Thompson Sampling for Bayesian Bandits with Resets

Paolo Viappiani

▶ **To cite this version:**

Paolo Viappiani. Thompson Sampling for Bayesian Bandits with Resets. The 3rd International Conference on Algorithmic Decision Theory, ADT 2013, Nov 2013, Bruxelles, Belgium. pp.399-410, 10.1007/978-3-642-41575-3_31 . hal-01215254

HAL Id: hal-01215254
https://hal.science/hal-01215254v1

Submitted on 16 Jan 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thompson Sampling for Bayesian Bandits with Resets

Paolo Viappiani

CNRS-LIP6 and Univ. Pierre et Marie Curie, France
paolo.viappiani@lip6.fr

Abstract. Multi-armed bandit problems are challenging sequential decision problems that have been widely studied as they constitute a mathematical framework that abstracts many different decision problems in fields such as machine learning, logistics, industrial optimization, management of clinical trials, etc. In this paper we address a non stationary environment with expected rewards that are dynamically evolving, considering a particular type of drift, that we call *resets*, in which the arm qualities are re-initialized from time to time. We compare different arm selection strategies with simulations, focusing on a Bayesian method based on Thompson sampling (a simple, yet effective, technique for trading off between exploration and exploitation).

1 Introduction

The multi-armed bandit problem is a general framework that can represent several different sequential decision problems. Essentially, a multi-armed bandit is a a slot machine with n arms (representing possible decisions), each associated with a different and unknown expected payoff (reward). The problem is to select the optimal (or near-optimal) sequence of arms to pull in order to maximize reward (in expectation). Previous rewards obtained from earlier steps are taken into account in order to identify arms that are associated with high payoff; but since reward is uncertain, several pulls of the same arms are usually necessary to assess the quality of an arm with some confidence. A key concept in bandit problems is the trade-off between *exploitation* (pulling the arm with the highest estimated expected payoff) and *exploration* (focusing on getting more information about the expected payoffs of the other arms).

A large number of works have addressed bandit problems [12, 11, 2, 9, 14, 10]. In particular, large attention has been given to methods (including heuristics) that are computationally fast and produce a decision about the next arm to pull in very short time. This include action-value strategies, but also the family of UCB methods [1].

Thompson sampling is a simple and effective strategy for multi armed bandits. It can be implemented very efficiently and it is based on principled Bayesian reasoning. Essentially, this strategy maintain a probabilistic estimate on the value of the arms, and select arms according to their probability of being the optimal arm (it is a randomized selection strategy). This idea was first described eighty years ago [13]. However, it has been surprisingly neglected until it has been recently rediscovered [7, 8, 5, 9], showing its effectiveness in a number of different settings.

Most of the works on bandits assume a stationary distribution of rewards. In many situations, however, one cannot expect the quality of the different arms to be constant. If

the values of the different possibilities change with time (non stationarity) the situation is that of dynamic bandits (also called *restless* in the literature [3]). In this work, in particular, we consider the situation in which drastic changes in the quality of an arm occur (we say that the arm is "reset"). This can model situations of drastic drift, but also situations in which an arm is substituted with a new one (for example in the problem of choosing a seller in electronic commerce, where vendors can suddenly disappear and new ones arrive).

The goal of this work is to show how probability matching with Thompson sampling can be efficiently implemented in non stationary domains (in particular in the case of resets) and to evaluate its performance compared to a number of classic bandit methods.

2 Bayesian Bandits

In bandit problems, one is accumulating rewards from an unknown distribution. There are different possible views on this problem, mainly distribution-free methods (such as the UCB family of strategies [1]) aiming at providing bounds on the worst-case performance, and Bayesian methods (aiming at optimizing average performance). Bayesian bandit strategies maintain an estimation of the goodness of the different arms in term of probability distributions (known as "beliefs") on the value of the arm. A prior (usually uninformative) is given, that is updated using Bayes rules every time an arm is pulled and a reward is observed.

In order to specify a Bayesian approach for bandit problems, we need to address two issues. First, we need to represent distribution information in a practical way. Second, we need to define a strategy that based on the current belief picks the next arm to pull.

