Note that the starting instances of epochs for Agent 1, denoted by $\{s_{i_1}\}_{i_1=1,2,..}$ is random. To handle this issue, the learning epoch is split into several (deterministic) blocks and the total regret guarantee is given over these deterministic splits.

G.7. Total Regret for Agent 1

From Lemma G.6, the minimum length of an epoch of Agent 1 is given by $\Omega(\delta^{-2/3}k^{1/3}\log^{1/3}T)$. Motivated by this, we fix the deterministic blocks of length $\delta^{-2/3}k^{1/3}\log^{1/3}T$ so that each block can accommodate at most 2 phases. Using this, we write the regret of Agent 1 as

$$R_1 \leq C\sum_{\ell=1}^{m}\frac{1}{\lambda_{\min,\ell}[1]}\,k\log T,$$

where m denotes the number of blocks, each having length at most $\min\{c\ \delta^{-2/3}k^{1/3}\ \log^{1/3}\ T,\ T\}$, and $\lambda_{\min,\ell}[1] = \min_{t\in\ell\text{-th block}}\lambda_t$.

G.7.1. Total Regret of Agent 2

Now let us look at Agent 2. Note that in the exploitation phase of Agent 1, Agent 2 either plays on a constrained set with $k-1$ arms, or uses the optimistic estimates to exploit. So, using Lemma G.6, now on a constrained set of size $(k-1)$ the minimum length between 2 epochs of Agent 2 is given by

$$\Omega\Big(\delta^{-2/3}(k-1)^{1/3}\log^{1/3}T\Big).$$

Note that the since an entire phase of Agent 1, which includes exploration as well as exploitation is lower bounded by $\Omega(\delta^{-2/3}k^{1/3}\log^{1/3}T)$, trivially the exploitation phase is at least, $\Omega(\delta^{-2/3}k^{1/3}\log^{1/3}T)$. Hence the number of epochs played by Agent 2 for during the i_1-th phase of Agent 1 is given by

$$N_{i_1} \leq 2\times 2\times\left\lceil\left(\frac{k}{k-1}\right)^{1/3}\right\rceil.$$

We are now ready to write the total regret of Agent 2 upto time T. It is given by

$$\begin{aligned}R_2 \leq C_1\sum_{\ell=1}^{m}\Bigg\{&\left(\frac{1}{\lambda_{\min,\ell}[2]}\right)k\log T+\left(\frac{1}{\lambda_{\min,\ell}[1]}\right)^2 k\log T\\&+\left\lceil\left(\frac{k}{k-1}\right)^{1/3}\right\rceil\left(\frac{1}{\min_{a\in[k]}\lambda^{(a)}_{\min,\ell}[2]}\right)(k-1)\log T\Bigg\},\end{aligned}$$

where the number of blocks is denoted by m, each having length at most $\min\{c\delta^{-2/3}k^{1/3}\log^{1/3}T, T\}$, and

$$\lambda_{\min}[\ell] = \min_{t\in\ell\text{-th block}}\lambda_t.$$

Furthermore, $\lambda^{(a)}_{(.)}$ denotes the dynamic gap in the problem without arm a.

G.8. Event $\mathcal{E}$ and the total regret

All the above calculations are done conditioned on the good event. Now the total regret (of agent 1) is upper bounded by

$$(1-\mathbb{P}(\mathcal{E})R_1+\mathbb{P}(\mathcal{E}^c)\times T,$$

where the second term comes from the trivial fact that the rewards are within $[0,1]$, and so the trivial regret is T. Substituting the expression of $\mathbb{P}(\mathcal{E}^c)$ from Proposition G.2, we get the regret upper bound as

$$R_1+\frac{2Nk}{T}.$$

Note that the second term is decreasing with T and hence considered as the minor term with respect to R_1. So, we conclude that the overall regret is (order-wise) upper bounded by R_1.

Similar arguments and exact conclusion can be drawn for Agent 2 as well.

H. Proof of Theorem C.2; NSCB for N agents

In this theorem, we consider the generic case of N agents, and we characterize the regret of agent ranked r. We consider the learning of Agent $r-1$ as the action of Agent r will be dominated by that. The proof here follows in the same lines as of Theorem 4.5. The problem has an inductive structure, and this proof exploits that. We may only focus on the behavior of $r-1$-th agent; very similar to focusing on the first agent in the previous theorem.

H.1. Behavior of $r-1$-th ranked Agent

We consider 1 epoch of agent $r-1$. From the notation of Algorithm 3, it starts at $t_{i_{r-1}}$, and let the exploration period is $\Lambda_{i_{r-1}}$. Similarly, the exploitation period duration is $t_{i_{r-1}+1}-\Lambda_{i_{r-1}}$.

