Improving Particle Thompson Sampling through Regenerative Particles

Zeyu Zhou
Department of Radiology
Mayo Clinic
Rochester, MN, USA
zeyuzhou91@gmail.com

Bruce Hajek
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Champaign, IL, USA
b-hajek@illinois.edu

***Abstract*—This paper proposes regenerative particle Thompson sampling (RPTS) as an improvement of particle Thompson sampling (PTS) for solving general stochastic bandit problems. PTS approximates Thompson sampling by replacing the continuous posterior distribution with a discrete distribution supported at a set of weighted static particles. PTS is flexible but may suffer from poor performance due to the tendency of the probability mass to concentrate on a small number of particles. RPTS exploits the particle weight dynamics of PTS and uses non-static particles: it deletes a particle if its probability mass gets sufficiently small and regenerates new particles in the vicinity of the surviving particles. Empirical evidence shows uniform improvement across a set of representative bandit problems without increasing the number of particles.**

***Index Terms*—stochastic bandit, Thompson sampling, particles**

I. Introduction

Thompson sampling (TS) is a Bayesian heuristic for solving general stochastic bandit problems, in which the rewards are generated according to a given distribution with a fixed unknown system parameter. TS maintains a posterior distribution on the parameter and selects an action according to the posterior probability that the action is optimal. TS is known for its ability to automatically handle bandit setups with a complex information structure and its strong empirical performance. However, efficient updating, storing, and sampling from the posterior distribution in TS are only feasible for some special cases (e.g. conjugate distributions). For general bandit problems, one has to resort to various approximations, most of which require specific problem structures and are complicated.

Particle Thompson sampling (PTS) is an approximation of TS obtained by replacing the continuous posterior distribution by a discrete distribution supported at a set of weighted static particles. Updating the posterior distribution then becomes updating the particles' weights by Bayes formula, followed by normalization. PTS applies to very general bandit setups and is easy to implement. However, it may seem on the surface that the crude approximation may bring down the performance of TS significantly, because the system parameter may live in a high-dimensional space and the set of particles in PTS is finite and static and may not contain the actual parameter. Intuitively, the performance of PTS can be improved by using more particles. However, that comes with an increasing computational cost.

The main contribution of this paper is the proposal of *regenerative particle Thompson sampling* (RPTS), an improvement of PTS without using more particles. RPTS exploits the particle weight dynamics of PTS observed in [1] that the weights of all but a few fit particles converge to zero. RPTS is based on the following heuristic: replace the decaying particles in PTS with new generated particles in the vicinity of the survivors. Empirical results show that RPTS outperforms PTS uniformly for a set of representative bandit problems. RPTS is very flexible and easy to implement.

The remainder of this paper is organized as follows. Section II reviews some related work and introduces the problem setup. Section III proposes and explains RPTS. Section IV empirically evaluates the performance of RPTS. Section V discusses the limitations of RPTS and concludes the paper.

II. Related Work and Problem Setup

This paper is a follow up work of [1]. Much of this section refers to the corresponding sections of [1] to avoid repetition.

See Section II of [1] for a review of related work on bandit problems, Thompson sampling (TS), particle Thompson sampling (PTS), and some other approximations of Thompson sampling.

The fewness of survivors in PTS is proved in [1] for Bernoulli bandits and speculated for general stochastic bandits. This suggests the potential of improving PTS by eliminating some particles. [1] also shows that the survivor particles tend to have high fitness in terms of KL divergence.

RPTS proposed in this paper resembles particle filter/sequential Monte Carlo [2], as both contain particle updating/selection and exploration/mutation processes. However, they differ much in implementation details. RPTS maintains a weighted set of particles and only selects/eliminates particles when the weights satisfy a condition. Particle filter maintains unweighted particles (after selection) and selects at every time step. The mutation process in particle filter normally relies on a Markovian evolution of the hidden state, which is absent in the version of bandit problems considered here. Fundamentally,

This work was supported in part by NSF Grant Grant CCF 19-00636. It was done while the first author was a Ph.D. student at the Electrical and Computer Engineering Department of UIUC.

RPTS and particle filter have different problem contexts: the former needs to balance between exploration and exploitation, whereas the latter does not consider exploitation.