More formally, at each round t we have to choose an action (arm) $a \in \mathcal{A}$, where $|\mathcal{A}| = n$, obtaining reward r_t; each arm is associated with a probability density $P(r|a)$ that dictates an average reward $\mu_a = \mathbb{E}_{P(r|a)}[r|a]$; the "best" arm is the one associated with $\mu^* = \max_{a \in \mathcal{A}} \mu_a$. Since the true distribution $P(r|a)$ is not known with certainty, one will often pull suboptimal arms. Let $a[t]$ be the arm pulled at time t. One can state that the goal of a strategy for bandit problems is to maximize long-term (either discounted or undiscounted) cumulative reward $\sum_{t=1,..,T} r_t$, or alternatively, minimize cumulative expected regret, defined as $\sum_{t=1,..,T} \mu^* - \mu_{a[t]}$. Expected regret is the difference between the expected reward associated with the best arm and that of choice made; cumulative expected regret is the sum of this quantity over time.

The Bayesian approach maintains a belief on the possible reward distributions. Assuming a particular type of distribution, we write $P(r|a;\theta)$ to explicitly express the dependency over a set of parameters θ. The belief is then a distribution $P(\theta)$ over the possible instantiations of the parameters θ; that is updated whenever a new pair action-reward (a, r) is observed. The belief $Bel_t(\theta)$ at time t is the probability $P(\theta|(a[1], r_1), .., (a[t-1], r_{t-1}))$ conditioned to the whole history of pulls and rewards. The posterior $Bel(\theta|a[t], r_t)$ becomes the new belief $Bel_{t+1}(\theta)$ (taking the role of "prior") for the timestep $t + 1$.

When considering a Bayesian approach, as we do here, a fundamental issue is that to update the beliefs whenever a new reward is obtained (this essentially mean to apply Bayes theorem). A practical way to do that is to choose distributions of particular forms,

so that they are easy to update. In this paper we focus on Bernoulli bandits, where the reward associated to an arm follows a Bernoulli distribution. Bernoulli bandits can for instance model click behavior and purchase activity in electronic commerce settings [5]. The sequence of rewards/penalties obtained from each arm forms a Bernoulli process with (unknown) probability q_a of "success" (and probability $1 - q_a$ of "failure"). Thus, in the Bernoulli case, possible rewards are in $\{0, 1\}$ and the parameters θ of the model are the elements of the vector $\boldsymbol{q} = \{q_1, ..., q_n\}$ of the success probabilities for each of the n arms; q_a is also the expected reward of the arm ($\mu_a = \mathbb{E}[r|a] = q_a$) and $q^* = \max_a q_a$ is the (true) optimal arm. The q values are not known with certainty, and a belief is maintained. Let $t_a(k)$ be the timestep in which arm a was used for the k-th time; the belief $Bel_t(q_a)$ is the probability $P(q_a \mid r_{t_a(1)}, r_{t_a(2)}, ...)$ conditioned to the rewards obtained when using arm a.

In order to facilitate the operation of Bayesian update, we use the Beta distribution for representing beliefs. The Beta distribution is a conjugate prior for the Binomial distribution; we can therefore implement Bayesian reasoning very efficiently. We maintain two sets of hyper-parameters $(\alpha_1, ..., \alpha_n)$ and $(\beta_1, ..., \beta_n)$. The distribution Beta(α_i, β_i) is the prior belief for arm i. Whenever a success is observed (reward is 1) after pulling arm i, we increment the corresponding α_i; if, on the contrary, we observe a failure (reward is 0) we increment the corresponding β_i. If one assumes an uniform prior, then the initial α_0 and β_0 are set to 1, but a different choice is possible. [1]

Based on the current information about the value of the arms, a strategy needs to select the next arm to pull. Traditional methods, such as action-value strategies and the UCB-1 method, based their selection on the empirical mean alone [2]. Bayesian strategies use the current belief distribution $P(\boldsymbol{q}) = P(q_1, ..., q_n)$ to select the next arm. We now discuss, in the next Section, probability matching with Thompson sampling as a method for arm selection. Then, in Section 4, we adapt this method for environments with resets.

3 Probability Matching with Thompson Sampling

The idea of Thompson sampling is to choose the arm that maximizes the expected reward with respect to a randomly drawn belief. It is a Bayesian method because the current belief (about the $\boldsymbol{q}$ values of the arms) is used directly in order to decide which arm to pull. This technique is also known as "probability matching" and it is based on the intuition that if the number of pulls for a given arm matches its (estimated) probability of being the optimal arm, one can have a good compromise between exploitation and exploration. It is consistent with intuition: if one arm has very low chances of being a good arm (probability of optimality close to zero), it will be (almost) never pulled; similarly if an arm is very likely to be the best, it will be pulled very often. Thompson sampling has been showed to be effective in the context of stationary Bernoulli bandits [7, 5]. Moreover, it has been showed [8] to be effective in the presence of Brownian motion.

