A Last Switch Dependent Analysis of Satiation and Seasonality in Bandits*

Pierre Laforgue pierre.laforgue@unimi.it
Dept. of Computer Science
Università degli Studi di Milano
Milan, Italy

Giulia Clerci giulia.clerici@unimi.it
Dept. of Computer Science
Università degli Studi di Milano
Milan, Italy

Nicolò Cesa-Bianchi nicolo.cesa-bianchi@unimi.it
Dept. of Computer Science
Università degli Studi di Milano
Milan, Italy

Ran Gilad-Bachrach rgb@tauex.tau.ac.il
Department of Bio-Medical Engineering
Tel-Aviv University, Israel
Edmond J. Safra Center for Bioinformatics
Sagol School of Neuroscience

Abstract

Motivated by the fact that humans like some level of unpredictability or novelty, and might therefore get quickly bored when interacting with a stationary policy, we introduce a novel non-stationary bandit problem, where the expected reward of an arm is fully determined by the time elapsed since the arm last took part in a switch of actions. Our model generalizes previous notions of delay-dependent rewards, and also relaxes most assumptions on the reward function. This enables the modeling of phenomena such as progressive satiation and periodic behaviours. Building upon the Combinatorial Semi-Bandits (CSB) framework, we design an algorithm and prove a bound on its regret with respect to the optimal non-stationary policy (which is NP-hard to compute). Similarly to previous works, our regret analysis is based on defining and solving an appropriate trade-off between approximation and estimation. Preliminary experiments confirm the superiority of our algorithm over both the oracle greedy approach and a vanilla CSB solver.

Keywords: Online learning, Non-stationary multi-armed bandits

1. INTRODUCTION

As the range of applications of multi-armed bandits increases, new algorithms able to deal with a wider range of phenomena must be designed and analyzed. When interacting with humans, one can often observe behaviours such as satiation and seasonality, that are hardly captured by stationary policies. Satiation typically occurs when making recommendations, e.g., for books, music, or movies (Kunaver and Požrl, 2017; Leqi et al., 2020). Stationary policies, which learn to recommend a single cuisine or musical genre, fail to capture the human desire for novelty (Dimitrijević et al., 1972; Kovacs et al., 2018). Another widespread type of nonstationary behaviour is seasonality. For example, certain music genres may be preferred during working hours, whereas chill playlists could be more appreciated on evenings or weekends (Schedl et al., 2018). Nonstationary behaviours also naturally occur in the medical domain.

*. This paper has been previously accepted at the *25th International Conference on Artificial Intelligence and Statistics* (AISTATS 2022).

Consider people suffering from various mental disorders, such as post-traumatic stress disorder, depression, or anxiety. Some remedies include micro-interventions (mindfulness, positive psychology exercises), Cognitive Behavioural Therapy, or Dialectical Behavioral Therapy (Meinlschmidt et al., 2016; Owen et al., 2018; Schroeder et al., 2018; Fuller-Tyszkiewicz et al., 2019). However, studies have shown that a diminishing return effect exists when the same intervention is applied repeatedly (Paredes et al., 2014), motivating the modelling of satiation effects.

In this work, we introduce the first bandit model which captures nonstationary phenomena that include satiation and seasonal effects. Unlike previous works, where the state structure is fully determined by how long ago each arm was last played, our notion of state keeps track of the time elapsed since an action took part in a switch (i.e., when the arm being pulled changes). This allows to modulate satiation effects at a level of detail not within reach of previous models. In addition, our model drops many assumptions on the shape of the expected reward function (such as concavity, Lipschitz continuity, monotonicity), while only retaining boundedness which is plausible in human interactions. Dispensing with monotonicity enables the modelling of periodicity, which is key to capture seasonality.

Technical contributions. The backbone of our analysis goes along the lines of previous works: (1) we show that computing the optimal policy (with value OPT) in our setting is NP-hard; (2) we prove that OPT is well approximated by a certain class of simple policies; (3) we show how to learn the best policy in the approximating class. Also similarly to previous works, we initially consider cyclic policies (with bounded block length) as approximating class. One of the main technical hurdles when learning a cyclic policy in a nonstationary setting is the calibration problem. Namely, the expected reward of the cycle block (i.e., the sequence of arm pulls that is being repeated in the cycle) depends on the current state of all the arms appearing in the block. A simple, yet impractical solution, is to force the learning algorithm to play every block twice in a row, where the first play of a block calibrates the reward estimates computed in the second play. Our solution is radically different: rather than forcing the algorithm to artificially play additional arms, we calibrate an arm by referring to the first time the arm is pulled in the block. These calibration pulls are not used to compute reward estimates, because their reward depends on the last time the arms took part in a switch among the blocks previously played by the learning algorithm. By ignoring calibration pulls, we underestimate the block reward by at most K (the number of arms), which in typical instances is much smaller than the block length.

Our calibration approach reduces the problem of learning the best calibrated block to that of solving an instance of Combinatorial Semi-Bandits (CSB), which we can solve using algorithms whose regret is well understood. As both the regret bound and the approximation factor for OPT depend on the block length d, we can then choose d to optimize our performance, thus obtaining a regret bound against OPT of order $KT^{3=4}$, ignoring log factors. Note that this does imply polynomial-time convergence to OPT, as solving the decoding problem (an intermediate step in the regret minimization procedure) for our instance of CSB requires solving an integer linear program, which is NP-hard in general. The decoding problem amounts to compute the reward-maximizing block given the current estimates of the state-value function (which we represent with a $K \times d$ table of reward estimates). In practice, we can approximately solve it through a branch-and-bound approach using a LP relaxation to bound the value of the objective. We implement an approximate solver based on this approach and run an experiment on a simple instance of LSD where our algorithm is seen to outperform two natural baselines.

Related works. Previous works addressed various extensions of bandits where rewards depend on past pulls. These include rested models (Gittins, 1979), such as Rotting Bandits (Bouneffouf and Féraud, 2016; Heidari et al., 2016; Cortes et al., 2017; Levine et al., 2017; Warlop et al., 2018; Seznec et al., 2019) and restless models (Whittle, 1988; Tekin and Liu, 2012). Similarly to Cella and Cesa-Bianchi (2020), our framework is neither rested nor restless, because the reward of arms that are pulled changes differently from the reward of arms that are not pulled. As a consequence, we highlight that we have a different definition for the regret, see (3). Indeed, (Tekin and Liu, 2012) compete against the best arm in hindsight, represented by the expected reward of the arm given the stationary distribution, which makes few sense in our setting since our transitions are pulls-dependent. Instead, we compete against the optimal trajectory in hindsight, that can be seen as a dynamic version of the restless regret, and is therefore a much harder comparator. Note however that we only allow for a single pull per time step. Unlike Cella and Cesa-Bianchi (2020), we do not assume a specific form for the reward function and so we can model satiation and seasonalities. Kleinberg and Immorlica (2018) also look at a similar model, but —unlike us— they assume the reward functions to be concave, increasing, and Lipschitz. Exploiting concavity (but not Lipschitzness) they prove that for every $0 < " < 1$ there exists a periodic schedule of length $T \geq K=$" whose asymptotic average reward is at least $(1 - ")\mathsf{OPT}$. Pike-Burke and Grunewalder (2019) also assume that the expected reward is a function of the time since the arm is last pulled, but consider reward functions drawn from a Gaussian Process with known kernel. Another relevant work is Simchi-Levi et al. (2021),

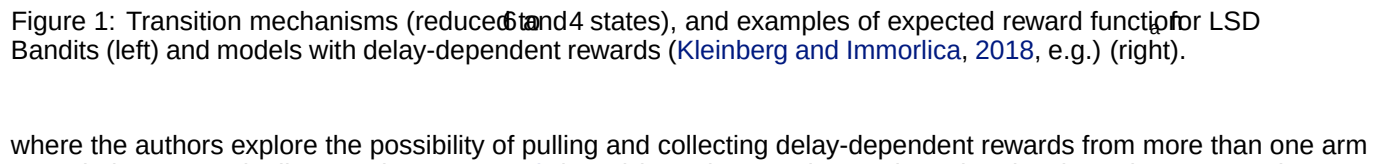

Figure 1: Transition mechanisms (reduced to 4 states), and examples of expected reward functions for LSD Bandits (left) and models with delay-dependent rewards (Kleinberg and Immorlica, 2018, e.g.) (right).

where the authors explore the possibility of pulling and collecting delay-dependent rewards from more than one arm at each time step. Finally, note that Basu et al. (2019) investigate an interesting related variant where an arm becomes unavailable for a certain amount of time after each pull. Although semantically close, we highlight that bandits with delayed feedback (Pike-Burke et al., 2018) are unrelated to our setting.

2. LSD BANDITS

Let A be an action set, with cardinality $K = |A|$. When necessary, individual actions are distinguished using superscripts, $A = \{a^{(1)}, \dots, a^{(K)}\}$, whereas subscripts are used for the temporal indexing, so that a_t refers to the action played at time step t. In our setting, the expected reward of an arm is assumed to be fully determined by the last time t it took part in a switch. That is, the last switch when the tuple (a_{t-1}, a_t) contains a exactly once. Therefore, every arm $a \in A$ is endowed with a state s_a (s_a actually depends on t, and we should use $s_a(t)$, but we drop this dependence whenever it is understood from the context) such that:

$$s_a(0) = 1 \quad \text{and} \quad s_a(t+1) = \sigma_a(s_a(t), a_t), \tag{1}$$

where σ_a is the transition function given by:

$$\sigma_a(s, a') = \begin{cases} 1 & \text{if } a' = a, s = 0 \\ 1 & \text{if } a' = a, s > 0 \\ -1 & \text{if } a' \neq a, s < 0 \\ s+1 & \text{if } a' \neq a, s \geq 0 \end{cases} \tag{2}$$

The transition graph (reduced to 6 states) is pictured in Figure 1 (bottom left). For each arm $a \in A$, the state s_a keeps track of the last time a switch of actions involved action a. A positive s_a means that the last switch involving arm a was $[a, \text{not } a]$, and occurred s_a time steps ago. In other words, arm a has not been played for the last s_a rounds. A negative s_a means that the last switch involving arm a was $[\text{not } a, a]$, and occurred $|s_a|$ time steps ago. In other words, arm a has been consistently played for the $|s_a|$ last rounds. We can now define Last Switch Dependent (LSD) Bandits, in which the expected reward of an arm only depends on its last switch state.

Definition 1 (LSD Bandit) A stochastic bandit with action set A is a LSD bandit if for every action $a \in A$ there exists an (unknown) function $\rho_a : \mathbb{Z}^* \to [0, 1]$, nondecreasing on $\mathbb{Z}^-$, such that the expected reward of arm a is given by $\rho_a(s_a)$, where s_a is the last switch state of a, as defined in (1) and (2).

