Regret Minimisation in Multi-Armed Bandits Using Bounded Arm Memory

Arghya Roy Chaudhuri, Shivaram Kalyanakrishnan
[1]Department of Computer Science and Engineering
Indian Institute of Technology
Bombay, Mumbai 400076, India
{arghya, shivaram}@cse.iitb.ac.in

Abstract

Regret minimisation in stochastic multi-armed bandits is a well-studied problem, for which several *optimal* algorithms have been proposed. Such algorithms depend on (sufficient statistics of) the empirical reward distributions of the arms to decide which arm to pull next. In this paper, we consider the design of algorithms that are constrained to store statistics from only a *bounded* number of arms. For bandits with a finite set of arms, we derive a sub-linear upper bound on the regret that decreases with the "arm memory" size M. For instances with a large, possibly infinite, set of arms, we show a sub-linear bound on the quantile regret.

Our problem formulation generalises that of Liau et al. (2018), who fix $M = O(1)$, and so do not obtain bounds that depend on M. More importantly, our algorithms keep exploration and exploitation tightly coupled, without a dedicated exploration phase as employed by Liau et al. (2018). Although this choice makes our analysis harder, it leads to much-improved practical performance. For bandits with a large number of arms and no known structure on the rewards, our algorithms serve as a viable option. Unlike many other approaches to restrict the memory of bandit algorithms, our algorithms do not need any additional technical assumptions.

1 Introduction

We consider the well-studied problem of regret minimisation in stochastic multi-armed bandits (Robbins 1952; Berry and Fristedt 1985), which has applications in areas as diverse as drug-testing (Armitage 1960; Colton 1963), crowd-sourcing (Tran-Thanh et al. 2014), and on-line advertising (Li et al. 2010). Having to maximise its rewards by sequentially sampling unknown distributions, an agent must strike an optimal balance between *exploring* all the distributions and *exploiting* the most rewarding ones. Several algorithms have been proposed in the literature (Auer, Cesa-Bianchi, and Fischer 2002; Audibert and Bubeck 2009) with their loss (or regret) within a constant factor of optimal (Lai and Robbins 1985; Auer et al. 2003). Invariably, these algorithms maintain statistics of the pulls performed on each arm (such as the number of pulls and the observed mean reward); on every round, they determine the next arm to pull based on this empirical evidence. Hence, even if a constant number of statistics are recorded for each arm, the memory size needed to run such algorithms is *linear* in the number of arms. In this paper, we consider a setup in which algorithms may only store statistics from a fixed, specified, number of arms, $M \geq 2$.

Our setting is motivated by a number of practical applications, and is also of independent theoretical interest. In crowd-sourcing (Tran-Thanh et al. 2014) instead of tasking all the available workers simultaneously, an experimenter is often constrained to recruit a fixed number of workers for a given duration and assign jobs sequentially among them. In on-line advertising (Schwartz, Bradlow, and Fader 2017) an ad agency might have a large number of products to advertise, but the budget or space only to display a small, fixed number of ads for some specified period.

Researchers have long been drawn to the design of learning agents with a bounded memory size—starting with problems in hypothesis testing (Robbins 1956; Isbell 1959; Cover, Freedman, and Hellman 1976). In the context of bandits, Cover (1968) considers the realisation of an algorithm as a finite state machine, which for two-armed Bernoulli instances, asymptotically converges to the optimal average reward. The model we present in this paper does allow the memory size to depend on the horizon—but restricts that the memory be populated based on rewards from a bounded number of arms. To the best of our knowledge, the earliest analysis of such a constraint was undertaken by Herschkorn, Peköz, and Ross (1996). They present a *non-recalling* algorithm, which can store statistics from only a single arm (that is, $M = 1$). For infinite bandit instances with Bernoulli arms, their algorithm maximises the almost sure average reward over an infinite horizon. Berry et al. (1997) improve these results when the sampling distribution over the infinite set of Bernoulli arms is uniform. Peköz (2003) demonstrates peculiarities that arise in this setting if the the reward distributions of the arms are not stochastically-ordered.

With $M = 1$, an algorithm essentially has to begin afresh when it replaces the arm in its memory with a new one. Hence, the design of algorithms in this regime mainly focuses on an optimal schedule for sampling and erasing from memory. The algorithms are also restricted to 0-1 reward

distributions, which are intimately tied to decision making. Expanding the memory size to $M \geq 2$ opens up a completely new dimension, since one or more "good" arms can be kept in memory as a safeguard against bad luck. Indeed Liau et al. (2018) have recently proposed UCBCONSTANTSPACE, an algorithm that achieves sub-linear regret for finite bandit instances when $M = 4$. Unlike previous results, their bounds hold in finite time, and for arbitrary bounded reward distributions. Yet, their result is limited in three respects, all of which we address in this paper.

The algorithm proposed by Liau et al. (2018) follows an "explore-then-commit" paradigm, wherein a pure-exploration phase targeted at identifying a rewarding arm is followed by a phase to exploit the same arm. *The primary technical contribution of Liau et al. (2018) is therefore a best arm identification algorithm that needs to remember the statistics of exactly four arms at any time.* Whereas this approach greatly simplifies the analysis—and delivers sub-linear regret in finite bandits—it is known to be inferior to strategies that fully interleave exploration and exploitation (Agrawal and Goyal 2013; Auer, Cesa-Bianchi, and Fischer 2002). Our algorithm, UCB-M, adopts the latter approach, which, as expected, performs much better in practice. We also formally show a sub-linear regret bound for UCB-M.

Even if they restrict the arm memory size, Liau et al. (2018) require that their algorithm eventually sample *all* the arms in the bandit instance. In other words, they assume that there is sufficient *time* to pull all the arms even if there is not *space* to simultaneously monitor all the arms. In modern applications, the constraint is usually the opposite: one is apt to run out of time before running out of space. With an eye on practical relevance, we devise an algorithm, QUCB-M, which does not have to explore the entire set of arms. Naturally we cannot guarantee sub-linear regret for QUCB-M, as we do for UCB-M. However, we show a sub-linear bound on the quantile regret (Roy Chaudhuri and Kalyanakrishnan 2018)—which applies even when the set of arms is infinite.

