Exploring Online Learning Algorithms for Causal Bandits

Yash Shah, Gaurav Didwania, Rupesh, Kumar Saurav
Department of Computer Science and Engineering
Indian Institute of Technology Bombay, Mumbai, India
{160050002, 160050020, 160050042, 160050057}

Abstract

The causal bandit problem is an extension of the conventional multi-armed bandit problem in which the *arms* available are not independent of each other, but rather are correlated within themselves in a Bayesian graph. This extension is more natural, since day-to-day cases of bandits often have a causal relation between their actions and hence are better represented as a causal bandit problem. Moreover, the class of conventional multi-armed bandits lies within that of causal bandits, since any instance of the former can be modeled in the latter setting by using a Bayesian graph with all independent variables. In this project, we try to explore the application of standard multi-armed bandit algorithms and a variant of Thompson Sampling to causal bandits, and also propose two extensions and a new graph-based ϵ_t-greedy approach for learning in such settings. We also publicly release our code[1] to promote future research in this domain.

Keywords Causal Bandits · Thompson Sampling · ϵ_t-greedy · Multi-Armed Bandits

1 Introduction

The multi-armed bandit is a stochastic scheduling problem wherein a user can choose amongst an independent set of *arms* (or actions), and the reward that they will get is stochastically dependent on the arm pulled. This problem covers many real life instances (for eg. which advertisement will have a higher click rate) and derives its name from the one-arm slot machines in casinos. Albeit its simple nature and relaxing assumptions, the multi-armed bandit can model a lot of problems making it widely prevalent.

However, there are several scenarios in which multi-armed bandits cannot be applied due to contradiction(s) with one or more of its fundamental assumptions. A very common example, especially in the context of our country, is related to the cropping season. Consider a young farmer who just started farming this season, naturally with the objective of procuring maximum yield. He is aware of various factors affecting the same, say quality of seeds, organic manure, pesticides and fertilizers, but can afford to spend on only one due to a fixed budget; other factors are out of his reach and obtained (randomly) via government schemes. Only at the end of the season can he estimate the quantity of each 'out-of-reach' resource consumed and the resulting yield, and plan his approach/expenditure for the next season. Another thing to be noted here is that the decisions of the farmer on spending his budget on a resource are not independent — if the farmer is already getting good quality of seeds (which are, say, robust to diseases) and decides not to spend on them, it is intuitive that he won't have to spend on anti-disease fungicides/pesticides either.

Clearly, a multi-armed bandit is not the best formulation for this problem since the actions are not independent. Instead, such instances are better modeled by causal bandits in which we assume that a Bayesian graph governs the dependencies amongst various inputs and other hidden factors, with the final 'reward' also being a node in the graph. Various algorithms have already been proposed for causal bandits that try to model the Bayesian graph structure and distribution, and consequently find the best regret-minimizing action [1, 2]. Our objective in this project is to compare such an algorithm [2] with conventional multi-armed bandit approaches and extend it to work better in our setting (more on that in section 2), as well as develop an ϵ_t-greedy algorithm which estimates the graph distribution through exploration and then exploits it to minimize cumulative regret.

[1] https://github.com/rs9899/causal-bandit

2 Problem Statement

A causal bandit instance can be described as follows. We are given a set of random variables $\mathcal{X} = \{X_1,\ X_2\ \ldots\ X_N\}$, a reward variable Y and a set of allowed *actions* $\mathcal{A}$. The dependencies between $\mathcal{X}$ and Y are represented using a causal graph $\mathcal{G}$, and the amount of information we have beforehand on $\mathcal{G}$ can be varied. Each action $a_t \in \mathcal{A}$ is an *intervention* of the form $do(\mathbf{X}_t = \mathbf{x}_t)$ (where $\mathbf{X}_t = \{X_{j_1},\ X_{j_2}\ \ldots X_{j_m}\},\ X_{j_i} \in \mathcal{X}\ \forall i \in [m]$) that involves assigning X_{j_i} the corresponding value in $\mathbf{x}_t$ and removing all incoming edges for X_{j_i} in $\mathcal{G}$ to obtain a *mutilated* graph $\mathcal{G}_{a_t}$ (the empty intervention, $do()$, is also permitted). *After* the intervention has been performed, values of the non-intervened variables $\mathcal{X}_c = \mathcal{X} \setminus \mathbf{X}_t$ and the reward y_t are sampled from the distribution of $\mathcal{G}_{a_t}$. This process is repeated upto a fixed horizon T, and the objective is to choose $\{a_t\}_{t=1}^T$ such that the cumulative (or simple) regret is minimised.