Note that a LSD bandit is fully characterized by its reward functions $(\rho_a)_{a \in A}$ and the noise distribution. In the rest of the paper, we also use the following shortcut notations: $\rho_i := \rho_{a^{(i)}}$, $s_i = s_{a^{(i)}}$, and $s(t) = (s_a(t))_{a \in A}$. Given a

horizon T, the learner interacts with a LSD bandit as follows. First, the arm states are initialized. Then, for all time steps t from 1 to T:

1. the learner chooses an action $a_t \in A$,
2. the learner obtains a stochastic reward X_t with expected value $\mu_t := \mu_{a_t}(s_{a_t}(t))$,
3. states update: $\forall a \in A$, $s_a(t+1) = f_a(s_a(t); a_t)$.

The goal is to maximize the expected sum of rewards, and the performance is measured through the regret

$$R_T = \sum_{t=1}^{T} \mu_{a_t}(s_{a_t}(t)) - \mathbb{E}\left[\sum_{t=1}^{T} \mu_{a_t}(s_{a_t}(t))\right]; \tag{3}$$

where $[a_1 \dots a_T]$ is the optimal sequence of actions, i.e., the sequence maximizing the expected sum of rewards over steps $1, \dots, T$. LSD Bandits generalize bandits with delay-dependent rewards (Kleinberg and Immorlica, 2018; Pike-Burke and Grunewalder, 2019; Cella and Cesa-Bianchi, 2020) in two ways.

First, we emphasize that the notion of last switch generalizes the notion of delay. Whereas both definitions coincide on the positive values, last switches also allow to model progressive satiation, while delays cannot. Indeed, in the case where the same arm is pulled consecutively, the last switch keeps updating towards the negative values, see (2), thus allowing to associate different rewards with different numbers of consecutive pulls. On the contrary, no matter if the arm is pulled two or ten times in a row, the delay remains equal to 0 (or 1, depending on the convention), as only the fact that the same arm was pulled in the previous round matters. In other words, a bandit with delay-dependent rewards is a special case of LSD, where the reward function μ_a is constant over the negatives, while for a general LSD bandit it is only assumed to be nondecreasing—see Figure 1, top. Assuming μ_a to be nondecreasing on $\mathbb{Z}$ can be interpreted as a diminishing return requirement: the more the same arm is consecutively pulled, the smaller its reward gets (before resetting to the "default" value $\mu_a(1)$ when the series is interrupted). This is natural in most scenarios, and key to derive nonvacuous approximations (Table 1).

The second generalization regards the assumptions made on the values of $\mathbb{Z}^+$. In LSD bandits, μ_a is only supposed to be bounded. This assumption encompasses the framework of Cella and Cesa-Bianchi (2020), who considers a parametric class of bounded expected reward functions that are increasing and have bounded memory (their maximum is attained after a certain arm-dependent delay). The Recharging Bandit model of Kleinberg and Immorlica (2018), instead, assumes expected reward functions that are concave, increasing, Lipschitz continuous, but not bounded, making their framework and ours incomparable. We nonetheless highlight several limitations of their setting: (1) concavity excludes simple examples, such as $\mu_a(s) = \mathbb{I}\{s \geq \tau_a\}$ for some predefined threshold τ_a, (2) concavity prevents from modelling series of arm pulls, as a negative branch would force rewards to decrease unbounded, and (3) monotonicity prevents from modelling seasonality, see Figure 1, top. We finally highlight that relaxing the monotonicity of μ_a on $\mathbb{Z}^+$ unlocks many more applications than just modelling seasonality. Note that seasonality is understood with respect to the sequence of arm pulls. This matches the absolute time seasonality as soon as the learner is asked to make predictions at regular intervals.

Relaxing the concavity assumption has also important consequences on the difficulty of the problems which can be considered. For concave reward functions the oracle greedy policy provides a $1-1/e$ approximation of the optimal policy (Kleinberg and Immorlica, 2018), yet Example 1 shows that oracle greedy can be arbitrarily bad in our setting, highlighting that LSD Bandits can be more difficult than Recharging Bandits.

Example 1 Consider the LSD bandit defined by the following reward functions: $\mu_1(s) = \varepsilon + (1-\varepsilon)\mathbb{I}\{s = 1\}$, and $\mu_2(s) = 0 \ \forall s$. An oracle greedy strategy, which pulls at each time step the arm with highest expected reward (assuming the knowledge of the μ_a) would always pull arm $a^{(1)}$ obtaining a reward of $1 + \varepsilon(T-1)$ after T rounds. Instead, the optimal policy alternates between arms $a^{(1)}$ and $a^{(2)}$ and gets a $T/2$ overall reward. By making ε arbitrary close to 0, we can thus make oracle greedy arbitrarily bad. We conclude with a remark on why this example would be ruled out by concavity. In Kleinberg and Immorlica (2018), concavity is actually defined with respect to the origin, such that the reward obtained in case of consecutive pulls (here $\mu_1(1)$) is considered as an increment from 0. Concavity then prevents the next increments from being bigger, while here it is equal to 1, which is greater than as soon as $\varepsilon < 1/2$. And for $\varepsilon = 1/2$, one can note that oracle greedy is indeed $1/2$ approximation of the optimal strategy, as revealed by the above computations.

a. In case of a nondecreasing μ_a on $\mathbb{Z}^+$, the initialization $s_i(0) = +\infty$ would be the most sensible. In the absence of monotonicity however, all positive states become equivalent choices for the initialization, $s_i(0) = 1$ is one of them.

2.1 Hardness Results

Going beyond Example 1, we emphasize the difficulty of solving LSD bandits by showing several hardness results. All proofs are deferred to Appendix A.

Proposition 1 Computing the optimal policy for LSD bandits is NP-hard.

Since computing the global optimal policy is NP-hard even with full knowledge of the problem instance, a common approach then consists in learning the optimal policy within a simpler class of approximating policies. For LSD bandits, a natural approximating class is the set of policies with cyclic play sequences. A play sequence $a_1, a_2, \dots$ is cyclic if there exist t_0 and $d > 0$ such that $\forall t \geq t_0$ it holds that $a_{t+d} = a_t$. Cyclic policies can be shown to be optimal for 2-armed LSD bandits (Lemma 1), and give good constant factor approximations in general (Proposition 2).

Lemma 1 For a LSD bandit with two arms, any deterministic policy induces a sequence of pulls which is cyclic from a certain time step onward. This holds in particular for the optimal deterministic policy.

With 3 arms however, we can exhibit optimal policies which do not induce cyclic sequences. Note that this does not imply that no optimal policy is cyclic. Answering this question is however quite involved, and deferred to future work. In Proposition 2, we show a slightly weaker result, namely that cyclic policies can almost reach the best average reward, i.e., up to a term which is inversely proportional to the cycle length.

Proposition 2 Let $(\rho_i)_{i=1}^K$ be an LSD bandit with K arms and constant expected rewards $\in \mathbb{Z}$. Let $T > 0$ be the horizon, and $d > 0$ that divides T. Then, there exists a cyclic policy with cycle length d (and $t_0 = 1$) such that

$$\frac{1}{T}\sum_{t=1}^{T} r_\pi(t) \geq \frac{1}{T}\sum_{t=1}^{T} r_{\pi^*}(t) - \frac{K}{d} \tag{4}$$

where $r_\pi(t)$ and $r_{\pi^*}(t)$ are the expected rewards obtained at time step t by π and the optimal policy, respectively. When the ρ_i are not constant on $\mathbb{Z}^-$, (4) holds with $K+2$ instead of K in the right-hand side. Note also that (4) requires $d \geq K$ (respectively $d \geq K+2$) to be non-vacuous, as we have $\forall \pi, t$: $r_\pi(t) \leq 1$.

Thus, looking for the best cyclic policy simplifies the problem (as the search space is reduced), while maintaining a control on the approximation error via the cycle length. Unfortunately, this strategy is not viable either: it can be shown that finding the optimal cyclic policy for LSD bandits remains NP-hard, even when arms are totally ordered.

Proposition 3 Finding the optimal cyclic policy for LSD bandit problems is NP-hard, even with separable reward functions of the form $\rho_i(\cdot) = \rho_i^0(\cdot)$.

For $B = [a_1 \dots a_d]$, let $r(B \mid \mu_{init}) = \sum_{t=1}^{d} \mu_{a_t}(\rho_{a_t}(t))$ be the sum of the expected rewards obtained by playing the actions in B when μ is initialized to $\mu(1) = \mu_{init}$. Proposition 2 is essentially derived from the following key lemma, which exploits the boundedness assumption to relate $r(B \mid \mu)$ and $r(B \mid \mu')$ for different initial states.

Lemma 2 Let $(\rho_i)_{i=1}^K$ be an LSD bandit with K arms and constant expected rewards $\in \mathbb{Z}$. Then, for all block B and any initial states $\mu_{init}, \mu'_{init} \in \mathbb{Z}^K$, we have

$$r(B \mid \mu'_{init}) \geq r(B \mid \mu_{init}) - K. \tag{5}$$

If the ρ_i are not constant on $\mathbb{Z}^-$, a slight modification of B in the left-hand side is required to maintain a similar bound. Formally, for any block B of length d, there exists a block B' of length d such that for any initial states μ_{init}, μ'_{init}, we have

$$r(B' \mid \mu'_{init}) \geq r(B \mid \mu_{init}) - (K+2). \tag{6}$$

The approximation errors by block of size d that can be achieved depending on the assumptions made on ρ are summarized in Table 1. Relaxing the assumptions $\in \mathbb{Z}$ or ≥ 0 comes with a low price, while the monotonicity on $\mathbb{Z}^-$ is critical. The proof of Proposition 2 is provided in the appendix A.3. Here we show an example where our analysis is essentially tight.

Table 1: Approximation errors by block of size d when α is constant (—), nondecreasing ($\nearrow$), or non-monotone ($\circlearrowright$) on $\mathbb{Z}$ and $\mathbb{Z}^+$. The grey cell represents previous works. Our setting covers the blue ones. The red cells indicate vacuous approximations.