[1] α and β are often called "pseudo-counts" for this reason.

[2] UCB-tuned considers the sample variance as well; however in Bernoulli events the variance is a simple function of the mean.

Algorithm 1: Thompson sampling: general case

$Bel_0(\theta) \leftarrow P(\theta)$ (Initialize belief to some initial distribution);
Set $t \leftarrow 0$;
while *true* **do**
  Sample $\hat{\theta} \sim Bel_t(\theta) \;\; \forall i \; 1 \leq i \leq n$;
  Select arm $a \leftarrow \arg\max_i \mathbb{E}[r|\hat{\theta}_i]$;
  Observe reward r_t;
  Bayesian update: $Bel_{t+1}(\theta) \leftarrow Bel_t(\theta|r_t)$;
  $t \leftarrow t + 1$;
end

Algorithm 2: Bayesian Bernoulli bandits with Thompson sampling

Initialize pseudo-counts: $(\alpha_i, \beta_i) \leftarrow (\alpha_0, \beta_0)$;
while *true* **do**
  Sample $\hat{q}_i \sim Beta(\alpha_i, \beta_i) \;\; \forall i \in \{1, ..., n\}$;
  Select arm $a \leftarrow \arg\max \hat{q}_i$;
  Observe reward r_t;
  $\alpha_a \leftarrow \alpha_a + r_t$;
  $\beta_a \leftarrow \beta_a + (1 - r_t)$;
end

Formally, for Bernoulli bandits, assume indicator variable $I_i^{opt}(\boldsymbol{q})$ to yield 1 iff $q_i = q^*$ and 0 otherwise. Probability matching with Thompson sampling is a randomized strategy that consists, at any time t, in pulling arm i with probability equal to $P^{opt}(i)$, being the probability that the arm i is optimal according to the current beliefs. For each arm i, this is

$$P^{opt}(i) = \int I_i^{opt}(\boldsymbol{q}) P(\boldsymbol{q}) d\boldsymbol{q} = \int_0^1 ... \int_0^1 I_i^{opt}(q_1, ..., q_n) \prod_i P(q_i) \; dq_1 ... dq_n. \quad (1)$$

(where we use the fact the q values are probabilistically independent) Since the value P^{opt} might not be easy to compute, in practice, the rule is implemented in the following way (that is what is more strictly referred as *Thompson sampling*). In each round, a set of parameter $\hat{q}$ is sampled from the posterior $P(q|r_1, ..., r_t)$, and the arm with highest value $\hat{q}^*$ is chosen. Conceptually, this means that the agent instantiates the value of the arms randomly according to his beliefs in each round, and then he acts optimally according to this instantiation (the general algorithm is shown in Algorithm 1 and the algorithm specific to Bernoulli bandits in Algorithm 2). An advantage of Thompson sampling is the absence of tuning parameters, in contrast with most (if not all) heuristic methods, where setting the right value for the parameters is crucial.

Thompson sampling is particularly simple to implement in the case of Bernoulli bandits assuming Beta priors. The set of hyper-parameters α_i and β_i are initially initialized according to prior information (setting them all to 1 coincides to an uniform prior). Bayesian update consists in updating the hyper-parameter in the following way:

Algorithm 3: Particle Filter reset-aware Thompson for Bayesian Bernoulli bandits

Data: Prior hyper-parameters (pseudo-counts): α_0, β_0
Initialize set of particles $\mathcal{P}_i = (q_i^1, ..., q_i^L)$ for each i uniformly in $Beta(\alpha_0, \beta_0)$;
Set time $t \leftarrow 0$;
while *true* **do**
  for *each arm i in 1,...,n* **do**
    for *each particle j in 1,...,L* **do**
      while *rand*() $< p_{reset}$ **do**
        resample $q_i^j \sim Beta(\alpha_0, \beta_0)$;
      end
    end
    Sample a particle from each particle set: $\hat{q}_i \sim \mathcal{P}_i \; \forall i$;
  end
  Select arm $a \leftarrow \arg\max_i \hat{q}_i$;
  Observe reward r_t;
  $\mathcal{P}_a \leftarrow$ ImportanceSampling$(\mathcal{P}_a, r_t)$;
  $t \leftarrow t + 1$;
end

when arm i is pulled, α_i is incremented if a positive reward ($r = 1$) is observed, otherwise in the case of no reward ($r = 0$) β_i is incremented. The decision of the arm to pull consists in sampling the values for the q_i according from Beta with the current value hyper-parameters, and selecting the highest one.