We design our algorithms to work with an arbitrary arm memory size $M \geq 2$, given as an input parameter. Consequently we obtain upper bounds on the regret and quantile regret as functions of M. Such bounds are novel; the algorithm and analysis of Liau et al. (2018) assume M is fixed to 4. Hence, in a scenario where the experimenter allows only $M < 4$, UCBCONSTANTSPACE (Liau et al. 2018) will not be a choice. Alternatively, if the experimenter allows $M > 4$, UCBCONSTANTSPACE is again unable to take advantage of the increased arm memory.

In short, restricting arm memory adds an extra layer to the exploration-exploitation dilemma, which our algorithms negotiate effectively. To the best of our knowledge, the structure and analysis of UCB-M are novel, and at any rate distinct from UCBCONSTANTSPACE (Liau et al. 2018). In fact, UCB-M represents a whole spectrum of algorithms parameterised by the arm memory size. We formalise our problem and then present our algorithms.

2 Problem Definition

A bandit instance $\mathcal{B} = (\mathcal{A}, \mathcal{D})$ comprises a set of arms $\mathcal{A}$ and a set of reward distributions $\mathcal{D}$. Each arm $a \in \mathcal{A}$, when pulled, generates a reward drawn i.i.d. from the corresponding reward distribution $D_a \in \mathcal{D}$, which we assume to be supported in $[0, 1]$. The expected reward of arm $a \in \mathcal{A}$ is given by $\mu_a = \mathbb{E}_{r \sim D_a}[r]$. The experimenter has no information regarding $\mathcal{D}$, and can only gather knowledge about it by pulling the arms. The experimenter's interaction with the bandit instance results in a *history* H_t at each round $t \geq 1$: $H_t \stackrel{\text{def}}{=} \{(a_i, r_i)\}_{i=1}^{t}$, where, $r_i \in [0, 1]$ is the reward produced at i-th step by pulling the arm $a_i \in \mathcal{A}$.

Regret Minimisation. Assuming $\mu^* \stackrel{\text{def}}{=} \min\{y \in [0, 1] : \forall a \in \mathcal{A}, \mu(a) \leq y\}$, and a given horizon of pulls T, the (conventional) cumulative regret incurred by an algorithm is defined as

$$\mathbb{R}_T^* \stackrel{\text{def}}{=} T\mu^* - \sum_{t=1}^{T} \mathbb{E}[\mu_{a_t}], \tag{1}$$

where a_t is the arm pulled by the algorithm on round t. The expectation is taken over random rewards and possible randomisation introduced by the algorithm. In recent work, Roy Chaudhuri and Kalyanakrishnan (2018) introduce "quantile regret", an alternative measure of performance that is especially relevant to infinite-armed bandits. Under their definition, a problem instance $\mathcal{I} = (\mathcal{B}, P_{\mathcal{A}})$ consists of a bandit instance $\mathcal{B}$ as well as a sampling distribution $P_{\mathcal{A}}$ for selecting arms from $\mathcal{A}$. Letting $\rho \in [0, 1]$, the $(1-\rho)$-th quantile of $P_{\mathcal{A}}$ is given by $\mu_\rho = \inf\{x \in [0, 1] : \Pr_{a \sim P_{\mathcal{A}}}\{\mu_a \leq x\} \geq 1 - \rho\}$. Then, for a specified horizon T, the quantile regret for parameter ρ, or the "ρ-regret", is given by

$$\mathbb{R}_T(\rho) \stackrel{\text{def}}{=} T\mu_\rho - \sum_{t=1}^{T} \mathbb{E}[\mu_{a_t}]. \tag{2}$$

If $\mathcal{A}$ is infinite, one cannot hope to achieve sub-linear regret unless some side information is available regarding $\mathcal{A}$ and/or $\mathcal{D}$ (for example, that the mean reward is a linear function of features of the arms (Dani, Hayes, and Kakade 2008)). Yet, even without side information, one can achieve sub-linear ρ-regret for $\rho > 0$ (Roy Chaudhuri and Kalyanakrishnan 2018).

Word RAM Model. Constrained to use a bounded memory size, an algorithm can either curtail the precision with which individual arms' statistics are stored and/or store statistics only from a small number of arms. In this paper, we consider a model that allows individual arms' sufficient statistics to be stored *exactly*, but which limits the storage to a total of M such arms. We consider the "word RAM" model (Aho, Hopcroft, and Ullman 1974; Cormen et al. 2009), which considers a word as the unit of space. This model assumes that each of the input values and variables can be stored in $O(1)$ word space. For finite bandit instances ($|\mathcal{A}| < \infty$), assuming the horizon $T \gg |\mathcal{A}|$, we consider a word to consist of $O(\log T)$ bits. Therefore, the $O(M)$ words available to our algorithm result in a total space complexity of $O(M \log(|\mathcal{A}|T))$ bits, where the additive $\log |\mathcal{A}|$ bits are required to uniquely address the arms.

For infinite bandit instances ($|\mathcal{A}| = \infty$), assuming that the experimenter needs to analyse ρ-regret for some $\rho \in [0, 1]$ and a horizon $T \gg 1/\rho$, we take the word size as $O(\log(T/\rho))$ bits, which leads to a total space complexity of $O(M \log(T/\rho))$ bits. In this case an algorithm encounters an arm only if it is sampled by $P_{\mathcal{A}}$. It does not need the memory to address every arm in $\mathcal{A}$. Roughly $O(1/\rho)$ arms need to be sampled to get one from the top ρ-fraction of arms, and so we allow each sampled arm to be indexed using $O(\log(1/\rho))$ bits. We refer to the set of arms stored in memory as the *arm memory*, whose cardinality is at most M. Before pulling a new arm—one that is not currently in the arm memory— the algorithm must first load it in memory, possibly by deleting an existing arm.

Our aim is to devise algorithms that minimise (conventional) regret and quantile regret while using an arm memory of size at most M; below we formalise these two problems.