α	— on $\mathbb{Z}$	$\nearrow$ on $\mathbb{Z}$	$\circlearrowright$ on $\mathbb{Z}$
$\nearrow$ on $\mathbb{Z}^+$	$(K-1)$	$(K+1)$	d
$\circlearrowright$ on $\mathbb{Z}^+$	K	$(K+2)$	d

Example 2 Consider the LSD bandit defined by the following reward functions: $r_i(s) = \mathbb{1}\{s \geq K-1\}$ for all $i \leq K$. The optimal policy consists in repeating the block $[a^{(1)} \dots a^{(K)}]$ and obtains an average reward of 1. Let $d = 2K-1$, such that (up to permutations) we have $B = [a^{(1)} \dots a^{(K)} a^{(1)} \dots a^{(K-1)}]$. The average reward of B among the optimal sequences is 1, which is equal to the global average reward. Now, it is easy to check that playing repeatedly B yields an average reward of $K/(2K-1)$. On the other hand, the lower bound given by Proposition 2 is $1 - K/d = (K-1)/(2K-1)$, which matches the average reward as $K \to +\infty$.

3. PROPOSED APPROACH

We now introduce our approach to solve LSD bandits. It is based upon the CUCB-1 algorithm introduced in (Gai et al., 2012) to solve Combinatorial Semi-Bandits. Our adaptation is designed to cancel the interference created by the blocks previously played, and enjoys an approximation-estimation tradeoff which can be solved by an appropriate choice of the block size. In order to keep the exposition concise, we only focus here on the case where α is the constant on $\mathbb{Z}$. Results for the general case can be found in the Supplementary Material.

Approximation. A general idea that we can retain from Section 2.1, and from Proposition 2 in particular, is that approximating the optimal sequence using a series of smaller blocks of length $d < T$ is reasonable. However, the block B exhibited during the proof of Proposition 2 is seemingly impossible to estimate, as it requires the computation of the optimal sequence first. Another key challenge in LSD bandits is to handle the impact of the state: the same block played in different states may have different outcomes, making it difficult to identify "good blocks". One natural way to tackle this problem is to play every block twice: the rewards obtained during the second play are then fully representative of the block value, if repeated as a cycle. Let σ_B be the state reached by the system after a play of block B from initial state 1. We can then introduce

$$B_{\text{double}} = \operatorname*{argmax}_{|B|=d} r(B \mid \sigma_B) \,. \tag{7}$$

Playing cyclically B_{double} almost benefits from the same approximation guarantee as playing cyclically $B^\star$ (Proposition 4). However, solving (7) is nontrivial, as B influences both the reward-generating sequence and the initial state (it is essentially equivalent to compute the optimal cycle, which has been proven NP-hard in Proposition 3). Moreover, using an entire block simply to control the initial state of the system seems quite excessive, especially when d has to grow to control the approximation error. Instead, a cheaper way to calibrate the arms, i.e., to pull the arms before playing the real block such that the arms are in a controlled state independent of the past, consists of playing $B = [a^{(\sigma(1))} \dots a^{(\sigma(K))}]$ for any permutation $\sigma \in S_K$. Indeed, even if no reward can be exploited from B, it only takes K pulls to calibrate the system, while pre-playing the block is a $d > K$ long calibration. Let σ be the state reached by the system after a play of block B (note that it only depends on σ), we set

$$B_\sigma = \operatorname*{argmax}_{|B|=d} r(B \mid \sigma) \,. \tag{8}$$

A natural idea then consists in playing cyclically the block $[B_\sigma ; B_{\sigma'}]$ of size $K+d$, and approximation results (independent from σ) can be derived for this approach, see Proposition 4. Still, this strategy is not satisfactory for three reasons: (1) the approximation guarantee is worse than for double plays; (2) playing B_σ might still be highly inefficient: all arms are calibrated, while only those present in B would require calibration; (3) in domains like song recommendation, this would amount to regularly play a representative song of each genre. To remedy this issue, we introduce the following modified version of the block expected reward, which does not take into account the reward obtained if the arm is played for the first time in the block. Formally, for any block $B = [a_1 \dots a_d]$ we define

$$\tilde{r}(B) = \sum_{t=1}^{d} \mathbb{1}_{a_t}(a_t(t))\, \mathbb{1}\{\exists\, t_0 < t : a_{t_0} = a_t\}, \quad \text{and} \quad B^\star = \operatorname*{argmax}_{|B|=d} \tilde{r}(B) \,; \tag{9}$$

where t is indexed with respect to B and we note that B^{∞} is well defined since it is independent from the initial state. The rationale behind d is the following: first pulls do not provide reliable rewards because they are influenced by the past. Therefore, we use them as a calibration for the future pulls. So, only the arms which are used are calibrated. Observe also that calibration and exploitation phases might be intertwined, which was impossible with B^{∞}. But most importantly, the loss incurred by maximizing r_{avg} and not r is controllable, as we have for any block B and initial state μ_{init}: $r(B \mid \mu_{\mathrm{init}}) \geq K \geq r(B) \geq r(B \mid \mu_{\mathrm{init}})$. We can now state our complete approximation result.

Proposition 4 Let $(\mu_i)_{i=1}^{K}$ be a LSD bandit with K arms and constant expected rewards $\mathbb{Z}$ on. Let $T \geq 0$ be the horizon, and $d \geq 0$ that divides T. Let $r_{\mathrm{double}}(t)$ be the expected rewards obtained at time step t by the policy playing cyclically B_{double}. We have

$$\frac{1}{T}\sum_{t=1}^{T} r_{\mathrm{double}}(t) \geq \left(1 - \frac{d}{T}\right)\frac{1}{T}\sum_{t=1}^{T} r^{\star}(t) - \frac{K}{d}. \tag{10}$$

Let $\sigma \in \mathfrak{S}_K$, and assume that $d + K$ divides T. Let $r_{\sigma}(t)$ be the expected rewards obtained at time step t by the policy playing cyclically $[B_{\sigma}, B_{\sigma}]$. We have

$$\frac{1}{T}\sum_{t=1}^{T} r_{\sigma}(t) \geq \frac{d}{d+K}\left(\frac{1}{T}\sum_{t=1}^{T} r^{\star}(t) - \frac{K}{d+K}\right). \tag{11}$$

Let $e(t)$ be the expected rewards obtained at time step t by the policy playing cyclically B^{∞}. We have

$$\frac{1}{T}\sum_{t=1}^{T} e(t) \geq \frac{1}{T}\sum_{t=1}^{T} r^{\star}(t) - \frac{K}{d}. \tag{12}$$

The proof of Proposition 4 is similar to that of Proposition 2, and essentially combine Lemma 2 with the definitions of B_{double}, B_{σ}, and B^{∞}. Based on these results, using $\mathrm{alg}(B^{\infty})$ seems by far the best option. First, it is immediate to check that (12) is tighter than both (10) and (11) as soon as $(1 - \frac{d}{T})\sum_{t=1}^{T} r^{\star}(t) \geq K/d$, which is implicitly assumed for the bounds to be nonvacuous. Moreover, what cannot be seen from (10) is that computing a sequence of blocks with small regret against B_{double} requires to play each block twice, with no guarantee that the first play will provide any reward, thus dividing (10) by 2. On the contrary, as we see next, we can use the CombUCB1 algorithm of Gai et al. (2012) to estimate B^{∞} with tight regret bounds.

Estimation. A Combinatorial Semi-Bandit (CSB) —see, e.g., (Audibert et al., 2014)—is an online learning problem where, at each time step, the learner has to select a subset of N actions in a universe of $L > N$ base actions, under some combinatorial constraints. The individual rewards for each selected action are then revealed, and the learner receives their sum as total reward[b]. The regret is measured against the best subset of N base actions satisfying the constraints. Note for instance that a standard K-armed bandit is a particular instance of CSB, with $N = 1$, $L = K$, and no constraints. We now show that computing a series of blocks with small regret against B^{∞} with respect to e can be reduced to a CSB problem. In our case $N = d$ (the blocks contain d actions). The universe of base actions is however slightly more intricate to determine. While the analysis of CSB uses the fact that a block can only be filled with a subset of base actions, in our setting the same arm might be used several times in the same block. In addition to that, in CSB base actions have i.i.d. rewards, while the rewards of our arms also depend on the state. Our solution is to consider a universe of Kd^2 base actions: namely a base action is indexed by an arm $i \in \{1, \dots, K\}$, a state $\sigma \in \{1, \dots, d\}$, and a position $t \in \{1, \dots, d\}$ in the block. The state coordinate ensures the i.i.d. nature of the rewards, while the time coordinate allows to remove the arm multiplicities, therefore making the map from a block to its representation one-to-one. This is needed to extract valid sequences from solutions to (13), where the combinatorial constraints (ensuring that only subsets deriving from a sequence of pulls can be selected), are made explicit. Note that the structure of the problem plays a critical role here, as we derive an action space of size Kd^2, as opposed to K^d in the absence of any structure. From a pure estimation point of view, a space of size Kd is enough, as the rewards are independent of the position in the sequence. CombUCB1 is an algorithm introduced by Gai et al. (2012) to solve

b. In Chen et al. (2013) authors consider more general rewards, but the linear case is sufficient for our application

CSBs. Its analysis was later refined by Chen et al. (2013). Kveton et al. (2015, Theorem 6) prove a regret bound of order $O(\sqrt{NLn \log n})$, where n is the number of rounds. This bound can thus be applied to our setting in a black box fashion, with $n = T/d$, $L = d$, and $N = Kd^2$. The resulting algorithm, which we call ISI-CombUCB1 for Initial States Independent CombUCB1, produces a sequence of blocks with small regret against with respect to σ. Combining this result with Proposition 4, we can bound the regret of ISI-CombUCB1 with respect to OPT.

Theorem 1 Let $(\nu_i)_{i=1}^K$ be an LSD bandit with K arms and constant expected rewards in $[0,1]$. Let $T > 0$ be the horizon, and choose $d > 0$ that divides T. Then ISI-CombUCB1, run with block size d and exploration parameter $\alpha = 1.5$, has regret bounded by

$$R_T \leq \frac{KT}{d} + 47\, d \sqrt{KT \log \frac{T}{d}} + \left(\frac{2}{3} + 1\right) Kd^3 .$$

Choosing $d \propto T^{1/4}$, we obtain $R_T = \tilde{O}(KT^{3/4})$, where $\tilde{O}$ is neglecting logarithmic factors.

Note that the second claim of Theorem 1 requires a horizon-dependent tuning. One can use the doubling trick to make the algorithm anytime without harming the bound. Regarding the optimality of Theorem 1, we recall that our approximation result in $O(KT/d)$ is tight, see e.g., Example 2. As for the estimation part, Kveton et al. (2015) proved a lower bound for CombUCB1 that matches the upper bound up to a factor of $\sqrt{\log n}$, where n is the horizon. Instantiating this lower bound to our case, we obtain an overall lower bound of order $KT/d + d\sqrt{KT}$. Theorem 1 is thus tight up to a polylogarithmic factor of $\sqrt{\log(T/d)}$.