4 Reset-aware Thompson Sampling

In this paper, we consider dynamic (restless) bandits, focusing on the particular of rewards that can drastically change from time to time (resets). This can model situations like a new user of a electronic commerce website, or when a supplier changes ownership. In our analysis, we assume that there exists a (fixed) probability p_{reset} under which the payoffs are changed, and reset rewards are re-sampled according to the prior $Beta(\alpha_0, \beta_0)$, with hyper-parameters α_0 and β_0 (prior pseudo-counts).

The principle of Thompson sampling can be extended to the case of presence of resets. The key issue in using Thompson is being able to sample from the posterior. The problem is now that we cannot anymore represent the posterior by simply maintaining a vector of pseudo-counts. We will show however that we can still use Thompson sampling from the posterior distribution, considering two different methods.

4.1 Particle Filter Thompson

This method (Algorithm 3) computes an unbiased estimation of the belief distribution of the quality of each arm using particles. $Bel_t(\boldsymbol{q})$ is approximated by a set of particles: a set of L particles is associated to each arm (each particle is a scalar between 0 and 1,

Algorithm 4: Geometric-Beta reset-aware Thompson sampling for Bayesian Bernoulli bandits

Data: Prior pseudocount: α_0, β_0
Initialize history log;
Set time $t \leftarrow 0$;
while *true* **do**
  for *each arm i in 1,...,n* **do**
    $\bar{t}_i \leftarrow lastUse(i, t)$;
    while *rand*() $< 1 - p_{reset}$ **do**
      $\bar{t}_i \leftarrow lastUse(i, \bar{t}_i)$;
    end
    $\alpha_i \leftarrow \alpha_0 +$ number of sucessful pulls of arm i between time $\bar{t}_i$ and t;
    $\beta_i \leftarrow \beta_0 +$ number of unsucessful pulls of arm i between time $\bar{t}_i$ and t;
    Sample $\hat{q}_i \sim Beta(\alpha_i, \beta_i)$;
  end
  Select arm $a \leftarrow \arg\max \hat{q}_i$;
  Observe reward r_t;
  Log results;
  $t \leftarrow t + 1$;
end

representing a particular hypothesis about the value q_i). The approximation is unbiased, meaning that for a large number of particles, the mean of the distribution converges to the true mean. The belief is propagated during time using a particle filer, composed of two parts: a *transition model* (that simulates the dynamics; i.e. the possibility that an arm can be reset) and an *observation filtering*, where using importance sampling, a new set of particles is selected from the old ones, favoring the particles that better explain the observed reward.

The transition model (the innermost loop in Algorithm 3) iterates over the particles of each arm and substitutes a particle with a new one (sampled from the prior) with probability p_{reset}. In the observation filtering model, particles are weighted by the likelihood of the observed reward given their hypothesis: q_i if $r_t = 1$ and $1 - q_i$ otherwise (if $r_t = 0$). The weights are used to resample a new set of L particles; the new particles represent an approximation of the posterior distribution. At any given time-step, Thompson sampling is realized by sampling exactly one particle for each arm uniformly at random from the associated particles, and pulling the arm whose sampled particles has the highest value.

4.2 Geometric-Beta Thompson

The key insight of this method is what it matters is the last time that the arm was reset. The belief distribution of the value of the arm *assuming the arm was reset at time* $\hat{t}$ is a Beta distribution with hyper-parameters dictated by the number of successes and failures since $\hat{t}$. Let x_t^i be 1 if the arm i is pulled at time t (0 otherwise), and r_t be the reward received at time t. Following our intuition, and assuming Beta priors and a

probability of drift p_{reset}, we approximate the probability distribution (belief) $Bel_t(q)$ of the value of an arm i at time t with the following:

$$\sum_{k=1}^{t} p_{reset}(1-p_{reset})^k \cdot f_{Beta}\left(q\,;\alpha_0+\sum_{m=k}^{t} r_m\, x_m^i,\ \beta_0+\sum_{m=k}^{t}(1-r_m)\, x_m^i\right)$$

where $f_{Beta}(x;\alpha,\beta)$ is the density of the Beta distribution with parameters α and β.