CR-M. An algorithm $\mathcal{L}$ is said to *solve* CR-M if for all bandit instances $(\mathcal{A}, \mathcal{D})$ and $M \geq 2$: for a sufficiently large horizon T (not necessarily known beforehand), $\mathcal{L}$ achieves $\mathbb{R}^*_T \in o(T)$ using an *arm memory size* at most M.

QR-M. An algorithm $\mathcal{L}$ is said to *solve* QR-M if for all problem instances $(\mathcal{B}, P_{\mathcal{A}})$ and $M \geq 2$: for all $\rho \in (0, 1]$ and sufficiently large horizon T (that depends on ρ, and may not necessarily known beforehand), $\mathcal{L}$ achieves $\mathbb{R}_T(\rho) \in o(T)$ using an *arm memory size* at most M.

In the following section, we outline the building blocks of our algorithms for solving CR-M and QR-M. These algorithms are then presented in detail in sections 4 and 5, respectively.

3 Key Intuitions

To get familiar with our approach, first consider our simpler problem: CR-M for finite bandit instances ($|\mathcal{A}| < \infty$). Assume $M < |\mathcal{A}|$; otherwise one may apply regular algorithms such as UCB1 (Auer, Cesa-Bianchi, and Fischer 2002). Also assume, to simplify our argument, that there is a unique optimal arm $a^* \in \mathcal{A}$.

Minimising a Product of Probabilities. Intuitively, an algorithm that solves CR-M for finite bandit instances must ensure that the probability of pulling a^* is increased by progressively increasing at least one of two probabilities: (1) the probability that a^* is in arm memory, and (2) if a^* is in arm memory, the probability that it will be pulled more often than the other arms in arm memory. For any algorithm that achieves sub-linear regret, we can write, with T denoting horizon: $\mathbb{R}^*_T \in o(T) \implies \lim_{T\to\infty} \mathbb{R}^*_T/T = 0 \implies \lim_{T\to\infty} \sum_{t=1}^{T} \Pr\{a_t = a^*\}/T = 1$. Now, imposing the arm memory constraint, and letting X_t be the arm memory at t-th pull, we notice $\{a_t = a^*\} \implies \{a^* \in X_t\}$. Therefore,

$$\Pr\{a_t = a^*\} = \mathbb{E}_{X_t}\left[\Pr\{a_t = a^* | a^* \in X_t\} \Pr\{a^* \in X_t\}\right]. \quad (3)$$

Given a bandit instance, the UCBCONSTANTSPACE algorithm of Liau et al. (2018) first solves a pure-exploration problem up to a horizon $\bar{T}$ to maximise the quantity $\Pr\{a^* \in X_t\}$ in the R.H.S. of Equation (3). Once the number of pulls crosses $\bar{T}$ (which is determined based on the accrued rewards), the algorithm chooses the arm with the highest empirical reward in X_t and assigns the remaining pulls to that arm. Therefore, for $t > \bar{T}$, it switches to a pure-exploitation mode, now looking to maximise $\Pr\{a_t = a^* | a^* \in X_{\bar{T}}\}$.

We adopt a more balanced exploration strategy, aiming to simultaneously increase $\Pr\{a^* \in X_t\}$ and $\Pr\{a_t = a^* | a^* \in X_t\}$ throughout the sampling process—with no explicit switch at $\bar{T}$. Note that for the sake of sufficient exploration, an algorithm must not stick to the same arm memory for too long. However, while selecting new arms (not in the current arm memory), it must judiciously decide which arms in memory to replace. It is desirable that the empirically-best arms in memory do not get replaced—this would help retain a^* in memory if it is already present. Achieving this property amounts to minimising *simple regret*, which is defined for the t-th round to be $\mu^* - \mathbb{E}[\mu_{a_t}]$ if $a_t \in \mathcal{A}$ denotes the arm pulled by the algorithm (Bubeck, Munos, and Stoltz 2009). To design our algorithms, we take advantage of a result by Bubeck, Munos, and Stoltz (2009) on the relation between (conventional) regret and simple regret.

Minimising Simple Regret. To present our approach, first we recall the definition of a general forecaster (Bubeck, Munos, and Stoltz 2009), depicted in Figure 1. Given a set of arms $\mathcal{A}$ as input, at each step t, possibly depending on history H_{t-1}, the forecaster selects an arm a_t by using a strategy called the *"allocation strategy"*. By pulling a_t it receives a reward r_t. Then it executes a *"recommendation strategy"* that takes H_t as the input and outputs an arm b_t. The forecaster alternates between its *allocation* and *recommendation* strategies until some stopping condition is met.

$t = 1$, $H_0 = \{\emptyset\}$.
While (stopping condition is not met){
1. Execute *allocation strategy* that possibly depending on H_{t-1} selects an arm a_t.
2. Pull the arm a_t, and get a reward r_t. Update $H_t = H_{t-1} \cup \{(a_t, r_t)\}$.
3. Execute *recommendation strategy* that possibly depending on H_t outputs an arm recommendation b_t. Update $t = t + 1$.

}

Figure 1: A general forecaster, as defined by Bubeck, Munos, and Stoltz (2009).

In Figure 1, if $b_t \equiv a_t$, then the cumulative regret (Equation 1, defined using a_t) of the forecaster is identical to the sum of the simple regret over time steps t (defined using b_t). Theorem 1 by Bubeck, Munos, and Stoltz (2009) shows that the simple and the cumulative regret cannot be minimised all the way to optimality simultaneously. However, as a positive result, they present an upper bound on the simple regret for a number of forecasters (Bubeck, Munos, and Stoltz 2009, Table 1), one of which is defined below.

UCB-MPA. A forecaster, which on each round uses UCB1 (Auer, Cesa-Bianchi, and Fischer 2002) as the allocation strategy, and uses a recommendation strategy that outputs the *most played arm* (MPA) thus far, is denoted UCB-MPA.