We now give more details about the implementation of ISI-CombUCB1, which specializes CombUCB1 to the minimization of σ. After an initialization step ensuring that each base action is played at least once, CombUCB1 maintains upper confidence bounds (UCBs) for each base action. At each round, the algorithm plays the admissible subset of actions with the biggest sum of UCBs, and updates the UCBs according to the individual rewards obtained. In ISI-CombUCB1, the optimization problem to select the best subset amounts to solve the right hand side of (9), but using the UCBs instead of the true expected rewards. As pointed out during the construction of the universe, in our case some actions share the same rewards, and it is actually enough to maintain Kd UCBs (one for each arm and each delay). Although Theorem 1 applies to this version of ISI-CombUCB1 as well, the need of taking multiplicities into account makes Kd^2 appear in the bound. This memory-efficient version, which is preferred in practice, is summarized in Algorithm 2a. Note that the initialization is such that pulling an uninitialized arm is always better than pulling any combination of initialized arms. As a consequence, the latter lasts at most Kd rounds, like in standard CombUCB1.

The key step in Algorithm 2a consists in computing the block b_t maximizing the current reward estimates. We now formally describe this optimization problem. Let $F \in \{0,1\}^{K \times d}$, such that $F[i,t] = 1$ if and only if arm i is played for the first time in the block at time step t. Let $Y \in \{0,1\}^{K \times d \times d}$, such that $Y[i,j,t] = 1$ if and only if arm i is played with delay j at time step t. Let $Z = (F, Y)$ be the Kd^2-sized representation we introduced earlier. Note that the column associated to $t = 1$ in Y is filled with zeros, as a pull here is by definition a first pull, and thus encoded in F. This notation of size $Kd(d+1)$ is however more convenient for coherence. The constraints on Z needed to describe a valid sequence of arm pulls are: Action Consistency (AC), i.e., at each time step, one and only one action is played, Unique First Pull (UFP), i.e., there is only one first pull per arm, First Pulls First (FPF), i.e., first pulls must precede any other pull of the same arm, and Time Consistency (TC), i.e., an arm can be pulled with a delay j at time step t only if it was pulled at time step $t - j$, and not pulled since. These constraints write (for index limits that make sense)

$$\forall t \quad \sum_i F[i,t] + \sum_{i,j} Y[i,j,t] = 1 \tag{AC}$$

$$\forall i \quad \sum_t F[i,t] \leq 1 \tag{UFP}$$

$$\forall i,t \quad \sum_j Y[i,j,t] \leq \sum_{s=1}^{t-1} F[i,s] \tag{FPF}$$

$$\forall i,j,t \quad Y[i,j,t] \leq F[i,t-j] + \sum_l Y[i,l,t-j] - \sum_s Y[i,j-s,t-s] \tag{TC}$$

The objective function is the sum of the current UCBs for the second actions present in the block, and writes $\sum_{i,j,s} Y[i,j,s]\, U_t(i,j)$. Noticing that all the relations are linear, we can derive $c_t \in \mathbb{R}^{Kd^2}$, $G \in \mathbb{R}^{Kd^2 + K \quad Kd^2}$, $h \in \mathbb{R}^{Kd^2+K}$, $A \in \mathbb{R}^{d \quad Kd^2}$, and $b \in \mathbb{R}^d$, such that the optimization problem writes as

$$\max_{z \in \{0,1\}^{Kd^2}} c_t^\top z \qquad \text{s.t.} \quad \begin{matrix} Gz & h \\ Az = & b \end{matrix} \tag{13}$$

where z is a vector version of Z. Note that (13) is an integer linear program, which is NP-hard to solve in general. This is expected, as our approach enjoys a sublinear linear regret against OPT (which is NP-hard to compute), and is therefore bound to be intractable. In Simchi-Levi et al. (2021), a Fully Polynomial-Time Approximation Scheme (FPTAS) is used to address a similar problem, see Lemma 6 therein. However, we highlight that, although similar at first sight, our two ILPs are fundamentally different. While the authors try to select the K best terms at a fixed time step, we aim at selecting an optimal block, that takes into account the evolution of the rewards among time. This time constraint is specific to our problem, and prevents from using standard FPTASs. Instead, we propose a heuristics based on a branch-and-bound-like approach (Land and Doig, 2010; Clausen, 1999), where we use a LP relaxation to estimate the value of the objective. This amounts to testing every admissible (discrete) first action, and then keeping the one maximizing the relaxed objective (in which all subsequent actions are relaxed in $[0,1]$). The same is repeated to choose the following $d-1$ discrete actions. Overall, we solve d Linear Programs, of sizes $Kd, 2Kd, \dots, Kd^2$. Given a horizon T, and choosing d as in Theorem 1, the total running time is $O(K^{5/2} T^{9/4})$. For more details about the heuristic, the reader is referred to Appendix B.3. Note that to ease readability we restricted ourselves to positive delays only in the core text. The complete resolution when negative delays are also considered is detailed in Appendix B.2.

Although it is generally hard to derive approximation guarantees, we point out that: (1) this approach works well in pratice, delivering the optimal solution in all the cases we tested; (2) as discussed in Kveton et al. (2015), if ISI-CombUCB1 is run with the approximate solver, Theorem 1 can be adapted to bound the regret against the best block according to the solver's approximation. Finally, note that our calibration approach is not affecting the complexity of finding the reward-maximizing block. Indeed, CombUCB1 is also required to solve an integer linear program similar to (13) to find the subset of base actions maximizing the block reward.

4. EXPERIMENTS

In this section, we benchmark ISI-CombUCB1 against vanilla CombUCB1 and OracleGreedy, which plays $\operatorname{argmax}_i \mu_i(t)$ at each time step t, and breaks ties randomly. We measure the algorithms performance in terms of the total cumulative reward, averaged over ten repetitions. For CombUCB1 and ISI-CombUCB1, the exploration parameter is set to 1.5, as in (Kveton et al., 2015). Let d be the block size of CombUCB1, which consequently maintains Kd UCBs. Whenever an arm a is pulled with state $s_a(t) < d$ (this might happen as the algorithm is not calibrated), we assume that the algorithm updates the estimate $\hat{\mu}_a(s_a(t))$ of — it actually becomes an estimate of $\mu_a(s_a(t))$. For ISI-CombUCB1 to similarly maintain Kd UCBs, we consider blocks of size $d+1$. Indeed, recall that rewards are ignored the first time an arm is pulled, so that reaching state $s_a = d$ requires (at least) $d+1$ steps. This "extra" first pull does not provide any information (it is only used for calibration purposes), but it is nevertheless considered in the performance. Note finally that both algorithms use the same heuristic based on branch-and-bound and LP relaxations to compute the reward-maximizing block at each time step. We consider the 5-armed LSD bandit with Bernoulli rewards and $\mu_1(s) = 0.95 \mathbb{1}\{s = 3\}$, $\mu_2(s) = 0.14 + 0.02 \mathbb{1}\{s = 6\} + 0.8 \mathbb{1}\{s \geq 9\}$, and $\mu_{3,4,5}(s) = 0.15$ for all s, see Figure 2b (top). The empirical results for $d = 3$ are reported in Figure 2b (bottom). ISI-CombUCB1 converges towards the block $[a^{(1)}, a^{(i)}, a^{(i)}, a^{(1)}]$, where i is any arm in $\{3,4,5\}$. Here, the first pull of arm $a^{(1)}$ is needed to calibrate the state of this action, and to get an expected reward $\mu_1(3)$ in the last pull of the block. On the first 8 time steps, OracleGreedy plays any constant arm, except at $t = 3$ and $t = 6$, where it plays $a^{(1)}$. At $t = 9$, it faces a choice between $\mu_1(3)$ and $\mu_2(9)$ and thus goes for the latter. Then, it will never be able to get $\mu_1(3)$ again, and will only get $\mu_2(9)$ every nine steps with any combination of constant actions in between. As for CombUCB1, it is unable to see the spike at $\mu_1(3)$. Precisely, whenever the algorithm plays an $a^{(1)}$ with state $s > 3$, the expected reward is small, but still contributes to the UCB of $\mu_1(3)$. This decreases the UCB for the arm, and the algorithm pulls it less and less frequently. More experiments testing the calibration sequence approach are provided in the Appendices.

input : number of arms K, block size d, horizon T

init : $\forall i \in [K], j \in [d]$, $T_1(i,j) = 0$, $\overline{X}_1(i,j) = 0$, $U_1(i,j) = +\infty$.

for t from 1 to T/d do

// Play best block for the UCBs

Play $B_t = [a_{t,1}, \dots, a_{t,d}]$ that maximizes $\sum_{s=1}^{d} U_t(a_s, a_s(s)) \mathbb{1}\{\forall s_0 < s : a_{s_0} \neq a_s\}$

Get rewards $X_t(a_{t,1}, a_{t,1}(1)), \dots, X_t(a_{t,d}, a_{t,d}(d))$

// Update the statistics

for $i \in [K]$ and $j \in [d]$ do

if arm i is played with delay j in block B_t then

$$T_{t+1}(i,j) = T_t(i,j) + 1,$$

$$\overline{X}_{t+1}(i,j) = \frac{T_t(i,j)\overline{X}_t(i,j) + X_t(i,j)}{T_{t+1}(i,j)}$$

else

$$T_{t+1}(i,j) = T_t(i,j); \quad \overline{X}_{t+1}(i,j) = \overline{X}_t(i,j)$$

$$U_{t+1}(i,j) = \overline{X}_{T_{t+1}(i,j)}(i,j) + \sqrt{\frac{\log(t+1)}{T_{t+1}(i,j)}}$$

(a) Algorithm ISI-CombUCB1.

(b) Reward functions with respect to (top) and cumulative rewards in thousands (bottom).

Acknowledgments

Acknowledgements This work has been partially supported by the academic grant Innovative Technologies, Interventions and Novel Approaches in the Field of Neuro-Wellness from Joy Ventures and by the EU Horizon 2020 ICT-48 research and innovation action under grant agreement 951847, project ELISE (European Learning and Intelligent Systems Excellence).

References

Martin S Andersen, Joachim Dahl, Lieven Vandenberghe, et al. CVXOPT: A Python package for convex optimization, 2013. Available at cvxopt.org .

Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. Regret in online combinatorial optimization. Mathematics of Operations Research, 39(1):31–45, 2014.

Amotz Bar-Noy, Randeep Bhatia, Joseph Naor, and Baruch Schieber. Minimizing service and operation costs of periodic scheduling. Mathematics of Operations Research, 27(3):518–544, 2002.