We now adapt Thompson sampling for the case of resets by sampling, independently, for each arm, the time of the last reset. At each time step, for each arm, we independently sample a time-step $\bar{t_i}$ from a geometric distribution, parameterized by p_{reset} the reset probability, in order to decide "how far in the past we should go" to set the temporal "window" that it will be used to compute the hyper-parameters α_i and β_i. Let $lastUse(i,t)$ be a function that returns the last time-step t' before t in which arm i was pulled. We repeat the following procedure: we sample a uniform random number between 0 and 1 and, while it is lower than p_{reset}, we set $\hat{t}_i \leftarrow lastUse(i,\hat{t}_i)$ and repeat the loop. The sampled time-step will be used to derive the pseudo-counts that will be used for Thompson sampling.

Considering the history of previous pulls, we consider the interval from $\hat{t_i}$ to the current t, and count the number of successful and unsuccessful pulls (for arm i). The belief distribution of the value of the arm i *assuming the arm was reset at time* $\hat{t}_i$ can be modeled as a Beta distribution with hyper-parameters dictated by the number of success and failures since $\hat{t}$. Using our notation, the meta-parameters are $\alpha_i = \alpha_0 + \sum_{t'=\bar{t}_i}^{t} r_{t'} x_{t'}^i$ and $\beta_i = \beta_0 + \sum_{t'=\bar{t}_i}^{t}(1-r_{t'})x_{t'}^i$. The complete algorithm is shown in Algorithm 4; the computation of pseudo-counts can be made more efficient by maintaining a cumulative sum at each time step. We note that this is a (biased) approximated method: in general the belief (posterior probability) of an arm being reset at a time-step is not independent from the belief for the arm value (for example, if an arm has given reward 1 for 10 times and then we observe reward 0 for several times since then, our estimation for a reset at timestep 11 becomes much higher).

5 Experiments

In order to evaluate the effectiveness of different methods, we need to define some evaluation metrics. In principle, one would like to be able to accumulate as much reward as possible. Cumulative reward is therefore a natural criterion. It is also interesting to compare the reward obtained following a policy with that of always pulling the "best" arm (that of course is not known with certainty by the bandit strategy). Expected regret for a single pull of a Bernoulli bandit is $R = q^* - q_a$, and cumulative expected regret is $\sum_{t=1,..,T} q^* - q_{a[t]}$ for a particular run of the algorithm. We remark that all algorithms make choices (on which arm to pull) based on the history of rewards obtained, often including some explicit randomization (that is the case of epsilon-greedy methods, and also Thompson sampling). In theory, each strategy could be measured according to its *expected expected regret* (in expectation over possible history of pulls and rewards obtained) and *expected expected reward*, but these measures are extremely

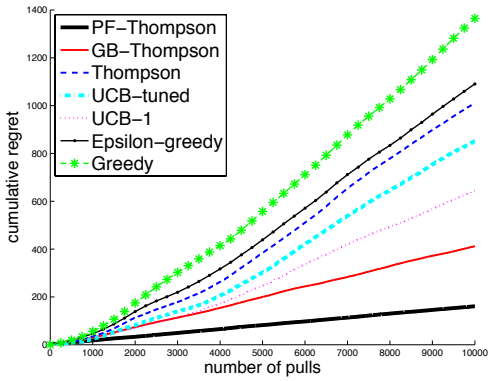

Fig. 1. Regret; $p_{reset} = 0.001$, $n = 2$.

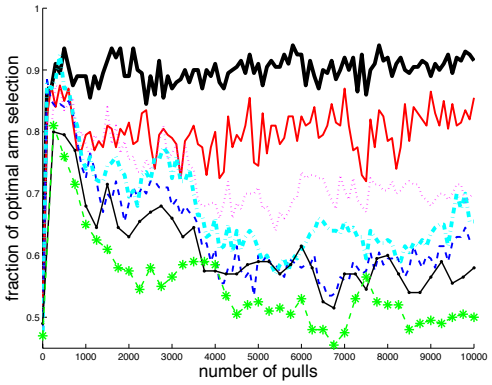

Fig. 2. Fraction of optimal arm selection; $p_{reset} = 0.001$, $n = 2$.