On their attempt to establish a problem-dependent upper bound on the simple regret incurred by UCB-MPA, Bubeck, Munos, and Stoltz (2009) [Section 4.2, Theorem 3] obtain a problem-independent bound, which, interestingly, becomes valid for horizons that exceed a problem-dependent threshold. The problem-dependent bound they present closely resembles the problem-independent bound, and does not offer much additional insight. In this paper, we quote their *problem-independent* upper bound (Bubeck, Munos, and Stoltz 2009, Theorem 3) in Theorem 3.1 which becomes the cornerstone of our analysis of UCB-M.

Theorem 3.1 (Theorem 3 in Bubeck, Munos, and Stoltz (2009))**.** *Given a K-sized set of arms $\mathcal{A}$ as input, if* UCB-MPA *runs for a horizon of T pulls such that $T \geq K(K+2)$, then for some constant $C > 0$, its simple regret is at most $C\sqrt{K \log T/T}$.*

In summary, although UCB1 was originally designed as a regret minimisation algorithm (Auer, Cesa-Bianchi, and Fischer 2002), it still performs reasonably well as an exploration strategy—in fact ideal for the exploration-exploitation balance we seek for CR-M. We choose UCB-MPA over other forecasters as it is easy to comprehend and leads to a simpler derivation. Next we present an algorithm for solving CR-M on finite instances.

4 Algorithm for Finite-armed Bandit Instances

We present UCB-M (which internally uses UCB-MPA) and establish a problem-independent upper-bound on its cumulative regret. We empirically compare UCB-M and its variations with the algorithm by Liau et al. (2018). In principle, one may replace the underlying call to UCB1 with any other allocation strategy, such as Thompson sampling (Agrawal and Goyal 2012) or MOSS (Audibert and Bubeck 2009), as we do in our experiments.

4.1 Upper-Bounding Conventional Regret

Algorithm 1 describes UCB-M, which solves CR-M on finite bandit instances. We improve upon the contribution of Liau et al. (2018) in three aspects: (1) UCB-M is empirically seen to be more efficient even with $M = 2$ (as opposed to $M = 4$ for theirs) as it avoids pure-exploration-based elimination; (2) it scales with the arm memory size; (3) we present a problem-independent upper bound on its regret.

Given a finite set of arms $\mathcal{A}$ ($|\mathcal{A}| = K < \infty$) and arm memory size M ($2 \leq M < K$), UCB-M approaches the task in phases. It breaks each phase into $h_0 = \lceil (K-1)/(M-1) \rceil$ sub-phases. Inside any phase w, at each sub-phase j, it runs UCB-MPA on an M-sized subset of arms $S^{w,j}$ (the arm memory), assigns the recommended arm to $\hat{a}$, and forwards it to the next sub-phase. On the subsequent sub-phase (which might belong to the next phase), it chooses $M-1$ new arms from $\mathcal{A}$, along with the arm $\hat{a}$ forwarded from the previous sub-phase, and repeat the previous steps. It is to be noted that the horizon spent on each sub-phase of a phase w is the same and is given by b_w. Also, for $w \geq 2$, the total horizon spent in phase w is given by $h_0 b_w = 2h_0 b_{w-1}$. To satisfy the assumption in Theorem 3 of Bubeck, Munos, and Stoltz (2009), in the first phase, for each of the sub-phases, UCB-M chooses a horizon of $b_1 = M(M+2)$ pulls. For $M \geq K$, as the arm memory size is large enough, it effectively removes the memory constraint. Without the constraint, it is preferable to directly run UCB1 (Auer, Cesa-Bianchi, and Fischer 2002) on the whole instance, which will incur a lower regret. We handle this case within UCB-M. Theorem 4.1 upper-bounds the regret incurred by the algorithm.

Algorithm 1: UCB-M: For finite bandit instances.

Input: $\mathcal{A}$: the set of K arms indexed by $[K]$, $M(\geq 2)$: Arm memory size.

if $M \geq K$ **then**
  Run UCB1 on $\mathcal{A}$ until the horizon runs out.
else
  $b_1 = M(M+2)$. `// Initial horizon per sub-phase`
  $\hat{a} = 1$. `// Initial arm recommendation`
  $w = 1$. `// Counts number of phases`
  $h_0 = \lceil (K-1)/(M-1) \rceil$. `// The number of sub-phases in a phase`
  Randomly shuffle the arm indices.
  while *the horizon is not finished* `// start a new phase` **do**
    $l = 0$.
    for *each sub-phase* $j = 1, \cdots, h_0$*; if the horizon is not finished* **do**
      $S^{w,j} = \{\min\{l+1, K\}, \cdots, \min\{l+1+(M-1), K\}\}$.
      $l \stackrel{\text{def}}{=} \max_a \{S^{w,j} \setminus \hat{a}\}$ **if** $\{\{\hat{a} \in S^{w,j}\} \wedge \{\hat{a} > 1 + \max_a \{S^{w,j} \setminus \hat{a}\}\}\}$ **else** $\max_a S^{w,j}$.
      if $\hat{a} \notin S^{w,j}$ **then**
        $S^{w,j} = \{\hat{a}\} \cup S^{w,j} \setminus \{l\}$.
        $l = l - 1$.
      `/* Allocation Strategy */`
      Run UCB1 on $S^{w,j}$ for horizon of b_w pulls or the remaining horizon; whichever is smaller.
      `/* Recommendation Strategy */`
      $\hat{a} \stackrel{\text{def}}{=}$ The *most played arm* in $S^{w,j}$. `// To be forwarded to next sub-phase`
    $w = w + 1$. `// Increment phase count`
    $b_w = 2 \cdot b_{w-1}$. `// Increment horizon per sub-phase`

Theorem 4.1. *Given as input a set of K arms $\mathcal{A}$, with $K \geq 3$, and an arm memory size M, such that $2 \leq M < K$, for a horizon of T pulls, with $T > KM^2(M+2)$,* UCB-M *incurs regret $\mathbb{R}^*_T = O\left(KM + (K^{3/2}/M)\sqrt{T \log(T/KM)}\right)$.*

Proof Summary. We outline our proof strategy here and furnish the full proof in Appendix A (see supplemental material for all our appendices). For a given bandit instance with K arms, an arm memory size M, and horizon T, the total

number of phases is upper bounded by $\lceil \log_2(2T/MK) \rceil$ (Lemma A.2). In each phase, UCB-M ensures that every arm gets stored in the arm memory at least once (Corollary A.3). This fact, along with Theorem 3.1 helps us upper-bound the difference between the means of any optimal arm $a^* \in \mathcal{A}$, and the best arm in the current arm memory (Lemma A.5). We note that the regret incurred in each sub-phase is linearly dependent on that difference (Equation (4)). As the budget spent on each sub-phase progressively doubles in each phase, that difference falls quickly towards zero, resulting a sub-linear growth in regret. On the other hand, in each sub-phase the incurred regret is sub-linear with respect to the optimal arm in the arm memory (Lemma A.7). Subsequently, we take a sum over all the sub-phases across all the phases to prove the theorem (lemmas A.6 and A.8).