Soumya Basu, Rajat Sen, Sujay Sanghavi, and Sanjay Shakkottai. Blocking bandits. arXiv preprint arXiv:1907.11197, 2019.

Djallel Bouneffouf and Raphaël Féraud. Multi-armed bandit problem with known trend. Neurocomputing, 205:16–21, 2016.

Leonardo Cella and Nicolò Cesa-Bianchi. Stochastic bandits with delay-dependent payoffs. In International Conference on Arti cial Intelligence and Statistics, pages 1168–1177. PMLR, 2020.

Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In International Conference on Machine Learning, pages 151–159. PMLR, 2013.

Jens Clausen. Branch and bound algorithms-principles and examples. Department of Computer Science, University of Copenhagen, pages 1–30, 1999.

Corinna Cortes, Giulia DeSalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Discrepancy-based algorithms for non-stationary rested bandits. arXiv preprint arXiv:1710.10657, 2017.

Milan Radovan Dimitrijević, Janez Faganel, Matej Gregorič, PW Nathan, and JK Trontelj. Habituation: effects of regular and stochastic stimulation. Journal of Neurology, Neurosurgery & Psychiatry, 35(2):234–242, 1972.

Matthew Fuller-Tyszkiewicz, Ben Richardson, Vivienne Lewis, Jake Linardon, Jacqueline Mills, Kerry Juknaitis, Charlotte Lewis, Kim Coulson, Renee O'Donnell, Lilani Arulkadacham, et al. A randomized trial exploring mindfulness and gratitude exercises as ehealth-based micro-interventions for improving body satisfaction. Computers in Human Behavior, 95:58–65, 2019.

Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. IEEE/ACM Transactions on Networking, 20 (5):1466–1478, 2012.

John C Gittins. Bandit processes and dynamic allocation indices. Journal of the Royal Statistical Society: Series B (Methodological), 41(2):148–164, 1979.

Hoda Heidari, Michael J Kearns, and Aaron Roth. Tight policy regret bounds for improving and decaying bandits. In IJCAI, pages 1562–1570, 2016.

Robert Holte, Aloysius Mok, Louis Rosier, Igor Tulchinsky, and Donald Varvel. The pinwheel: A real-time scheduling problem. In Proceedings of the 22nd Hawaii International Conference of System Sciences, pages 693–702, 1989.

Robert Kleinberg and Nicole Immorlica. Recharging bandits. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 309–319. IEEE, 2018.

Geza Kovacs, Zhengxuan Wu, and Michael S Bernstein. Rotating online behavior change interventions increases effectiveness but also increases attrition. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW): 1–25, 2018.

Matevž Kunaver and Tomaž Požrl. Diversity in recommender systems–a survey. *Knowledge-based systems*, 123: 154–162, 2017.

Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvari. Tight regret bounds for stochastic combinatorial semi-bandits. In *Arti cial Intelligence and Statistics*, pages 535–543. PMLR, 2015.

Ailsa H Land and Alison G Doig. An automatic method for solving discrete programming problems. In *50 Years of Integer Programming 1958-2008*, pages 105–132. Springer, 2010.

Liu Leqi, Fatma Kilinc-Karzan, Zachary C Lipton, and Alan L Montgomery. Rebounding bandits for modeling satiation effects. *arXiv preprint arXiv:2011.06741*, 2020. To appear in NeurIPS 2021.

Nir Levine, Koby Crammer, and Shie Mannor. Rotting bandits. *arXiv preprint arXiv:1702.07274*, 2017.

Gunther Meinlschmidt, Jong-Hwan Lee, Esther Stalujanis, Angelo Belardi, Minkyung Oh, Eun Kyung Jung, Hyun-Chul Kim, Janine Alfano, Seung-Schik Yoo, and Marion Tegethoff. Smartphone-based psychotherapeutic micro-interventions to improve mood in a real-world setting. *Frontiers in psychology*, 7:1112, 2016.

Jason E Owen, Eric Kuhn, Beth K Jaworski, Pearl McGee-Vincent, Katherine Juhasz, Julia E Hoffman, and Craig Rosen. Va mobile apps for ptsd and related problems: public health resources for veterans and those who care for them. *Mhealth*, 4, 2018.

Pablo Paredes, Ran Gilad-Bachrach, Mary Czerwinski, Asta Roseway, Kael Rowan, and Javier Hernandez. Poptherapy: Coping with stress through pop-culture. In *Proceedings of the 8th International Conference on Pervasive Computing Technologies for Healthcare*, pages 109–117, 2014.

Ciara Pike-Burke and Steffen Grunewalder. Recovering bandits. *Advances in Neural Information Processing Systems*, 32:14122–14131, 2019.

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

Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, 7(2):95–116, 2018.

Jessica Schroeder, Chelsey Wilkes, Kael Rowan, Arturo Toledo, Ann Paradiso, Mary Czerwinski, Gloria Mark, and Marsha M Linehan. Pocket skills: A conversational mobile web app to support dialectical behavioral therapy. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2018.

Julien Seznec, Andrea Locatelli, Alexandra Carpentier, Alessandro Lazaric, and Michal Valko. Rotting bandits are no harder than stochastic ones. In *The 22nd International Conference on Arti cial Intelligence and Statistics*, pages 2564–2572. PMLR, 2019.

David Simchi-Levi, Zeyu Zheng, and Feng Zhu. Dynamic planning and learning under recovering rewards. In *International Conference on Machine Learning*, pages 9702–9711. PMLR, 2021.

Cem Tekin and Mingyan Liu. Online learning of rested and restless bandits. *IEEE Transactions on Information Theory*, 58(8):5588–5611, 2012.

Romain Warlop, Alessandro Lazaric, and Jérémie Mary. Fighting boredom in recommender systems with linear reinforcement learning. In *Neural Information Processing Systems*, 2018.

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

Appendix A. TECHNICAL PROOFS

In this section, we gather all the technical proofs omitted in the main body of the paper.

A.1 Proof of Proposition 1

First note that focusing on the case where the μ_i are constant or Z is enough to prove NP-hardness. Within this setting, we actually prove a slightly more general result than Proposition 1, namely that for any $\nu_{init} \in \mathbb{N}^K$, computing

$$B^* = \underset{B,\, |B| = d}{\operatorname{argmax}}\ r(B \mid \nu_{init}) \tag{14}$$

is NP-hard as soon as d is greater than a certain value d_0 precised later on. In particular, when $\nu_{init} = 1$, and $d = T$, solving (14) amounts to find the best policy. For some T large enough, we get $d \geq d_0$, and we obtain that computing the optimal policy is NP-hard (Proposition 1). But more interestingly, this result also highlights that the inner optimization problem of CombUCB1 approach is itself NP-hard. Note that the same result holds for the inner optimization problem of SI-CombUCB1, as the hardness can be similarly proved with $\tilde{r}$ instead of r in (14).

Our reduction is largely adapted from the one in (Basu et al., 2019). Indeed, our framework encapsulates the setting described by Equations (15), which is the one used by the authors to show the hardness of their problem. The main differences are: 1) we need to take into account the initial state ν_{init}, which can be handled by considering a larger block size d, and 2) in (Basu et al., 2019), any empty slot in the scheduling is automatically filled with a pull of the 0 arm (it is the only arm which can be played), while we have to invoke a finer argument, namely that at any empty slot, playing the 0 arm is the best possible action, since all actions yield a null expected reward, but playing the 0 arm allows every other arms to recharge.

As in (Basu et al., 2019), we thus consider the Pinwheel Scheduling Problem (PSP), see (Holte et al., 1989), which can be stated as follows. Given a set of actions $\{1, \dots, K\}$, associated to delays $(d_i)_{i=1}^K \in \mathbb{N}^K$, the PSP consists in deciding whether it is possible or not to design an action scheduling, i.e., a mapping σ from $\{1, \dots, K\}$, such that each arm i is played at least once in any sequence of d_i consecutive pulls. A PSP instance with such a scheduling is called a YES instance. Otherwise, it is referred to as a NO instance. Furthermore, a PSP instance is said to be dense if $\sum_{i=1}^K 1/d_i = 1$. In particular, this implies for YES instances that each arm is played exactly every d_i pulls. It has been shown in (Bar-Noy et al., 2002) that PSP on dense instances is NP-complete. In the next paragraph, we show that PSP on dense instances reduces to particular instances of our problem, described by Equations (15), proving the hardness of our problem.

Given a dense PSP instance, we construct an instance of our problem as follows. For every $i \in \{1, \dots, K\}$, we set:

$$\mu_i(\cdot) = \mathbb{1}\{\cdot = d_i\}. \tag{15}$$

We set an additional arm $K+1$, such that $\mu_{K+1}(\cdot) = 0$ for all $\cdot$. Given an initial state ν_{init}, we want to find the best block of actions $B^* = \operatorname{argmax}_{B,\, |B| = d} r(B \mid \nu_{init})$. Then, we have two possibilities:

- if the PSP is a YES instance, the PSP schedule gives a reward at each time step. Taking into account the fact that the initial state might not exactly suit the schedule, we obtain $r(B^* \mid \nu_{init}; \text{YES}) \geq d - K$.
- if the PSP is a NO instance, we can use the same argument as in (Basu et al., 2019) and prove that $r(B^* \mid \nu_{init}; \text{NO}) \leq d \left\lfloor \frac{d}{\sum_{i=1}^K d_i} \right\rfloor$.

Hence, for $d \geq d_0 := (K+1) \prod_{i=1}^K d_i$, if we were able to compute B^*, and therefore $r(B^* \mid \nu_{init})$, we could discriminate YES from NO instances, depending on whether $r(B^* \mid \nu_{init}) \geq d - K$ or not. Solving PSP on dense instances thus reduces to solving (14) for the particular choice of reward functions explicited in (15), therefore proving the hardness of the latter problem. ∎

A.2 Proof of Lemma 1

Assume without loss of generality that $a^{(1)}$ is the first action played. If there is no switch of actions, then $a^{(1)}$ is played indefinitely, which means that the sequence is cyclic. If there is only one (respectively two) switch(es) of actions, then

$a^{(2)}$ (respectively $a^{(1)}$) is played indefinitely after some time step, from which the sequence is cyclic. If there are three (or more) switches of actions, then the sequence witnesses twice the switch of actions $[a^{(1)}, a^{(2)}]$, and thus visits twice the state $(\ _1;\ _2) = (1;\ 1)$. Since the policy is deterministic, the same cycle will be repeated. ■

Note however that this property does not resist to the number of arms, as we can exhibit optimal policies with 3 arms that never visit the same state twice. Let $= \{a^{(1)}; a^{(2)}; a^{(3)}\}$, and consider the sequence of actions built as follows.