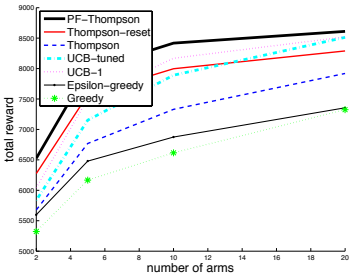

Fig. 3. Reward vs number of arms; $p_{reset} = 0.001$; $(\alpha, \beta) = (1, 1)$.

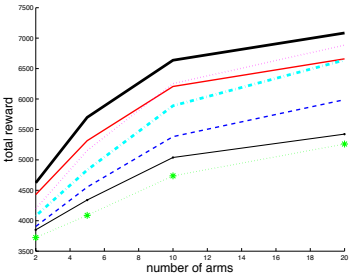

Fig. 4. Reward vs number of arms; $p_{reset} = 0.001$; $(\alpha, \beta) = (2, 1)$.

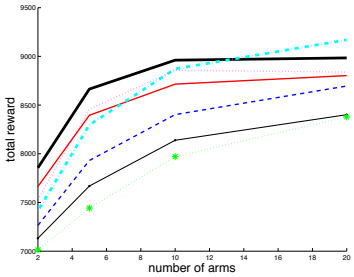

Fig. 5. Reward vs number of arms; $p_{reset} = 0.001$; $(\alpha, \beta) = (1, 2)$.

hard to calculate analytically. Therefore, with simulations, we compare the different strategies according to their average performance, in particular the values obtained for cumulative expected reward averaged over a large number of runs (this choice allows to directly compare the degradation of total reward when p_{reset} increases).

In the following experiments we evaluate Thompson sampling and some classic bandit strategies in presence of resets, in a variety of settings. We are interested to verify whether probability matching, implemented with Thompson sampling as presented in this paper, is an effective strategy for balancing exploration and exploration. At any time-step, let $\bar{\mu}_i$ be the empirical mean and $\bar{\sigma}_i$ the sample variance of the rewards observed when pulling arm i. We compare the following strategies for choosing the next arm to pull:

Thompson sampling with Particle-filter: (indicated as *PF-Thompson* in the plots below) our strategy, described in Algorithm 3, using a particle filter to estimate the belief distribution (we use 10000 particles in our simulations).

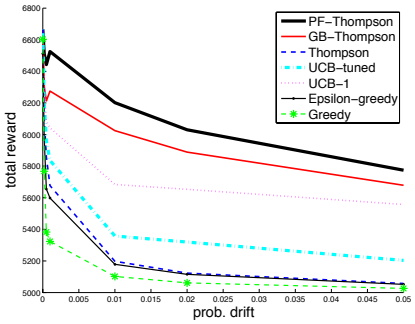

Fig. 6. Reward vs p_{reset}; $n = 2$, $(\alpha, \beta) = (1, 1)$.

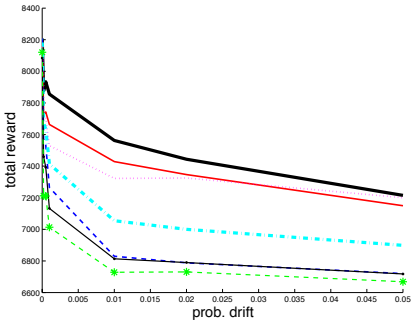

Fig. 7. Reward vs p_{reset}; $n = 2$, $(\alpha, \beta) = (2, 1)$.

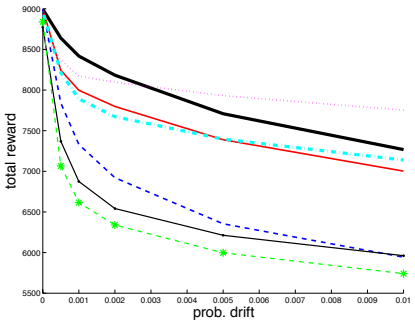

Fig. 8. Reward vs p_{reset}; $n = 10$, $(\alpha, \beta) = (1, 1)$.

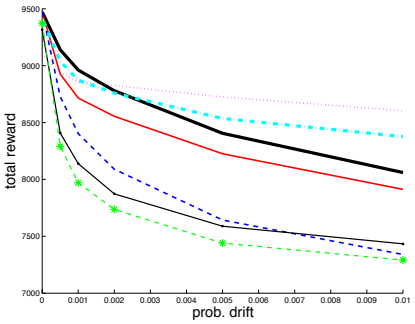

Fig. 9. Reward vs p_{reset}; $n = 10$,$(\alpha, \beta) = (2, 1)$.