4.2 Experimental Evaluation

We empirically compare UCB-M and its variations with the algorithm UCBCONSTANTSPACE (Liau et al. 2018). The use of UCB1 (Auer, Cesa-Bianchi, and Fischer 2002) as a subroutine in Algorithm 1 can be replaced with any other allocation strategy, which in effect will give rise to a different upper bound. We consider MOSS (Audibert and Bubeck 2009) and Thompson Sampling (Agrawal and Goyal 2013) in our experiments, and rename UCB-M to TS-M and MOSS-M, respectively. Everything else in Algorithm 1, including the recommendation strategy, is kept unchanged.

We run our set of experiments on three different kinds of K-armed Bernoulli instances earlier used by Jamieson and Nowak (2014): $\mathcal{B}_L^K$, $\mathcal{B}_{0.3}^K$, and $\mathcal{B}_{0.6}^K$. The means of the arms in $\mathcal{B}_L^K$ are equally spaced between $\mu_1 = 0.99$, and $\mu_K = 0.01$. For $\mathcal{B}_{0.3}^K$ and $\mathcal{B}_{0.6}^K$ the mean of the i-th arm is given by $\mu_i = 0.01 + \mu^* - (\mu^* - 0.01)((i-1)/(K-1))^\alpha$ for $\alpha = 0.3$, and 0.6, respectively.

For $K = 100$, Figure 2(a) compares the cumulative regret incurred by UCB-M, TS-M, and MOSS-M for an arm memory size $M = 2$, with the UCBCONSTANTSPACE algorithm of Liau et al. (2018). A comparison of cumulative regret and the number of pulls to individual arms (on instances with $K = 10$) is presented in Figure 3. It is important to note that despite using a larger arm-memory size $M = 4$, which is twice that of the others, UCBCONSTANTSPACE incurs a significantly higher regret. Due to its pure-exploration phase, UCBCONSTANTSPACE spends a prohibitively large number of pulls on sub-optimal arms, leading to a high regret. By contrast, our algorithms just make sure that at any instant, the expected difference between the mean of the optimal arm and the best arm in the current arm memory is not too large. Apparently, this difference increases with the subsequent sub-phases. However, UCB-M chooses the optimal arm in its arm memory at least once in any given phase, leading to a "reset" of the difference. On the other hand, this difference progressively reduces across phases due to doubling the horizon in each phase, explaining why UCB-M, TS-M, and MOSS-M incur significantly lower regret. In Appendix B we put additional results from these experiments.

As UCB-M can take advantage of a larger arm memory size, we compare the incurred regret as a function of

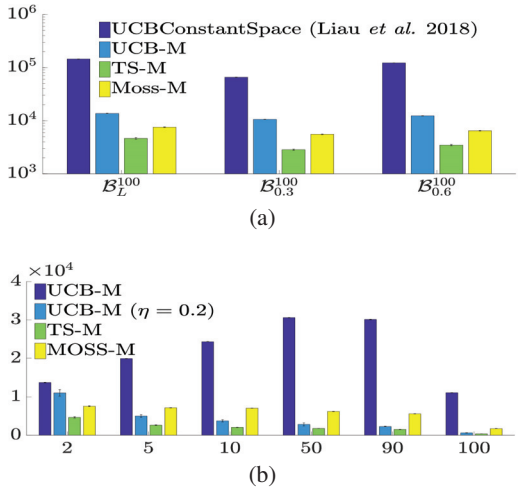
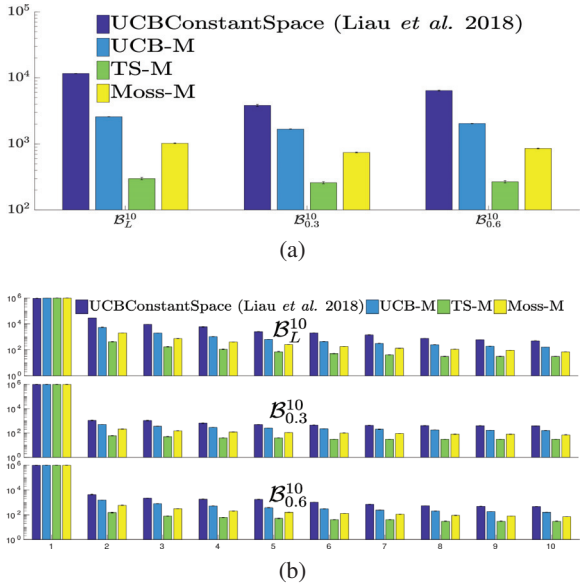


Figure 2: 2(a) presents a comparison of incurred cumulative regret in log scale (y axis), by UCB-M, TS-M, MOSS-M, for $M = 2$, and the UCBCONSTANTSPACE algorithm of Liau et al. (2018) (uses $M = 4$) after 10^6 pulls. Each bar represents the average over 100 runs, with one standard error. For details about the instances, see Section 4.2. 2(b) presents a comparison of incurred regret (y axis) on the instance $\mathcal{B}_L^{100}$ by different algorithms by varying arm memory size M (x axis), after 10^6 pulls. Each bar represents incurred regret averaged over 100 runs, with one standard error. For details about the instances, see Section 4.2.