For $m = 1; 2; \ldots$ play

- $a^{(1)}$ m times consecutively,
- $a^{(2)}$ m times consecutively,
- $a^{(3)}$ m times consecutively.

Assuming for simplicity that the states are initialized to 0 and not to 1 (this only impacts the states in the first block), it can be checked that, in block k, the state $(\ _1;\ _2;\ _3)$ is given by

$$\begin{cases} (\ t;\ m\ \ 1 + t;\ t) & \text{after the } t^{\text{th}} \text{ pull of } a^{(1)} \\ (t;\ \ \ t;\ m + t) & \text{after the } t^{\text{th}} \text{ pull of } a^{(2)} \\ (m + t;\ t;\ \ \ t) & \text{after the } t^{\text{th}} \text{ pull of } a^{(3)} \end{cases}$$

From the above formula, it is immediate to check that no state is visited twice by the sequence. Now it is easy to choose $_1;\ _2;\ _3$ such that this policy is optimal. For example, $_i\ \ 1$ for $i = 1; 2; 3$.

A.3 Proof of Proposition 2

We first provide a proof for the special case where the $_i$ are constant on $\mathcal{Z}$.

Note that there exists a time step $t_0\ \ T\ \ d + 1$ such that $(1=d)\sum_{t=t_0}^{t_0+d\ \ 1} r\ (t)\ \ (1=T)\sum_{t=1}^{T} r\ (t)$. Let $B = [a_{t_0} \ldots a_{t_0+d\ \ 1}]$, and $\ (t)$ be the sequence of states generated by the optimal policy, such that $\sum_{t=t_0}^{t_0+d\ \ 1} r\ (t) = r(B\ \ j\ \ (t_0))$. The policy consists in playing repeatedly block B. However, except for the first d steps, B is played by with an initial state $_B$, i.e., the state reached by the system after a play of B, which might be different from (t_0). Applying Lemma 2, and assuming for simplicity that states were initialized to $_B$, we obtain

$$\frac{1}{T}\sum_{t=1}^{T} r\ (t) = \frac{r(B\ \ j\ \ _B)}{d}\ \ \frac{r(B\ \ j\ \ (t_0))\ \ K}{d}$$

$$\frac{1}{T}\sum_{t=1}^{T} r\ (t)\ \ \frac{K}{d}:$$

If the $_i$ are not constant on $\mathcal{Z}$, the repeated block is built using the second part of Lemma 2.

The proof in the general case is very similar, but uses the second claim of Lemma 2, and more specifically the way B' is constructed from B.

Similarly to the constant case, we know that there exists a block B of length d with average reward (at least) greater than the average reward of the complete optimal sequence. Recalling the notation from the proof in the constant case, we have $B = [a_{t_0} \ldots a_{t_0+d\ \ 1}]$ and

$$\frac{r(B\ \ j\ \ (t_0))}{d}\ \ \frac{1}{T}\sum_{t=1}^{T} r_t :$$

Previously, we could simply repeat B, which is not possible anymore due to the possible effects showed in (25). Let B', derived from B as in the proof of Lemma 2. Namely, if $a^{(1)}$ and $a^{(2)}$ are the first two different actions played in B,

we have $B^0 = [a_{t_0}\ a^{(2)}\ a_{t_0+1}\ \dots\ a_{t_0+d-1}]$. Let $_{B^0}$ be the state reached after a play of B^0 from initial state 1. Using Lemma 2, the expected rewards (t) obtained by policy which plays cyclically B^0 satisfy

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} r\ (t) &= \frac{r(B^0 j\ 1)}{T} + \frac{T\quad d}{T}\,\frac{r(B^0 j\ _{B^0})}{d} \\
&= \frac{d}{T}\,\frac{r(B^0 j\ 1)}{d} + \quad 1\quad \frac{d}{T}\quad \frac{r(B^0 j\ _{B^0})}{d} \\
&\quad \frac{d}{T}\,\frac{r(B\quad j\quad (t_0))\quad (K+2)}{d} + \quad 1\quad \frac{d}{T}\quad \frac{r(B\quad j\quad (t_0))\quad (K+2)}{d} \\
&= \frac{r(B\quad j\quad (t_0))\quad (K+2)}{d} \\
&\quad \frac{1}{T}\sum_{t=1}^{T} r_t \qquad \frac{K+2}{d}\ :
\end{aligned}$$

■

A.4 Proof of Proposition 3

Similarly to Proposition 1, we only need to focus on the case where the reward functions are constant, and actually prove here a stronger result than Proposition 3. Namely, we show that for some computing

$$B\quad = \operatorname*{argmax}_{jB j= d} r(B\ j\ _B) \tag{16}$$

where $_B$ is the state reached by the system after a play of block B, is NP-hard, even when the reward functions can be totally ordered. The optimal cyclic policy (with cycle length d) is obtained by repeating inde nitely the solution to (16), and the NP-hardness of the latter problem then yields Proposition 3.

This proof is also based on a reduction of the Pinwheel Scheduling Problem (PSP), see Appendix A.1. Given a PSP dense instance $(d_i)_{i=1}^{K}$, we construct an instance of our problem as follows. For every $i = 1; \dots; K$, we set:

$$_i(\ ) = \sqrt{\ =d_i}\ : \tag{17}$$

Note that setting (17) is fundamentally different from setting (15) as we have a total ordering of the arms, i.e., for all we have $_1(\ )\quad _2(\ )\quad \dots\quad _K(\ )$, with the convention $d_1\quad d_2\quad \dots\quad d_K$. We also highlight that we do not introduce an additional null arm here, as opposed to (17). Finally, although $_i$ are unbounded for the moment, thus breaking De nition 1, we see at the end of the proof how to consider bounded $_i$ without altering the subsequent analysis.

We now show that solving (16) with reward functions (17) allows us to determine if the PSP instance is a YES or a NO instance. Let n_i be the number of times action $i = 1; \dots; K$ is played in the block, and $_j$, for $j = 1; \dots; n_i$ be the different states in which arm i is pulled. Problem (16) is equivalent to:

$$\max_{(n_i)_{i=1}^{K}}\ \max_{(\ _{ij})_{i=1;j=1}^{K;n_i}}\ \sum_{i=1}^{K}\sum_{j=1}^{n_i} \sqrt{\ \ =d_i} \qquad \text{subject to} \qquad \begin{cases} \sum_{i=1}^{K} n_i = d \\ \sum_{j=1}^{n_i}\ _{ij} = d \quad i = 1; \dots; K \end{cases} \tag{18}$$

We start by maximizing with respect to the $= (\ _{ij})_{i=1;j=1}^{K;n_i}$. The Lagrangian writes

$$L(\ ;\ ) = \sum_{i=1}^{K}\sum_{j=1}^{n_i} \sqrt{\ \ =d_i} + \sum_{i=1}^{K}\ _i\left(\sum_{j=1}^{n_i}\ _{ij}\quad d\right) :$$

The KKT conditions (gradient of the Lagrangian and primal feasibility) write

$$\frac{\partial \mathcal{L}(\eta, \lambda)}{\partial \eta_{ij}} = -\frac{1}{2}\sqrt{\frac{1}{d_i \eta_{ij}}} + \lambda_i = 0 \qquad i = 1, \dots, K;\ j = 1, \dots, d \tag{19}$$

$$\sum_{j=1}^{n_i} \eta_{ij} = d \qquad i = 1, \dots, K. \tag{20}$$

Solving (19) for η_{ij} we obtain a quantity independent of j. Then (20) implies that $\eta_{ij} = d/n_i$ for each $i = 1, \dots, K$. Replacing $\eta_{ij} = d/n_i$ into (18), we can now maximize with respect to the n_i. The Lagrangian writes

$$\mathcal{L}(n, \lambda) = \sum_{i=1}^{K} \sqrt{\frac{d}{n_i d_i}} + \lambda \left(\sum_{i=1}^{K} n_i - d \right)$$

and the KKT conditions (gradient of the Lagrangian and primal feasibility) are

$$\frac{\partial \mathcal{L}(n, \lambda)}{\partial n_i} = -\sqrt{\frac{d}{4 n_i d_i}} + \lambda = 0 \qquad i = 1, \dots, K \tag{21}$$

$$\sum_{i=1}^{K} n_i = d \tag{22}$$

such that replacing (21) into (22), we obtain $n_i = d/d_i$, which implies $\eta_{ij} = d_i$.

Assume now that d can be divided by all the d_i, such that $n_i = d/d_i \in \mathbb{N}$. From the values of n_i and η_{ij} obtained, we can see that the optimal block for (16) corresponds to a Pinwheel schedule. It yields an average reward of 1, and is achievable if and only if the Pinwheel instance is a YES instance. Therefore, if we can solve (16), we can tell if the average reward is equal to 1 or smaller, and thus decide whether the instance is a YES or NO instance. We have reduced PSP on dense instances to (16), which is therefore shown to be NP-hard. To our knowledge, this is the first hardness result for decomposable reward functions of the form $\rho(m) = \sum_i^0 (\,)$.

Now, let $d_{\max} = \max_{i=1,\dots,K} d_i$. Note that replacing (17) with the bounded functions

$$\sum\nolimits_i^0(\,) = \begin{cases} \sqrt{1/d_i} & \text{if } d_i \geq d_{\max} \\ \sqrt{d_i/d_{\max}^2}\,/\,d_i & \text{otherwise} \end{cases}$$

does not change change the optimal schedule (arm i played every $d_i \geq d_{\max}$ time steps) and the analysis is unchanged. ∎

A.5 Proof of Lemma 2

First, we provide the proof for the the special case where the d_i are constant over $\mathbb{Z}$. Note that after the first play of an arm a, the switch $[a, \text{not } a]$ occurs, and the arm goes to $s_a = 1$, independently of its initial state. Now, what happens during the first play of a? Let t_a be the time step at which arm a is played for the first time. Even if there are many consecutive pulls of a, the rewards collected for s_{init} and s'_{init} are $\nu_a(s_{\text{init};a} + t_a - 1), \nu_a(1), \dots, \nu_a(1)$ and $\nu_a(s'_{\text{init};a} + t_a - 1), \nu_a(1), \dots, \nu_a(1)$ because the d_i are constant over $\mathbb{Z}$. Thus, the only difference is $\nu_a(s_{\text{init};a} + t_a - 1) - \nu_a(s'_{\text{init};a} + t_a - 1) \leq 1$. Over the K arms, the total difference cannot exceed K, giving (5). Note that in full generality, K could be replaced by the number of different arms played in B.