See Section III of [1] for the definitions and notation setup of stochastic bandit problems, Thompson sampling, and particle Thompson sampling. We add here the definition of *regret*, the performance measure of an algorithm for a stochastic bandit problem. Let $a^* \triangleq \arg\max_{a\in\mathcal{A}} \mathbb{E}_{\theta^*}[R(Y)|a]$ denote the optimal action that maximizes the mean reward, assuming complete knowledge about the true system parameter θ^*. Let $R^* \triangleq \mathbb{E}_{\theta^*}[R(Y)|a^*]$ denote the maximum expected reward. The regret of an algorithm that selects A_t at time t is $\mathrm{reg}_t \triangleq R^* - \mathbb{E}_{\theta^*}[R(Y)|A_t]$, the difference between the expected reward of an optimal action and the action selected by the algorithm. The cumulative regret up to time t is $\sum_{\tau=1}^{t} \mathrm{reg}_\tau$.

III. RPTS: Regenerative Particle Thompson Sampling

PTS often suffers from poor empirical performance, especially when the parameter space Θ has a high dimension and the number of particles is not large enough. Simply increasing the number of particles comes with a higher computational cost. The potential to improve PTS without using more particles lies in the observation that usually not many particles can survive. This phenomenon is proved in [1] for Bernoulli bandits and can be verified empirically for many other bandit problems. When the weights of the decaying particles become so small that they become essentially inactive, continuing using these particles would be a waste of computational resource. Also, the survived particles are usually close to θ^* in a certain sense, where the closeness is measured by KL divergence [1], related to Cartesian distance in many problems.

We propose *Regenerative particle Thompson sampling* (RPTS) based on the following heuristic inspired by biological evolution: *delete decaying particles, regenerate new particles in the vicinity of the fit surviving particles.* See Algorithm 1.

Algorithm 1 Regenerative particle Thompson sampling

Input: $\mathcal{A}, \mathcal{Y}, \Theta \subset \mathbb{R}^K, P_\theta(\cdot|a), R, \theta^*, \mathcal{P}_N$
Parameters: N, $f_{del} \in (0,1)$, $w_{inact} \in (0,1)$, $w_{new} \in (0,1)$
Initialize: $w_0 \leftarrow \left(\frac{1}{N}, \cdots, \frac{1}{N}\right)$

1: **for** $t = 1, 2, \cdots$ **do**
2: Generate θ_t from $\mathcal{P}_N$ according to weights w_{t-1}
3: Play $A_t \leftarrow \arg\max_{a\in\mathcal{A}} \mathbb{E}_{\theta_t}[R(Y)|A_t = a]$
4: Observe $Y_t \sim P_{\theta^*}(\cdot|A_t)$
5: **for** $i \in \{1, 2, \cdots, N\}$ **do**
6: $\widetilde{w}_{t,i} = w_{t-1,i}\, P_{\theta^{(i)}}(Y_t|A_t)$
7: **end for**
8: $w_t \leftarrow$ normalize $\widetilde{w}_t$
9: **if** CONDITION$(w_t, N, f_{del}, w_{inact})$ = True **then**
10: $\mathcal{I}_{del} \leftarrow$ the indices of the lowest weighted $\lceil f_{del} N \rceil$ particles in $\mathcal{P}_N$
11: $\{\theta^{(i)} : i \in \mathcal{I}_{del}\} \overset{\text{replace}}{\leftarrow}$ RPTS-Exploration
12: $w_{t,i} \leftarrow \frac{w_{new}}{\lceil f_{del} N \rceil}$ for each $i \in \mathcal{I}_{del}$
13: normalize w_t
14: **end if**
15: **end for**

CONDITION$(w_t, N, f_{del}, w_{inact})$:
$w'_t \leftarrow$ sort w_t in ascending order
If $\sum_{i=1}^{\lceil f_{del} N \rceil} w'_{t,i} \le w_{inact}$: Return True
Else: Return False

RPTS-Exploration:
$\mu_t \leftarrow \mathbb{E}_{\theta\sim w_t}[\theta]$, $\Sigma_t \leftarrow \mathbb{E}_{\theta\sim w_t}[(\theta - \mu_t)(\theta - \mu_t)^T]$
Generate $\lceil f_{del} N \rceil$ particles $\overset{i.i.d.}{\sim}$ $\mathcal{N}(\mu_t, \frac{1}{K}\mathrm{tr}(\Sigma_t) I_K)$, project to Θ