M. Recall that under UCB1 (Auer, Cesa-Bianchi, and Fischer 2002), if an arm a has been pulled u_a^t times up to round t, and if $\hat{\mu}_a^t$ is its empirical average reward, then the upper confidence bound of that arm is given by $ucb_a^t = \hat{\mu}_a^t + \eta\sqrt{2\ln t/u_a^t}$, with $\eta = 1$. It can be experimentally validated that tuning η can lead to achieving a smaller regret as claimed by the authors (Auer, Cesa-Bianchi, and Fischer 2002, Section 4). We present the regret incurred by UCB-M for $\eta = 0.2$, alongside the other algorithms.

Intuition suggests that increasing M should help in achieving a low regret, as it increases the chance of pulling the optimal arm more frequently. Also, the upper bound given by Theorem 4.1 supports this intuition. However, in practice, we notice an interesting behaviour. On the instance B_L^{100}, we compare the cumulative regret incurred by UCB-M, TS-M, and MOSS-M by varying M. Figure 2(b) shows the results; for a comparison on other instances, the reader is referred to Figure 4 in Appendix B. As expected, UCB-M, TS-M and MOSS-M always incur a higher regret than their unconstrained ($M = K$) counterparts. Also, for UCB-M with $\eta = 0.2$, TS-M and MOSS-M, increasing the arm memory size M reduces the regret. However, the behaviour of UCB-M with $\eta = 1$ is different. If $M < K$, it incurs a relatively low regret for $M = 2$, afterwards *increasing* with M, followed by a slow decrease. On the other hand, UCB-M with $\eta = 0.2$ and the other algorithms not only incur a lower regret but behave consistently. We posit that this peculiar behaviour is due to the looseness of the upper confidence



(a)

(b)

Figure 3: Comparison of incurred cumulative regret (3(a)) and the number of pulls to the individual arms (3(b)) in log scale (y axis), on the instances $\mathcal{B}_L^{10}$, $\mathcal{B}_{0.3}^{10}$ and $\mathcal{B}_{0.6}^{10}$ by algorithms UCB-M, TS-M, MOSS-M (for $M = 2$), and UCBCONSTANTSPACE (Liau et al. 2018), after 10^6 pulls. Each bar represents the average over 100 iterations, and with one standard error. For details about the instances and the algorithms we refer to Section 4.2.

bound in UCB1—which explains why it is absent if $\eta = 0.2$.

5 Algorithm for Infinitely-armed Bandit Instances

Quantile-regret minimisation is conceived by Roy Chaudhuri and Kalyanakrishnan (2018) as a practical approach in bandit instances in which the number of arms is too large for all to be sampled. It would be impossible to achieve sublinear regret on such instances unless some side information regarding the embedding of the arms or the rewards is available. By contrast, sublinear quantile regret can be obtained without such structural information: the simple idea is that if $1/\rho$ arms are sampled at random, in expectation one will lie in the top ρ fraction. Roy Chaudhuri and Kalyanakrishnan (2018) have proven that for any $\rho \in (0,1)$ (unknown to the algorithm), and for a sufficiently large horizon T, their algorithm QRM2 achieves $\mathbb{R}_T(\rho) \in O(((1/\rho)\log(1/\rho))^{2.89} + T^{0.674})$. In this section, we provide a bounded arm-memory algorithm, QUCB-M, that builds on this idea and enjoys sub-linear quantile regret. We empirically compare the algorithm with previous ones from the literature.

5.1 Upper-Bounding Quantile Regret

We solve the QR-M problem by modifying the QRM2 algorithm (Roy Chaudhuri and Kalyanakrishnan 2018) to make it use UCB-M as the sub-routine, and adjust its arm exploration rate accordingly to minimise the upper bound. We call the resulting algorithm QUCB-M and describe it in Algorithm 2. Theorem 5.1 presents its upper bound on its quantile regret.

Algorithm 2: QUCB-M

Input: $\mathcal{A}$, $P_{\mathcal{A}}$, and arm memory size M

Set $\alpha = 0.205$, $B = \left(M^2(M+2)\right)^{\frac{1}{1-\alpha}}$, and $\mathcal{K}_0 = \emptyset$.
for $r = 1, 2, 3, \cdots$ **do**
  $t_r = 2^r B$, $n_r = \lceil t_r^{\alpha} \rceil$.
  Form a set $\mathcal{K}_r$ by selecting additional $n_r - |\mathcal{K}_{r-1}|$ arms from $\mathcal{A}$ using $P_{\mathcal{A}}$, and adding to $\mathcal{K}_{r-1}$.
  Run UCB-M on $\mathcal{K}_r$, for horizon of t_r, with arm memory size M.

Theorem 5.1. *Given as input any problem instance and an arm memory size $M \geq 2$: for $\rho \in (0,1)$ and for a sufficiently large horizon T, the quantile regret incurred by* QUCB-M *is upper-bounded as* $R_T(\rho) \in o\left(\left(\frac{1}{\rho}\log\frac{1}{\rho}\right)^{4.89} + MT^{0.205} + T^{0.81}\sqrt{\frac{\log M}{M^2}\log\frac{T}{M}}\right)$.

The proof applies Theorem 4.1 and follows the steps of the proof of Theorem 3.3 in the paper by Roy Chaudhuri and Kalyanakrishnan (2018). We present the detailed proof in Appendix C. Roy Chaudhuri and Kalyanakrishnan (2018) obtain the upper bound on the quantile regret by simultaneously minimising (1) the regret incurred before discovering a good arm, and once such an arm is discovered, (2) the regret incurred by the subsequent exploration-exploitation routine. The memory constraint on UCB-M means its regret upper bound (Theorem 4.1) grows faster than that of UCB1 as a function of the number of arms K. Since QUCB-M uses UCB-M as a subroutine, naturally, its optimised dependence on T comes out larger than that of QRM2.

It is to be noted that inside QUCB-M one can use the algorithm UCBCONSTANTSPACE by Liau et al. (2018) instead of UCB-M. However, as we have already shown in Section 4.2 that UCB-M is empirically superior to their algorithm, we do not consider this variation in our experiments.