Geometric-Beta Thompson sampling: (indicated as *Thompson-reset* in the plots) our probability matching reset-aware strategy using Thompson sampling in two steps, sampling first from a Geometric distribution and then from Beta (see Algorithm 4).

(Standard) Thompson sampling: probability matching with Thompson sampling. A set of hyper parameters α and β (a pair for each arm) are maintained as explained above. A value $q_i \sim Beta(\alpha_i, \beta_i)$ is sampled for each arm and the arm $a = \arg\max_i q_i$ with maximal value is pulled (see Algorithm 2).

UCB-tuned: The strategy is a modification of UCB-1 that is claimed to be more effective in practice; the index associated to each arm is the following:

$$\tau_i = \bar{\mu}_i + \sqrt{\frac{ln(t)}{m_i} \min\left(0.25,\ \bar{\sigma}^2 + \sqrt{\frac{2ln(t)}{m_i}}\right)}$$

where m_i is the number of times arm i has been pulled.

UCB-1: Each arm i is associated with an index $\tau_i = \bar{\mu}_i + \sqrt{\frac{\log 2t}{m_i}}$; the arm $i^* = \arg\max \tau_i$ with highest index is picked.

Greedy strategy: This basic strategy always selects the currently best performing arm according to the empirical mean: $i^* = \arg\max \bar{\mu}$. Greedy always exploits, therefore it will often lead to suboptimal choices. In order to force some exploitation, especially at the beginning, the empirical mean of each arm is biased by adding 1 to both to the numerator and the denominator.

Epsilon-Greedy: This random strategy selects the currently best performing arm i^* (wrt empirical mean) with probability $1-\epsilon$ and any other arm $j \neq i^*$ is selected with probability $\frac{\epsilon}{n}$ (with $\epsilon = 0.05$ in the simulations below).

The initial rewards are sampled from a Beta prior; we consider the following possible values for the prior hyper-parameters (α_0, β_0) shared by all arms: $(1, 1)$ equivalent to a uniform distribution, $(2, 1)$, and $(1, 2)$. We simulate the behavior of each strategy; at each step of the simulation our Bayesian strategies choose which arm to pull based on the current belief, a reward is sampled and the belief is updated; the other non-Bayesian methods only keep aggregate information (sample mean and variance of each arm). We let simulations proceed for a relatively long time (10000 steps) in order to be able to witness the effect of (possibly multiple) resets. In order to compare the strategies in the fairest possible way (and reducing noise), at each run a complete history of the reward dynamics is generated beforehand, and experienced by all strategies in the same manner. Experimental results are averaged over 300 runs.

First, let's consider a specific setting with 2 arms and $p_{reset} = 0.001$ (this means we can expect approximately around 10 resets per arm during each simulation). We show the cumulative regret obtained by each of the strategy as function of the number of pulls in Figure 1. PF-Thompson clearly dominates all other methods; the approximated strategy Thompson-reset beats the other methods, but it is significantly worse than PF-Thompson. Action-value methods and UCB methods fail to effectively adapt to the reward dynamics. We also note that regret is practically stable for PF-Thompson, but it actually increases for the other strategies (this is due to the drift). We also show the fraction of optimal arm selection (Figure 2): the Thompson strategies are constantly selecting the true highest performing arm with high probability. The fraction of optimal arm selection decreases over time for action value methods, only UCB-1 remains somewhat competitive.

We evaluated the impact of the number of arms on total reward. The results are shown in Figures 3, 4, and 5 for prior hyper-parameters (α_0, β_0) set to $(1, 1)$, $(2, 1)$ and $(1, 2)$, respectively. The reset-aware Thompson strategy with particle filter (PF-Thompson) is dominating in most of the settings. The Geometric-Beta version is also very efficient in many settings (in particular when considering 2 arms); however, when considering 10 arms, it seems that either UCB-1 (in some settings) or UCB-tuned (in other settings) are better.

We also considered the impact of changing p_{reset} on the total cumulative reward (obtained with 10000 pulls) when fixing the number of arms (Figures 6, 7, 8, and 9). Surprisingly, UCB-1 is the best performing strategy when considering 10 arms and a high reset probability. When prior hyper-parameters are optimistic ($\alpha_0 = 2$, $\beta_0 = 1$), also UCB-tuned becomes very effective for $p_{reset} \geq 0.003$ and 10 arms.