Note that if $r\geq 3$, the exploration of Agent $r-1$ will be restricted. Let $\mathcal{C}_t(r-1)$ be the set of arms dominated by agents ranked 1 to $r-2$, i.e., $|\mathcal{C}_t(r-1)|\leq r-2$. With this, for a fixed $\mathcal{C}_t(r-1)$, the dynamic gap parameter for Agent ranked $r-1$ is given by $\lambda_t^{\mathcal{C}_t(r-1)}[r-1]$. Note that when $\mathcal{C}_t(r-1)=\phi$, Agent $r-1$ will Explore all arms.

H.1.1. Regret of Agent r in explore phase of Agent $r-1$

As presented in the previous theorem, we break the regret of Agent r, during the exploration and the exploitation phase of agent $r-1$.

During the exploration phase, Agent r can either Explore all arms, or explore within a restricted set. Recall that $\mathcal{C}_t(r)$ denotes the set of arms dominated by agents ranked higher than Agent r. If $\mathcal{C}_t(r)=\phi$, Agent r explores all the arms. Otherwise it will explore the set of arms given by $[k]\setminus\mathcal{C}_t(r)$.

Similar to the 2 agent case, here also, Agent 2 will face *forced exploration*, and the definition is identical—instead of conditioning on the behavior of Agent 1, here, we condition on the behavior of Agent $r-1$.

Following the same lines, we obtain the regret of Agent r in the exploration phase of Agent $r-1$ is given by

$$\mathcal{O}\left[\left(\frac{(k-|\mathcal{C}_t(r-1)|)\log T}{\lambda^{\mathcal{C}_t(r-1)}_{g_{i_{r-1}}-1}[r-1]}\right)+\left(\frac{(k-|\mathcal{C}_t(r)|)\log T}{\lambda^{\mathcal{C}_t(r)}_{g_{i_r}-1}[r]}\right)+\left(\frac{1}{\lambda^{\mathcal{C}_t(r-1)}_{g_{i_{r-1}}-1}[r-1]}\right)^2(k-|\mathcal{C}_t(r-1)|)\log T\right]$$

where the time instances, $g_{i_{r-1}}$ denote the time the $(\tilde{\lambda},\mathcal{A})$ test succeeds for Agent $r-1$ with $\mathcal{A}=[k]\setminus\mathcal{C}_t(r-1)$. Similarly, g_{i_r} denote the time $(\tilde{\lambda},\mathcal{A})$ test succeeds for Agent r with $\mathcal{A}=[k]\setminus\mathcal{C}_t(r)$. Note that $|\mathcal{C}_t(r-1)|\leq r-2$ and $|\mathcal{C}_t(r)|\leq r-2$, since Agent $r-1$ has not committed yet. We upper bound the following as

$$\mathcal{O}\left[\left(\frac{(k-|\mathcal{C}_t(r)|)\log T}{\lambda^{\mathcal{C}_t(r)}_{g_{i_r}-1}[r]}\right)+\left(\frac{1}{\lambda^{\mathcal{C}_t(r-1)}_{g_{i_{r-1}}-1}[r-1]}\right)^2(k-|\mathcal{C}_t(r-1)|)\log T\right]$$

H.1.2. Regret of Agent r in exploit phase of Agent $r-1$

Similar to the behavior of Agent 2, in this case Agent r may be multiple epochs inside an exploration period of Agent $r-1$.

Note that inside the exploit phase of Agent 1, Agent 2 plays with the arm-set $[k]\setminus\mathcal{C}_t(r)$. Hence, mimicking the regret of Agent 2 as explained i the proof of Theorem C.2, the regret owing to exploitation is given by

$$\mathcal{O}\left(\sum_{p=i_r+1}^{i_r+N_{i_r}}\sqrt{(k-|\mathcal{C}_t(r)|)\tilde{\tau}_p^{(\mathcal{C}_t(r))}\log T}\right),$$

where N_{i_r} is the number of phases of Agent 2 in the current exploitation phase of Agent 1, and $\tilde{\tau}_j^{(\mathcal{C}_t(r))}$ is defined as the duration of the exploration period before the $(\tilde{\lambda},\mathcal{A})$ test succeeds with $\mathcal{A}=[k]\setminus\mathcal{C}_t(r)$.