5.2 Experimental Evaluation

Although QUCB-M is designed with the aim of minimising quantile regret, we use conventional cumulative regret as the evaluation metric. Similar to UCB-M, the QUCB-M algorithm can be altered to use TS-M or MOSS-M as the subroutine instead; we call the resulting variants QTS-M and QMOSS-M, respectively. Algorithm 2 uses the value of α that minimises the upper bound in Theorem 5.1. However, for empirical efficiency, we keep $\alpha = 0.347$ as done in QRM2 (Roy Chaudhuri and Kalyanakrishnan 2018).

We compare the regret of each of these algorithms with that of the algorithms by Herschkorn, Peköz, and Ross (1996) and Berry et al. (1997), and present the results in Table 1. We use the same four Bernoulli instances used by Roy Chaudhuri and Kalyanakrishnan (2018)—instances I_1 and I_2 have $\mu^* = 1$, and the probability distributions on μ induced by $P_{\mathcal{A}}$ are given by $\beta(0.5, 2)$, and $\beta(1, 1)$, respectively. Similarly, instances I_3 and I_4 have $\mu^* = 0.6$, and the probability distributions on μ induced by $P_{\mathcal{A}}$ are given by scaled $\beta(0.5, 2)$,

Table 1: Comparison of cumulative regret ($/10^5$) of QUCB-M, QTS-M, QMOSS-M (for $\alpha = 0.347$) with Non-Stationary algorithm (NS) proposed by Herschkorn, Peköz, and Ross (1996), and $\sqrt{T}$-run (STR), $\sqrt{T}\ln T$-learning (STLT), and Non-recalling $\sqrt{T}$-run (NST) algorithms of Berry et al. (1997) after 10^6 pulls, on instances I_1, I_2, I_3 and I_4. Each result is the average of 20 runs, showing one standard error.

Algorithms	M	I_1: $\beta(0.5,2)$ $\mu^*=1$	I_2: $\beta(1,1)$ $\mu^*=1$	I_3: $\beta(0.5,2)$ $\mu^*=0.6$	I_4: $\beta(1,1)$ $\mu^*=0.6$
NS (1996)	1	3.58 ±0.4	1.11 ±0.2	1.64 ± 0.2	0.79 ± 0.1
STR (1997)	2	6.18±0.5	1.11±0.4	4.18±0.3	2.03±0.3
STLT (1997)	2	6.32±0.4	0.69±0.3	4.38±0.2	2.15±0.3
NST (1997)	1	5.35 ±0.5	**0.03** ±0.004	4.56 ± 0.001	2.55 ± 0.001
QUCB-M	2	1.84±0.17	0.41±0.02	1.29±0.10	0.49±0.02
	10	1.98±0.16	0.59±0.02	1.49±0.09	0.63±0.01
QUCB-M ($\eta = 0.2$)	2	2.00±0.20	0.32±0.05	1.41±0.10	0.69±0.04
	10	1.71±0.16	0.16±0.02	1.16±0.09	0.30±0.02
QTS-M	2	1.77±0.17	0.32±0.04	1.23±0.09	0.40±0.02
	10	1.91±0.16	0.18±0.03	1.14±0.10	0.30±0.02
QMOSS-M	2	1.74±0.17	0.31±0.02	1.20±0.10	0.39±0.02
	10	**1.69**±0.15	0.25±0.02	**1.13**±0.09	**0.30**±0.010

and $\beta(1,1)$, respectively. The rows of the table are labeled by the corresponding probability density function of encountering the mean rewards. As Table 1 shows, the existing algorithms incur a significantly higher regret in most of the cases. We put the comparison for $\alpha = 0.205$ in Table 2 in Appendix D.

It is interesting to note that like the finite instances, increasing arm memory leads to a lower regret. Specifically, the scaled version of QUCB-M (using UCB-M with $\eta = 0.2$), along with QTS-M and QMOSS-M, show an improvement with larger arm memory. However, with $\eta = 1$ in the underlying UCB-M, QUCB-M fails to take the advantage of the larger arm memory.

6 Conclusion

We address the problem of regret minimisation in stochastic bandits using a *bounded* arm memory. This problem is relevant in applications where the number of arms is too large to store in memory, and there is no recourse to generalisation over the arms to reduce the number of parameters to learn. Most previous approaches have relied on additional assumptions such as a particular family of distributions for the mean reward, and the rewards being 0-1 (Herschkorn, Peköz, and Ross 1996; Berry et al. 1997). Liau et al. (2018) present an "explore-then-commit" algorithm that escapes such assumptions and still enjoys sub-linear regret. Yet, its "pure exploration" subroutine makes it inefficient in practice. The algorithm depends on eventually sampling every arm in the bandit instance, and so cannot be used on large or infinite instances—which are, in fact, the more relevant targets for the use of a bounded memory.

We propose UCB-M, an algorithm based on UCB1 (Auer, Cesa-Bianchi, and Fischer 2002) for finite bandit instances, which enjoys a sub-linear upper bound on regret, and is significantly more efficient in practice. Unlike previous algorithms, UCB-M offers the flexibility of using an arbitrary arm memory size, facilitating the experimenter to use as much memory as is available. In fact the unique design of UCB-M gives us the freedom of varying the arm memory size M: a clear contrast with UCBCONSTANTSPACE (Liau et al. 2018) which is designed specifically for $M = 4$. Experiments validate our choice of using UCB1 in multiple phases instead of using an explore-then-commit approach: Figure 2(a) shows that UCB-M with a memory size of 2, exactly half that of UCBCONSTANTSPACE, already incurs *less than a tenth* of the regret on the instances and horizons tested. Additionally UCB-M can flexibly use other regret-minimisation routines other than UCB1 (Auer, Cesa-Bianchi, and Fischer 2002) (such as Thompson Sampling (Agrawal and Goyal 2012) and MOSS (Audibert and Bubeck 2009)) while achieving a smaller regret, and retaining its theoretical guarantee. To the best of our knowledge the idea and the analysis of UCB-M are novel and do not resemble that of any existing algorithm.