Steps 1-8 of RPTS are the same as PTS (Algorithm 2 in [1]). The difference is that RPTS adds steps 9-14. Three new hyper-parameters are introduced: f_{del}, the fraction of particles to delete; w_{inact}, the weight threshold for deciding inactive particles; w_{new}, the new (aggregate) weight of regenerated particles. The CONDITION in Step 9 checks if f_{del} fraction of the particles become inactive. If so, we find the lowest weighted f_{del} fraction of the particles (Step 10), delete them, and regenerate the same number of particles through RPTS-Exploration (Step 11). In RPTS-Exploration, we first calculate the empirical mean μ_t and covariance matrix Σ_t of all the particles based on their current weights w_t[1], i.e. $\mu_t = \sum_{i=1}^{N} w_{t,i}\theta^{(i)}$ and $\Sigma_t = \sum_{i=1}^{N} w_{t,i}\left(\theta^{(i)} - \mu_t\right)\left(\theta^{(i)} - \mu_t\right)^T$, then generate the new particles according to a multi-variate Gaussian distribution. I_K is the $K \times K$ identity matrix. We use $\frac{1}{K}\mathrm{tr}(\Sigma_t) I_K$ as the covariance matrix instead of Σ_t, in case Σ_t is or close to singular. This particle regeneration strategy requires that the parameter space Θ is a subset of $\mathbb{R}^K$. If a newly generated particle is outside of Θ, we project it to Θ in any natural way.[2] Step 12 means that the newly generated $\lceil f_{del} N \rceil$ particles are assigned a total weight of w_{new} and each of them has the same weight.

The recommended numerical values of the three hyper-parameters for RPTS (Algorithm 1) are $f_{del} = 0.8$, $w_{inact} = 0.001$, and $w_{new} = 0.01$. The behavior of the algorithm is relatively insensitive to these values, but further tuning may be beneficial in a given application. We comment on how these values influence the performance of the algorithm in the rest of this section.

f_{del}: Analysis for Bernoulli bandits (Corollary 7 in [1]) and empirical evidence for other bandit models indicate that with high probability all but a few particles eventually decay in PTS. Hence it may be attempting to make f_{del} very large. However, since the set of decaying particles is random, it may happen that some fit particles end up decaying. Also, a not-so-bad particle may have an oscillating weight due to counter-

[1] According to the RPTS heuristic, one may expect to calculate μ_t and Σ_t based on the weights of the surviving particles only, instead of all the particles. But because the surviving particles have a total weight of at least $1 - w_{inact}$, close to 1, the difference is negligible.

[2] Alternatively, we can reject it and regenerate until it is in Θ.

reinforcing effects[3] and thus may have low weight at times. Making f_{del} not too large gives those unfortunate fit and not-so-bad particles a chance to survive. We have tried $f_{del} = 0.8$ and $f_{del} = 0.5$ and both work fine.

w_{inact}: The value of w_{inact} should be small, but if it is too small, it may take a long time for the CONDITION in Step 9 to become true, especially when the particles become concentrated in a small subset of the parameter space.

w_{new}: The value of w_{new} should be small, but strictly larger than w_{inact}. There are three aspects of consideration here. First, it is desirable that the weight re-balancing in Step 13 due to normalization has minimal effect on the weights of the surviving particles. We discovered through experiments that it is good for heavy weight particles to remain heavily weighted. Therefore w_{new} should be small. Second, w_{new} should be larger than w_{inact}, because otherwise, the newly generated particles in a step will be immediately deleted in the next step. Third, the purpose of setting the value of w_{new} is to give some initial weights to the new particles so that they can participate in the weight updating in the subsequent steps. If a new particle is fit, its weight will boost up exponentially fast; if a new particle is unfit, it will decay exponentially fast. Therefore, the initial weights assigned to these new particles should not significantly affect their chance of survival and their long-term weight dynamics. Thus, as long as w_{new} is fairly small and larger than w_{inact}, the choice of its actual value may not make much difference qualitatively.