Overall, our reset-aware Thompson sampling is an effective technique for bandit problems with presence of resets. In particular, the version based on particle-filtering

is particularly effective, and it is dominating the other strategies in most of the settings we tested. However, it is not the best strategy in all the cases; UCB-1 and UCB-tuned are surprisingly competitive in a small number of settings (but perform very poorly in others).

6 Discussion and Conclusion

Multi-armed bandits are the quintessential problem facing the exploration/exploitation tradeoff. In this paper we addressed the problem of non-stationary multi armed bandits with resets, a form of drift that will typically occur (relatively) rarely but it is associated with drastic changes in the value of the choices. In our specific setting, a reset occur with a fixed i.i.d. probability at each step and, in case of reset, the value of the arm is reassigned to a random value sampled from a prior distribution.

We compared different strategies for multi armed bandits, aimed at achieving a good compromise between exploitation and exploration in presence of resets, evaluating them with respect to cumulative reward and regret. We simulated a number of bandit problems, with different values for the reset probability p_{reset}, the number of arms and the initial priors. In particular we showed how Thompson sampling can be effective in case of resets. Differently from drift-aware techniques based on computing pseudo-counts in fixed temporal windows [8], our is a principled solution that use Thompson sampling form the right posterior.

We stress that, while much interest in the Bandit community is about theoretical bounds, we are instead particularly interested in practical efficacy. We show, with simulations, how our strategy based on Thompson sampling is effective in practical circumstances. Thompson sampling is particularly appealing because of its simplicity and (when using conjugate-prior distributions) its efficient implementation. A practical problem is that generally one cannot assume that the reset probability is known a priori. Moreover, realistic domains will possibly include different type of drifts at the same type; for instance, one could consider dynamic values generated by random walks. We plan to investigate techniques for simultaneously learning the "drift" (the reset probability p_{reset} in our case) while estimating the value of the arms.

Our work is related to the Mortal bandit problem of Chakrabarti et al. [4], where arms may "die" and become unavailable, while at the same time new arms may appear. In our model, an arm that is reset could be viewed as dead arm substituted with a new one, but our problem is more challenging as we do not observe which arm dies. Gittins indices [6] are a solution to the bandit problems, and in principle can be used for dynamic settings as well. However they are computationally intensive to compute.

Other directions for future works include bandits with continuous rewards, further experimental evaluation, theoretical analysis of the worst-case.

References

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

2. S. Bhulai and G. Koole. On the value of learning for Bernoulli bandits with unknown parameters. *Automatic Control, IEEE Transactions on*, 45(11):2135–2140, nov 2000.
3. Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.
4. Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal Multi-Armed Bandits. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems (NIPS 2008)*, pages 273–280, 2008.
5. Olivier Chapelle and Lihong Li. An Empirical Evaluation of Thompson Sampling. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems (NIPS 2011)*, pages 2249–2257, 2011.
6. John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed Bandit Allocation Indices*. Wiley, 2 edition, March 2011.
7. Ole-Christoffer Granmo. Solving Two-Armed Bernoulli Bandit Problems Using a Bayesian Learning Automaton. *International Journal of Intelligent Computing and Cybernetics (IJICC)*, 3:207–234, 2010.
8. Neha Gupta, Ole-Christoffer Granmo, and Ashok Agrawala. Thompson Sampling for Dynamic Multi-armed Bandits. *Machine Learning and Applications, Fourth International Conference on*, 1:484–489, 2011.
9. Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson Sampling: An Asymptotically Optimal Finite-Time Analysis. In *Algorithmic Learning Theory - 23rd International Conference (ALT 2012)*, pages 199–213, 2012.
10. Chien-Tai Lin and C. J. Shiau. Some optimal strategies for bandit problems with beta prior distributions. *Ann. Inst. Stat. Math.*, 52(2):397–405, 2000.
11. William G. Macready and David Wolpert. Bandit problems and the exploration/exploitation tradeoff. *IEEE Trans. Evolutionary Computation*, 2(1):2–22, 1998.
12. Ilya O. Ryzhov and Warren B. Powell. The value of information in multi-armed bandits with exponentially distributed rewards. *Procedia CS*, 4:1363–1372, 2011.
13. William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):pp. 285–294, 1933.
14. Hamed Valizadegan, Rong Jin, and Shijun Wang. Learning to trade off between exploration and exploitation in multiclass bandit prediction. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2011)*, pages 204–212, 2011.