We bound the above as

$$\mathcal{O}\left[\sum_{p=i_r+1}^{i_r+N_{i_r}}\left(\frac{(k-|\mathcal{C}_t(r)|)\log T}{\lambda^{\mathcal{C}_t(r)}_{(g_{i_r,p})-1}[2]}\right)\right].$$

We now need to bound N_{i_r}. Note that, when Agent 1 commits, $|\mathcal{C}_t(r-1)|=r-2$. As a consequence, using G.6 with the constrained set $\mathcal{C}_t(r-1)$, the minimum length of an episode for Agent $r-1$ is $\Omega(\delta^{-2/3}(k-r+2)^{1/3}\log^{1/3}T$. Hence, we have

$$N_{i_r}\leq 2\times 2\times\left\lceil\left(\frac{k-r+2}{k-r+1}\right)^{1/3}\right\rceil.$$

We now break the learning horizon into deterministic epochs. We use deterministic blocks of fixed length given by $\mathcal{O}(\delta^{-2/3}k^{1/3}\log^{1/3}T)$. Now, within one such block, the number of epochs of Agent $r-1$ is upper bounded by

$$\left\lceil\left(\frac{k}{k-r+2}\right)^{1/3}\right\rceil.$$

Hence, in one such deterministic block the regret of Agent r will be multiplied by the regret in one phase of Agent $r-1$ times the number of phases of Agent $r-1$.

H.1.3. Regret Expression

We are now ready to write the expression of regret for Agent r. In the above calculation, we work with a fixed constrained set $\mathcal{C}_t(r-1)$. We now extend the result uniformly for all constrained set here. We obtain

$$R_r\leq C\sum_{\ell=1}^{m}\Bigg\{\left(\frac{k}{k-r+2}\right)^{1/3}\left[\left(\frac{1}{\min\limits_{\substack{\mathcal{C}\in[k]\\|\mathcal{C}|\leq r-2}}\lambda^{\mathcal{C}}_{\min,\ell}[r]}\right)+\left(\frac{1}{\min\limits_{\substack{\mathcal{C}\in[k]\\|\mathcal{C}|\leq r-2}}\lambda^{\mathcal{C}}_{\min,\ell}[r-1]}\right)^2\right]k\log T$$
$$+\left\lceil\left(\frac{k-r+2}{k-r+1}\right)^{1/3}\right\rceil\left(\frac{1}{\min\limits_{\substack{\mathcal{C}\in[k]\\|\mathcal{C}|\leq r-1}}\lambda^{\mathcal{C}}_{\min,\ell}[r]}\right)(k-r+1)\log T\Bigg\},$$

where we now discuss several terms.

The term $\min\limits_{\substack{\mathcal{C}\in[k]\\|\mathcal{C}|\leq r-2}}\lambda^{\mathcal{C}}_{\min,\ell}[r]$ denotes the (worst-case) gap, of Agent r on a subset $\mathcal{C}$ of cardinality at most $r-2$. Note that this is an lower bound on the term $\lambda^{\mathcal{C}_t(r)}_{g_{i_r}-1}[r]$. Furthermore, since we do not have a lower bound on $|\mathcal{C}_t(r)|$, we upper bound $k-|\mathcal{C}_t(r)|$ as k.

Similarly, the second term comes from forced exploration. The final term also follows from the exploitation of Agent r. Here, $\min_{\substack{\mathcal{C}\in[k]\\|\mathcal{C}|\leq r-1}} \lambda^{\mathcal{C}}_{\min,\ell}[r]$ denotes the (worst-case) gap, of Agent r on a subset $\mathcal{C}$ of cardinality at most $r-1$. Note that this is an lower bound on the term $\lambda^{\mathcal{C}_t(r-1)}_{g_{i_r}-1}[r-1]$.

This above argument proves the theorem under the good event $\mathcal{E}$. Similar to the proof of Theorem C.2, we can extend the result to the total regret using Proposition G.2.

I. Proof of Lemma D.1

The proof comes from a reduction argument from the setup without blackboard to the setup with blackboard. Here, we obtain a sufficient condition on the drift δ', such that the dynamics "without blackboard" setup can be reduced to the problem setting of "with blackboard".

In the case of two agents, the proof for this reduction uses the following fact established in Section D: in the absence of the black-board, Agent 2 requires at-most k time-steps to infer the state of agent 1. Thus, if $\delta^{'} = Ck\delta$, then the deviation in arm-means in these k time steps before communication can occur is at-most δ'. This coincides with the deviation of the setting "with blackboard" where in one time-step Agent 2 learns of the state of Agent 1.

Hence, we can upper bound the performance here by a worse system with drift δ'. However, from the point of view of a δ' shifted system, the framework is equivalent to having a black board present.

Thus, the regret proofs for the case "without blackboard" are just corollaries of the regret proof "with blackboard" with δ'.

J. Proof of Lemma D.4

The proof of Lemma D.4 follows identical argument. With the modified reward model, we argue in Section D that it takes at most k time steps for Agent r to learn the arms that are being dominated by Agents ranked 1 to $r-1$. Hence, essentially, the framework is equivalent to the proof of Lemma D.1, and hence the lemma follows.