IV. Simulation

We run simulations [4] to compare RPTS with PTS and TS (if possible) on the following representative bandit problems.

Example 1 (Bernoulli bandit)**.** Let K be a positive integer. A Bernoulli bandit problem depicts a player who picks an arm indexed by $a \in \{1, \cdots, K\}$ at each step, which generates a reward of either 0 or 1 according to a Bernoulli distribution parameterized by $\theta_a^* \in [0, 1]$, fixed and unknown. This is a stochastic bandit problem with $\mathcal{A} = \{1, 2, \cdots, K\}$, $\mathcal{Y} = \{0, 1\}$, $\Theta = [0, 1]^K$, $P_\theta(\cdot|a) \sim \text{Bernoulli}(\theta_a)$, and $R(y) = y$. This is a bandit problem with separable actions – the observation distribution for each action is parametrized by a corresponding coordinate of θ^*.

Example 2 (Max-Bernoulli bandit)**.** Let K, M be positive integers with $K \geq 2$ and $M < K$. A max-Bernoulli bandit problem is similar to the Bernoulli bandit, with arms indexed by $\{1, \cdots, K\}$ and each arm is associated with a Bernoulli distribution with a fixed and unknown parameter θ_a^*. The difference is that, in a max-Bernoulli bandit problem, the player picks M different arms at each step instead of one. The reward is the maximum of the M binary values generated by the M selected arms. This problem can be formulated as a stochastic bandit problem with $\Theta = [0, 1]^K$, $\mathcal{A} = \binom{[K]}{M} = \{S \subset [K] : |S| = M\}$, $\mathcal{Y} = \{0, 1\}$. Given $a = (a_1, \cdots, a_M) \in \mathcal{A}$, observe $Y = \max_{m \in [M]} X_m$, where $X_m \sim \text{Bernoulli}(\theta_{a_m}^*)$. That is, the observation model is $P_\theta(\cdot|a) \sim \text{Bernoulli}\left(1 - \prod_{m \in M}(1 - \theta_{a_m})\right)$. The reward function is $R(y) = y$. Actions in the max-Bernoulli bandit problem are not separable. The number of actions, $\binom{K}{M}$, can be much larger than K, the dimension of the parameter space. The problem is considered in [4].

Example 3 (Linear bandit)**.** A linear bandit problem has two parameters: a positive integer K and $\sigma_W^2 > 0$. It is a stochastic bandit problem with $\Theta = \mathbb{R}^K$, $\mathcal{A} = \mathcal{S}^{K-1} = \{x \in \mathbb{R}^K : \|x\|_2 = 1\}$, the surface of a unit sphere in $\mathbb{R}^K$, $\mathcal{Y} = \mathbb{R}$ and $R(y) = y$. Given an action $a \in \mathcal{A}$, we observe $Y = \langle \theta^*, a \rangle + W$, where $\theta^* \in \Theta^K$ is fixed and unknown and $W \sim \mathcal{N}(0, \sigma_W^2)$ is some Gaussian noise. That is, the observation model is $P_\theta(\cdot|a) \sim \mathcal{N}(\langle \theta, a \rangle, \sigma_W^2)$. The problem is named "linear" because the expected reward in each round is an unknown linear function of the action taken. This is an example of a bandit problem in which the dimension of the parameter space is finite, but the number of actions is infinite.

Simulation results are shown in Figure 1. For Bernoulli and linear bandits, TS is implemented as a benchmark. For max-Bernoulli bandit, it is not clear how TS can be implemented. For linear bandit, TS is implemented exactly by a Kalman filter. Each curve is obtained by averaging over 200 independent simulations. In each simulation of PTS or RPTS, for Bernoulli and max-Bernoulli bandits, the initial particles are generated uniformly at random from $[0, 1]^K$. For linear bandits, the initial set of particles are generated uniformly at random from the unit ball in $\mathbb{R}^K$. That is based on the assumption that we already know that θ^* is in the unit ball before running the algorithm. In practice, such knowledge may not be available and a common practice is to use a distribution that spreads out wide enough so that it should cover θ^*. For the purpose of demonstrating the performance of PTS and RPTS here, our practice should be acceptable.