Now we focus on the general version only, when the d_i are not constant over $\mathbb{Z}$ and a finer comparison of the first pulls is required. For any block B of size d, we want to find $B' \in \mathcal{B}$ of length d such that for any initial states $s_{\text{init}}, s'_{\text{init}} \in \mathbb{Z}^K$, we have

$$r(B' \mid s'_{\text{init}}) \geq r(B \mid s_{\text{init}}) - (K + 2). \tag{23}$$

First, we analyze what happens if we keep the same B instead of B', in the left-hand side of (23). Similarly to the constant case, it is important to note that an action is impacted by a change of initial state only the first time it is

played in the block (possibly with several consecutive pulls). Indeed, when this sequence (which might be of length 1 if a switch follows the first pull) is interrupted, the arm goes to state 1, no matter the state it was before, and similar subsequent actions then yield similar states/rewards. A second point to notice is that only the first action in the block can be pulled with a negative state at the beginning of its sequence of pulls. Indeed, let $(_{new,a}, a_1)$, where $_{new,a}$ is a componentwise version of $_a$, and a_1 is the first action played in B. It is easy to check that for all $a \neq a_1$, we have $_{new,a} \geq 1$ (either the action was in a negative state and was set to 1 as $a_1 \neq a$ has been played, or it was in a positive state and incremented by 1). Combining these two remarks, we know that for any action a different from $a^{(1)}$ (we can assume that $a_1 = a^{(1)}$ without loss of generality), the loss incurred by a change of initial state due to the pulls of a is bounded by 1. Indeed, the expected rewards obtained during the first play (possibly with consecutive pulls) with initial states $_{init}$ and $^0_{init}$ are respectively

$$_a(\ _a);\ \ _a(\ _a+1);\ \ _a(\ _a+2);\ :::\quad \text{and}\quad _a(\ ^0_a);\ \ _a(\ ^0_a+1);\ \ _a(\ ^0_a+2);\ :::\tag{24}$$

where $_a$ and ^0_a are two generic positive values (thanks to the remark we made earlier) that depend on $_{init,a}$, $^0_{init,a}$, and the place of the first pull of a in the block. Making these values explicit is not important here, as the boundedness of $_a$ ensures anyway that the difference of expected rewards obtained is smaller than 1.

Now, what happens for the first pulls of $a^{(1)}$? We assume that it is played n_1 times consecutively at the beginning of B (again, we might have $n_1 = 1$). Assuming that $_{init,1}$ is positive and $^0_{init,1}$ is negative, the expected rewards collected are respectively

$$_1(\ _{init,1});\ \ _1(\ _1+1)\ :::\ \ _1(\ _1+n_1-1)\quad \text{and}\quad _1(\ ^0_{init,1});\ \ _1(\ ^0_{init,1}+1)\ :::\ \ _1(\ ^0_{init,1}+n_1-1)\ :\tag{25}$$

Unlike in the previous case, the difference between these two sequences cannot be bounded (it might even be equal to n_1 if $^0_{init,1} \geq n_1$). This is why the same B cannot be used on both sides of (23). To break this sequence of pulls, we define B' as follows. Without loss of generality, let $a^{(2)}$ be the second different action played in B (if B is only composed of pulls of $a^{(1)}$, we can set $a^{(2)}$ to be any action of A different from $a^{(1)}$). The block B' is equal to B, except that the second pull in B' is necessarily $a^{(2)}$. We may now face 3 different cases.

- $n_1 = 1$. Note that here $B' = B$. Since $n_1 = 1$, the difference between the two sequences of rewards due to the pulls of $a^{(1)}$ is at most 1. For all other actions, we can use the analysis we developed at the beginning, and the total difference is at most K.
- $n_1 = 2$. Then, denoting by n_2 the number of times $a^{(2)}$ is played consecutively after $a^{(1)}$, the expected rewards obtained by B with initial state $_{init}$ and by B' with initial state $^0_{init}$ are respectively

  $$_1(\ _{init,1});\ \ _1(\ _1+1) \text{ or } _1(\ _{init,1}-1);\ \ _2(\ _2);\ \ _2(\ _2+1)\ :::\ \ _2(\ _2+n_2-1)$$
  $$\text{and}\quad _1(\ ^0_{init,1});\ \ _2(\ ^0_2);\ \ _2(\ ^0_2+1);\ \ _2(\ ^0_2+2)\ :::\ \ _2(\ ^0_2+n_2-1)$$

  where the or comes from the fact that we don't know if $_{init,1}$ is positive (then the next reward is $_1(\ _1+1)$) or negative (then next reward is $_1(\ _{init,1}-1)$). Here, $_2$ and $^0_2$ are two generic positive numbers, whose values are not important as the difference between the two sequences is contained in the red rewards, and thus bounded by 3 anyway. For all other $K-2$ actions, we can apply the standard analysis, such that in total the difference cannot exceed $K+1$.
- $n_1 \geq 3$. Using the same notation as above, we have now respectively for the n_1 pulls

  $$_1(\ _{init,1});\ \ _1(\ _1+1) \text{ or } _1(\ _{init,1}-1);\ \ _1(\ _1+2) \text{ or } _1(\ _{init,1}-2);\ :::\ \ _1(\ _1+n_1-1) \text{ or } _1(\ _{init,1}-n_1+1)$$
  $$\text{and}\quad _1(\ ^0_{init,1});\ \ _2(\ ^0_2);\ \ _1(1);\ :::\ \ _1(\ _1+n_1-3)$$

  The red terms incur a loss of at most 3. Then, if $_{init,1} \geq 1$, the remaining rewards (non red) in the play of B are $\sum_{j=3}^{n_1-1} {}_1(\ _1+j)$ $\sum_{j=1}^{n_1-3} {}_1(\ _1+j)$ as $_1$ is nondecreasing on $\mathbb{Z}$, so the rewards obtained by B' are greater. If $_{init,1} < 0$, we have $\sum_{j=3}^{n_1-1} {}_1(\ _{init,1}-j)$ $\sum_{j=1}^{n_1-3} {}_1(\ _1+j)$ for the same reasons. So overall, the difference is bounded by 3.

  As for the n_2 following pulls, recalling that $_2$ and $^0_2$ are positive since the preceding pulls are $a^{(1)}$, we have

  $$_2(\ _2);\ \ _2(\ _2+1);\ :::\ \ _2(\ _2+n_2-1)$$
  $$\text{and}\quad _2(\ ^0_2);\ \ _2(\ ^0_2+1);\ :::\ \ _2(\ ^0_2+n_2-1)$$

  with a difference bounded by 1. For the $K-2$ other actions, we still have the bound of 2, so that in total the difference does not exceed $K+2$. ■

A.6 Proof of Proposition 4

Recall the notation B_{double}, B, and $\Delta(t_0)$ introduced in the proof of Proposition 2 in the main paper. We have

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} r_{\text{double}}(t) &= \frac{r(B_{\text{double}} \mid 1)}{T} + \frac{T-d}{T}\,\frac{r(B_{\text{double}} \mid B_{\text{double}})}{d} \\
&\geq 1 - \frac{d}{T}\,\frac{r(B_{\text{double}} \mid B_{\text{double}})}{d} \\
&\geq 1 - \frac{d}{T}\,\frac{r(B \mid B)}{d} && (26) \\
&\geq 1 - \frac{d}{T}\,\frac{r\left(B \mid \Delta(t_0)\right) - K}{d} && (27) \\
&\geq 1 - \frac{d}{T}\left(\frac{1}{T}\sum_{t=1}^{T} r_\star(t) - \frac{K}{d}\right); && (28)
\end{aligned}$$

where (26) holds because B_{double} is a maximizer of $r(B \mid B)$, (27) holds due to Lemma 2, and (28) is a direct consequence of the definition of $r_\star$. Using similar arguments, we also have

$$\frac{1}{T}\sum_{t=1}^{T} r_\star(t) - \frac{r(B \mid B)}{d+K} \leq \frac{d}{d+K}\,\frac{r(B \mid B)}{d} - \frac{d}{d+K}\,\frac{r\left(B \mid \Delta(t_0)\right) - K}{d} \leq \frac{d}{d+K}\left(\frac{1}{T}\sum_{t=1}^{T} r_\star(t) - \frac{K}{d}\right);$$

and

$$\frac{1}{T}\sum_{t=1}^{T} e_\star(t) - \frac{e(B^\star)}{d} \leq \frac{e(B)}{d} - \frac{r\left(B \mid \Delta(t_0)\right) - K}{d} \leq \frac{1}{T}\sum_{t=1}^{T} r_\star(t) - \frac{K}{d}.$$

■

Appendix B. GENERAL STUDY WHEN μ_a IS NOT CONSTANT ON $\mathcal{Z}$

In this section, we detail the results in the general case where the μ_a are not constant on $\mathcal{Z}$, that were omitted in Section 3 for simplicity. If the definitions of B_{double} and B remain unchanged, we need to adapt the definition of $B^\star$. Indeed, the goal of $B^\star$ is to use first pulls as a calibration step, such that subsequent pulls are in a controlled state. However, assume that the system is in a state with unknown μ_0, and that SI-CombUCB1 returns a block starting with several pulls of a. None of the rewards obtained by the sequence can be used to update our estimates, as they are obtained in states μ_a, μ_{a-1}, μ_{a-2}, ... which are all unknown. This problem is avoided with constant μ_a on $\mathcal{Z}$, as the rewards would be obtained in states μ_a, μ_{a-1}, μ_{a-1}, ... such that the first pull indeed plays its calibration role. Therefore, we now define $B^\star$ as follows

$$B^\star = \operatorname*{argmax}_{B \mid |B|=d,\ a_1 \in B \subseteq a_2} e(B). \qquad (29)$$

The fact that the first two actions in $B^\star$ are now different prevents the issues described above. On the other hand, note that B' in the second claim of Lemma 2 also possesses two different first actions, such that the extra constraint in (29) is not harmful in terms of approximation.

B.1 Equivalent of Proposition 4 and Theorem 1

Proposition 5 and Theorem 2 are the analog of Proposition 4 and Theorem 1 respectively.