We extend the algorithm UCB-M to QUCB-M, which uses QRM2 (Roy Chaudhuri and Kalyanakrishnan 2018) as a subroutine, and achieves sub-linear quantile regret under the bounded arm memory constraint. We empirically verify that QUCB-M incurs a lower conventional cumulative regret than existing constrained-memory algorithms (Herschkorn, Peköz, and Ross 1996; Berry et al. 1997) on a wide range of infinite bandit instances.

It would be interesting from a theoretical standpoint to take even the simpler among our problems—CR-M in *finite* bandits—and derive a non-trivial *lower* bound on the regret (note that $\Omega(\sqrt{KT})$ carries over from the unconstrained setting (Auer et al. 2003)). The main technical challenge in so doing is to handle the arm memory—a random variable that limits algorithm's choices, while also being manipulated by the algorithm. To the best of our knowledge, related lower bounds have only been shown in special settings. For instance, Berry et al. (1997) show a lower bound on the failure-proportion for infinitely-armed Bernoulli bandits under the uniform sampling distribution, while Peköz (2003) shows that non-recalling algorithms ($M = 1$) *can* have a linear lower bound unless the reward distributions of the arms are stochastically ordered. Liau et al. (2018) provide a conjecture on the lower bound for their problem, but a formal proof does not appear forthcoming. There is another line of work in the memory-restricted bandit setting by Lu and Lu (2011), who produce both upper and lower bounds on the regret. They restrict the number of rewards that can be remembered for each arm, rather than—as we restrict—the number of arms for which statistics can be maintained. Due to this fundamental difference, their techniques and bounds do not apply in our setting. Hence, establishing a lower bound for CR-M is an interesting challenge, which we leave for future work—while noting that such a bound would also be a prerequisite for a lower bound for QR-M.

7 Acknowledgement

SK was partially supported by grants from SERB (ECR/2017/002479) and Ubisoft India.

References

Agrawal, S., and Goyal, N. 2012. Analysis of Thompson sampling for the multi-armed bandit problem. In *Proc. of the 25th Annual Conf. on Learning Theory*, volume 23, 39.1–39.26. Edinburgh, Scotland: PMLR.

Agrawal, S., and Goyal, N. 2013. Further optimal regret bounds for thompson sampling. In *Proc. AISTATS 2013*, volume 31, 99–107. PMLR.

Aho, A. V.; Hopcroft, J. E.; and Ullman, J. D. 1974. *The Design and Analysis of Computer Algorithms*. Addison-Wesley.

Armitage, P. 1960. *Sequential Medical Trials.* Blackwell Scientific Publications.

Audibert, J.-Y., and Bubeck, S. 2009. Minimax policies for adversarial and stochastic bandits. In *Proc. COLT 2009*, 217–226.

Auer, P.; Cesa-Bianchi, N.; Freund, Y.; and Schapire, R. E. 2003. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.* 32(1):48–77.

Auer, P.; Cesa-Bianchi, N.; and Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47(2-3):235–256.

Berry, D., and Fristedt, B. 1985. *Bandit Problems: Sequential Allocation of Experiments*. Chapman & Hall.

Berry, D. A.; Chen, R. W.; Zame, A.; Heath, D. C.; and Shepp, L. A. 1997. Bandit problems with infinitely many arms. *The Annals of Statistics* 25(5):2103–2116.

Bubeck, S.; Munos, R.; and Stoltz, G. 2009. Pure exploration in multi-armed bandits problems. In *Algorithmic Learning Theory*, 23–37. Springer Berlin Heidelberg.

Colton, T. 1963. A model for selecting one of two medical treatments. *Journal of the American Statistical Association* 58(302):388–400.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2009. *Introduction to Algorithms, Third Edition*. The MIT Press.

Cover, T. M.; Freedman, M. A.; and Hellman, M. E. 1976. Optimal finite memory learning algorithms for the finite sample problem. *Information and Control* 30(1):49 – 85.

Cover, T. M. 1968. A note on the two-armed bandit problem with finite memory. *Information and Control* 12(5):371 – 377.

Dani, V.; Hayes, T. P.; and Kakade, S. M. 2008. Stochastic linear optimization under bandit feedback. In *Proc. COLT 2008*, 355–366. Omnipress.

Herschkorn, S. J.; Peköz, E.; and Ross, S. M. 1996. Policies without memory for the infinite-armed Bernoulli bandit under the average-reward criterion. *Prob. in the Engg. and Info. Sc.* 10(1):21–28.

Isbell, J. R. 1959. On a problem of robbins. *Ann. Math. Stat.* 30(2):606–610.

Jamieson, K. G., and Nowak, R. D. 2014. Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting. *2014 48th Annual Conf. on Information Sciences and Systems (CISS)* 1–6.

Lai, T., and Robbins, H. 1985. Asymptotically efficient adaptive allocation rules. *Adv. in Applied Mathematics* 6(1):4 – 22.

Li, L.; Chu, W.; Langford, J.; and Schapire, R. E. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proc. WWW*, 661–670. ACM.

Liau, D.; Price, E.; Song, Z.; and Yang, G. 2018. Stochastic multi-armed bandits in constant space. In *Proc. AISTATS 2018*, volume 84, 386–394. PMLR.

Lu, C.-J., and Lu, W.-F. 2011. Making online decisions with bounded memory. In *ALT 2011*, 249–261. Springer Berlin Heidelberg.

Peköz, E. A. 2003. Some memoryless bandit policies. *Journal of Applied Probability* 40(1):250–256.

Robbins, H. 1952. Some aspects of the sequential design of experiments. *Bulletin of the AMS* 58(5):527–535.

Robbins, H. 1956. A sequential decision problem with a finite memory. *PNAS* 42(12):920–923.

Roy Chaudhuri, A., and Kalyanakrishnan, S. 2018. Quantile-regret minimisation in infinitely many-armed bandits. In *Proc. UAI 2018*, 425–434. AUAI Press.

Schwartz, E. M.; Bradlow, E. T.; and Fader, P. S. 2017. Customer acquisition via display advertising using multi-armed bandit experiments. *Marketing Science* 36(4):500–522.

Tran-Thanh, L.; Stein, S.; Rogers, A.; and Jennings, N. R. 2014. Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. *Artif. Intl.* 214:89 – 111.