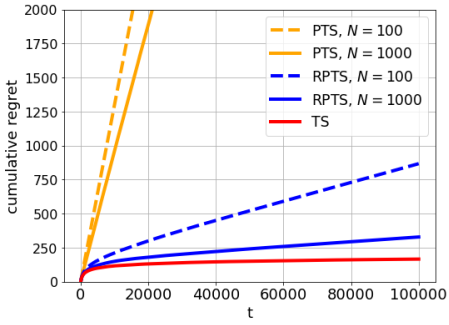
Observations: (1) For PTS, the improvement from $N = 100$ to $N = 1000$ is marginal in all setups. (2) For RPTS, the improvement from $N = 100$ to $N = 1000$ is non-negligible in most setups. (3) RPTS outperforms PTS uniformly in all setups, and in most of them, the improvement is significant. (4) For Bernoulli bandit with $K = 100$ (Figure 1(b) and (c)), RPTS even outperforms TS for a considerably long time. The reason is unclear.

V. Discussion and Conclusion

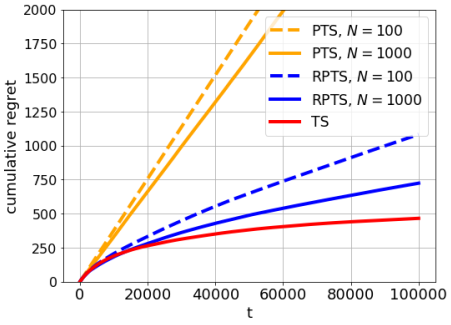
Limitations. We don't have any statistical guarantee for the performance of RPTS. In fact, in all simulations, the cumulative regret curves of RPTS eventually grow linearly at large times. This suggests that RPTS may not be consistent theoretically. But it doesn't attenuate the practical value of RPTS, given the time scale in simulations.

[3] In brief, this means the weights of a set of particles positively influence each other. But because they sum up to one, they cannot all keep increasing. So the particles' weights do not evolve monotonically, but oscillate unending. This is explained and illustrated in detail for two-arm Bernoulli bandits in [3] Appendix B.2.1.

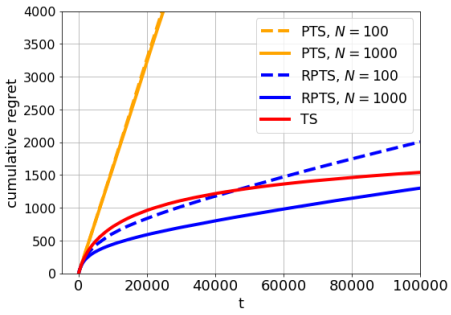
[4] Code available at: https://github.com/zeyuzhou91/CISS2023_RPTS



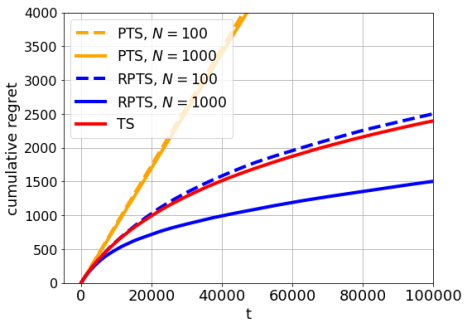
(a) Bernoulli bandit, $K = 10$ $\theta^* = [0.05, 0.10, \cdots, 0.50]$.



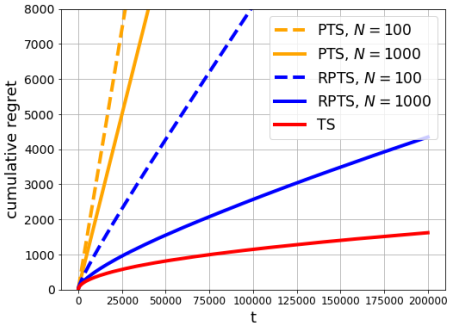
(b) Bernoulli bandit, $K = 10$ $\theta^* = [0.51, 0.52, \cdots, 0.60]$.