Proposition 5 Let $(\ _i)_{i=1}^K$ be a LSD bandit with K arms. Let T 0 be the horizon, and d 0 that divides T. Let $r_{\text{double}}(t)$ be the expected rewards obtained at time step t by the policy playing cyclically B_{double}. We have

$$\frac{1}{T}\sum_{t=1}^{T} r_{\text{double}}(t) \quad 1 \quad \frac{d}{T}\left(\frac{1}{T}\sum_{t=1}^{T} r\ (t) \quad \frac{K+2}{d}\right):$$

Let $2 S_K$, and assume that $d+K$ divides T. Let $r\ (t)$ be the expected rewards obtained at time step t by the policy playing cyclically $[B\ ; B\ ]$. We have

$$\frac{1}{T}\sum_{t=1}^{T} r\ (t) \quad \frac{d}{d+K}\left(\frac{1}{T}\sum_{t=1}^{T} r\ (t) \quad \frac{K+2}{d+K}\right):$$

Let $e\ (t)$ be the expected rewards obtained at time step t by the policy playing cyclically $B^{\ }$. We have

$$\frac{1}{T}\sum_{t=1}^{T} e\ (t) \quad \frac{1}{T}\sum_{t=1}^{T} r\ (t) \quad \frac{K+2}{d}:$$

Proof The proof is similar to that of Proposition 4, see Appendix A.6. The only difference is that we cannot use $r(B\ j\ _B) \quad r\ B\ j\ (t_0) \quad K$, as we did during the proof of the first claim of Proposition 4. Instead, we have to involve $(B\ )^0$, as defined in Lemma 2. Then, we have

$$r(B_{\text{double}}\ j\ _{B_{\text{double}}}) \quad r\ (B\ )^0 j\ _{(B\ )^0} \quad r\ B\ j\ (t_0) \quad (K+2);$$

$$r(B\ j\ _B) \quad r\ (B\ )^0 j\ _B \quad r\ B\ j\ (t_0) \quad (K+2);$$

$$e\ B^{\ } \quad e\ (B\ )^0 \quad r\ B\ j\ (t_0) \quad (K+2);$$

which allow to complete the missing parts in the proofs. Note that in the last equation, the first inequality holds true thanks to the fact that $(B\ )^0$ has also two first actions that are different, while the second inequality can be recovered from similar arguments as those used to prove Lemma 2. ■

Similarly to the constant case, we can use a CombUCB1 approach to produce a sequence of blocks with small regret against $B^{\ }$. The only change is the size of the representation: it is now of dimension $2Kd^2$ since we have $2d$ possible states for the arms (from d to d). Combining (Kveton et al., 2015, Theorem 6) and Proposition 5, we obtain Theorem 2.

Theorem 2 Let $(\ _i)_{i=1}^K$ be an LSD bandit with K arms. Let T 0 be the horizon, and choose d 0 that divides T. Then ISI-CombUCB1, run with block size d and exploration parameter = 1:5, has regret bounded by

$$R_T \quad \frac{(K+2)\,T}{d} + 47d\sqrt{2KT\log\frac{T}{d}} + \left(\frac{2}{3}+1\right)2Kd^3:$$

Choosing $d\ /\ T^{1=4}$, we obtain $R_T = \tilde{O}(KT^{3=4})$, where $\tilde{O}$ is neglecting logarithmic factors.

B.2 Integer Linear Program in the General Case

The last step we need to adapt is the Integer Linear Program which forms the optimization problem of ISI-CombUCB1. As explained at the beginning of the section, one difference is that we need to enforce the first two actions of the block to be different for calibration purposes. The second important difference is the size of the representation: we introduce Y^+ and $Y^{\ }$, both of size Kd^2, to encode pulls in positive and negative states respectively. The problem can be described as follows.

Let $F\ 2\ f0;1g^{K\ \ d}$, such that $F[i;t] = 1$ if and only if arm i is played for the first time in the block at time step t. Let $Y^+\ 2\ f0;1g^{K\ \ d\ \ d}$, such that $Y^+[i;j;t\ ] = 1$ if and only if arm i is played at time step t in state j. Let

$Y \in \{0,1\}^{K \times d \times d}$, such that $Y[i,j,t] = 1$ if and only if arm i is played at time step t in state j. In comparison to the previous Integer Linear Program, the objective function, i.e., the sum of the current UCBs for the second actions present in the block, and the conditions (AC), (UFP), and (FPF) remain unchanged. Their formula are recalled for completeness. As for the novelties, we introduce the Change Action (CA) constraint, i.e., the second pull in the block must differ from the first. Time Consistency (TC) is now divided into TC for positive states (TC^+), i.e., an arm can be pulled in state $j+1$ at time step t only if it was pulled at time step $t-j$, and not pulled since, and TC for negative states, (TC^-), i.e., an arm can be pulled in state $j-1$ at time step t only if it has been pulled consecutively for the last j time steps. Note that it is easier to express TC^- using two conditions, depending on whether $j=1$ or $j \geq 2$. Overall, we have (for index limits that make sense)

$$\sum_{i,j,s} Y^+[i,j,s] + Y^-[i,j,s] \, U_t(i,j) \qquad \text{(objective)}$$

$$\forall t \quad \sum_i F[i,t] + \sum_{i,j} Y^+[i,j,t] + \sum_{i,j} Y^-[i,j,t] = 1 \qquad \text{(AC)}$$

$$\forall i \quad \sum_t F[i,t] \leq 1 \qquad \text{(UFP)}$$

$$\forall i,t \quad \sum_j Y^+[i,j,t] + \sum_j Y^-[i,j,t] \leq \sum_{s=1}^{t-1} F[i,s] \qquad \text{(FPF)}$$

$$\sum_i F[i,2] = 1 \qquad \text{(CA)}$$

$$\forall i,j,t \quad Y^+[i,j,t] \leq F[i,t-(j+1)] + \sum_l Y^+[i,l,t-(j+1)] + \sum_{s \neq j} Y^+[i,j-s,t-s] + \sum_l Y^-[i,l,t-(j+1)] + \sum_{l \neq s} Y^-[i,l,t-j] \qquad (TC^+)$$

$$\forall i,t \quad Y^-[i,1,t] \leq \sum_l Y^+[i,l,t-1] + F[i,t-1] \qquad (TC_1)$$

$$\forall i,t,j \geq 2 \quad Y^-[i,j,t] \leq Y^-[i,j-1,t-1] \qquad (TC_2)$$

Similarly to Section 3, we can approximately solve the above Integer Linear Program by a Branch-and-Bound-like approach, that we detail in the next section.

B.3 Details about the Branch-and-Bound Heuristic

In this section, we provide more details about the heuristic we use to approximately solve the Integer Linear Program (ILP). It works as follows. For every $i \in K$, we set the first action of the block (of total size d) to $a^{(i)}$. We then solve a relaxed version of the ILP, optimizing only for actions $a_2, \ldots, a_d$ (recall that a_1 is fixed to $a^{(i)}$), and allowing for continuous values in $[0,1]$, instead of $\{0,1\}$. This can be done efficiently as the relaxed problem is a standard Linear Program. We finally set a_1 to the $a^{(i)}$ which has given the highest reward according to the relaxed ILP. We reiterate, by testing values for the second action, and solving the relaxed version with respect to $a_3, \ldots, a_d \in [0,1]^{d-2}$, and so on. Let LP be the function that takes as input the current UCBs and a fixed block of size $s+1$, and outputs the best continuous solution in $[0,1]^{d-s}$ for actions $a_{s+1}, \ldots, a_d$. Let $reward$ be the function returning the objective value of any sequence (possibly partially continuous). Our heuristic is summarized in Algorithm 1.

Algorithm 1 Approximate ILP Solver

input : Current UCBs $U_t = [U_t(i, j)] \in \mathbb{R}^{K \times d}$ for all $i \in K$ and $d \geq j \geq d$
init : block = []
for $s = 1 : : : d$ **do**
  for $i = 1 : : : K$ **do**
    $\text{block}_{\text{tmp}} = \text{block} + [a_i]$ // test a_i as next discrete action (current block size: s)
    $\text{block}_{\text{cont}} = \text{LP}(U_t ; \text{block}_{\text{tmp}})$ // find the best continuous continuation (of size $d - s$)
    $r^i = \text{reward}(\text{block}_{\text{tmp}} ; \text{block}_{\text{cont}})$ // compute the total relaxed reward of the half-discrete
                                                // half-continuous block (of overall size d)
  $i = \text{argmax}_{i \in K} r^i$
  block = block $+ [a_i]$ // keep the discrete action with highest relaxed reward
return block

Appendix C. ADDITIONAL EXPERIMENTS

In this section, we provide additional experiments to elaborate more on the performance of SI-CombUCB1. In particular, we enrich the comparison made in Section 4 by benchmarking two new algorithms based on calibration sequences (CS). These approaches first calibrate the system by playing a permutation of all the arms, and then play the best block according to the state reached after (see (8)). CS-based approaches are known to be suboptimal, as they calibrate more arms than necessary, 5 (here K) instead of 2 (number of different arms in the optimal block). In addition, since the calibration phase is of size $K = 5 > 3$, it might prevent from seeing the spike of an $a^{(1)}$ at $= 3$. We therefore benchmark two permutations, CS-worst $= [a^{(1)} ; a^{(2)} ; a^{(3)} ; a^{(4)} ; a^{(5)}]$, and CS-best $= [a^{(5)} ; a^{(4)} ; a^{(3)} ; a^{(2)} ; a^{(1)}]$, named depending on whether they allow to see the spike or not. Again, Algorithm 1 is used as an inner subroutine to approximately compute the solution to the optimization problem (8). Results are shown in Figure 3.

One may argue that the example of Section 4 is unfair to OracleGreedy, because the algorithm is tricked by phenomena that occur beyond the block size, see Figure 2b (top). We now present an example where it is not the case. Consider the 2-armed LSD bandit with Bernoulli rewards and

$$_1(\ ) = \begin{cases} 0{:}06 & \text{if } = 1 \\ 0{:}95 & \text{if } \geq 2 \end{cases} \quad \text{and} \quad _2(\ ) = 0{:}05 \quad 8 \ :$$

The reward functions are represented in Figure 4 (left). Here, OracleGreedy constantly pulls arm $a^{(1)}$, for an average reward of $0{:}06$. Instead, SI-CombUCB1 and CombUCB1 alternate between arms $a^{(1)}$ and $a^{(2)}$, for an optimal average reward of $0{:}5$. Note that SI-CombUCB1, as opposed to CombUCB1, requires a calibration pull before playing this optimal block. The latter bringing little reward, it slightly degrades the average reward and explains the small difference in performance seen in the plots. However, besides the calibration, both algorithms converge towards the same block, which is optimal. Results for $= 10$ are reported in Figure 4 (right).

In conclusion, calibration is an operation that might occasionally degrade the performance (in a controlled way), but that on the other hand guarantees to avoid risky decoys, such as the one discussed in Section 4. This behavior is in line with the fact that the regret of SI-CombUCB1 is well understood theoretically, while CombUCB1 is hard to analyze due to the interferences.

The code is written in Python, and uses the CVXOPT package (Andersen et al., 2013) to solve the Linear Programs. It is publicly available at the following GitHub repository: LSDBandits.