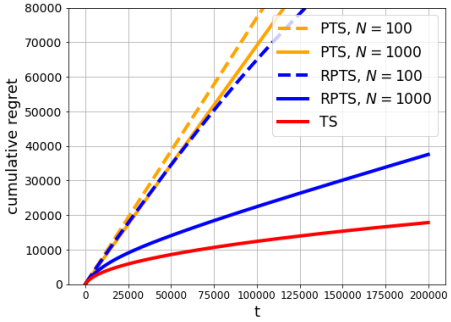
(c) Bernoulli bandit, $K = 100$ θ^* consists of $N = 100$ points uniformly spaced over [0.3,0.8].



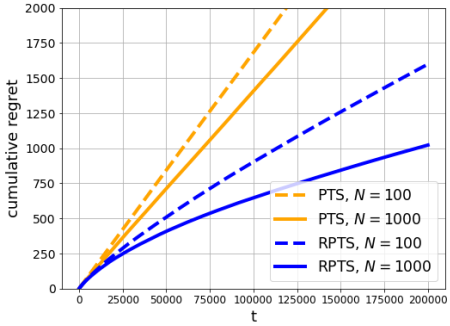
(d) Bernoulli bandit, $K = 100$ θ^* consists of $N = 100$ points uniformly spaced over [0.5,0.7].



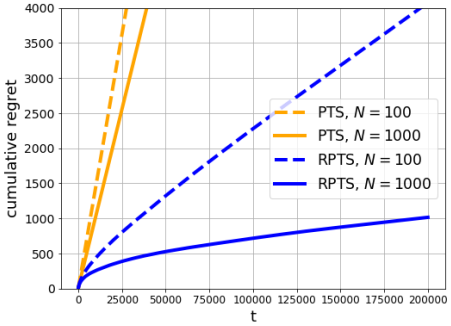
(e) Linear bandit, $K = 10$, $\sigma_W^2 = 0.1$, $\theta^* = [0.2, \cdots, 0.2]$.



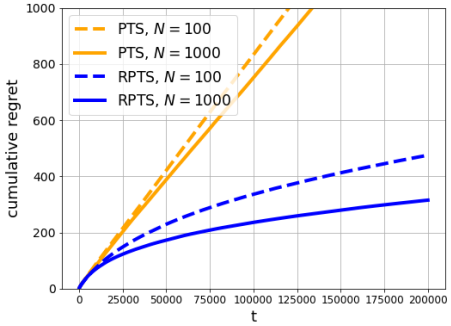
(f) Linear bandit, $K = 100$, $\sigma_W^2 = 0.1$, $\theta^* = [0.08, \cdots, 0.08]$.



(g) Max-Bernoulli bandit, $K = 10$, $M = 3$ $\theta^* = [0.51, 0.52, \cdots, 0.60]$.



(h) Max-Bernoulli bandit, $K = 10$, $M = 3$ $\theta^* = [0.05, 0.10, \cdots, 0.50]$.



(i) Max-Bernoullin bandit, $K = 100$, $M = 5$ θ^* consists of $N = 100$ points uniformly spaced over [0.3,0.8].

Fig. 1: Simulation results: comparison of RPTS, PTS, and TS for Bernoulli, linear and max-Bernoulli bandits.

Conclusion. This paper proposes RPTS, an improvement of PTS and a practical variation of Thompson sampling. RPTS exploits the particle dynamics of PTS and is based on a simple heuristic that deletes essentially inactive particles and regenerate new particles in the vicinity of survivors. We show empirically that RPTS significantly outperforms PTS in a set of representative bandit problems. This demonstrates the possibility of improving PTS without using more particles. A direction for future work is to design a PTS-based algorithm with non-static particles that has theoretical consistency guarantees.

REFERENCES

[1] Z. Zhou and B. Hajek, "Particle thompson sampling with static particles," in *Proceedings of the 57th Annual Conference on Information Sciences and Systems*, 2023.

[2] A. Doucet, N. De Freitas, N. J. Gordon *et al.*, *Sequential Monte Carlo methods in practice*. Springer, 2001, vol. 1, no. 2.

[3] Z. Zhou, B. Hajek, N. Choi, and A. Walid, "Regenerative particle thompson sampling," *arXiv preprint arXiv:2203.08082*, 2022.

[4] A. Gopalan, S. Mannor, and Y. Mansour, "Thompson sampling for complex online problems," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, pp. I–100–I–108.