For the purpose of this project, we restrict ourselves to those causal bandit instances where the *structure* of $\mathcal{G}$ is known beforehand but the joint probability distribution (i.e. $\mathbb{P}(\mathcal{X} \cup \{Y\})$ is unknown. Moreover, each $X \in \mathcal{X}$ and Y is a $\{0, 1\}$-valued random variable, and the set of allowed actions $\mathcal{A}$ is the set of all possible size-1 interventions (i.e. $|\mathbf{X}_t| = |\mathbf{x}_t| = 1$, and hence $|\mathcal{A}| = 2N$). We shall minimise cumulative regret in our algorithms, which is defined as $R_T = \mu^* T - \mathbb{E}\left[\sum_{t=1}^T y_t\right]$, where $\mu^* = \mathbb{E}[Y = 1|a^*]$.

3 Related Work

The causal bandit problem was first formulated in Lattimore et al. [1], where they assumed the causal graph $\mathcal{G}$ to be completely known beforehand. They suggested algorithms to minimise simple regret, which is defined as $R = \mu^* - \max_{a \in \mathcal{A}} \mu_a$ where $\mu_a = \mathbb{E}[Y|a]$, in two different scenarios — (i) when $\mathcal{G}$ was parallel i.e. all non-reward variables were independent of each other, and (ii) when $\mathcal{G}$ was arbitrary. Lattimore et al. [1] used the graph structure by looking at parents of Y and optimising a distribution over $\mathcal{A}$, with the distribution itself changing to minimise the time taken to reach the best action.

Sachidananda and Brunskill [2] extend Thompson Sampling to the causal bandit setting of Lattimore et al. [1] by breaking the procedure into two parts — given an intervention $a = do(\mathbf{X} = \mathbf{x})$, they first try to estimate a distribution over the possible assignments of parent variables of Y given a, and then they estimate the reward distribution given a particular parent configuration; both distributions are updated after each trial as new samples are observed.

For many real-life causal bandit instances, it makes sense to minimise cumulative regret for a fixed horizon which is not explored in detail in these papers. A comprehensive analysis and comparison of the proposed algorithms with classical multi-armed bandit algorithms is also lacking. Moreover, in most practical scenarios, the *structure* of the causal graph is known beforehand — we know how variables affect each other as well as the reward — it is the exact *probability distribution* which is unknown. We believe that these algorithms can be modified to exploit this additional information, which is one of the few things we plan to explore in this project.

4 Agents Implemented

We implement a variety of agents in order to carry out an investigation to assess how OC-TS fares against conventional multi-armed bandit algorithms and whether there is any scope of improvement. Other than some standard/already suggested approaches, we also propose a novel solution using an ϵ_t-greedy technique and a few modifications to the original OC-TS algorithm proposed by Sachidananda and Brunskill [2].

4.1 UCB, KL-UCB, Thompson Sampling

As a baseline against which the performance of all subsequent agents would be compared, we implement these conventional multi-armed bandit algorithms for our causal bandit setting. These agents simply ignore the causal graph dependencies amongst variables, and treat them as being independent instead (this is achieved by ignoring sampled values of all $X \in \mathcal{X}_c$ and using that of just the reward variable, Y).

We provide the implementations of these agents via `UCBAgent`, `KL_UCBAgent` and `TSAgent` in our code.

4.2 OC-TS

This agent is a direct re-implementation of the algorithm proposed by Sachidananda and Brunskill [2] (for their setting of unknown causal graph). We provide the implementation of this agent via `OC_TSAgent` in our code.

4.3 OC-TS Enhanced Dirichlet

The original OC-TS algorithm [2] assumed that the graph structure was unknown, and hence treated each $X \in \mathcal{X}$ as a potential parent to Y. For the same reason, it maintained a Dirichlet distribution with $2^N \times |\mathcal{A}|$ parameters to account for all possible assignments of $Pa(Y)$. However, in our setting, the graph structure is known beforehand; hence, the Dirichlet distribution can be pruned to remove extra variables from $Pa(Y)$. For a graph topology where each node has atmost k parents (eg. linear with $k = 1$), this modification reduces computational complexity from $\mathcal{O}(N \cdot 2^N)$ to $\mathcal{O}(N)$. This pruning of the Dirichlet distribution (over assignments of $\mathcal{X}$) can also be thought of as a marginalization step over all assignments of $X \in \mathcal{X} \setminus Pa(Y)$; this implies that the distribution over variable assignments is now more 'localized' towards $Pa(Y)$ assignments only, and so the model is expected to estimate the distribution much faster.

We provide the implementation of this agent via `OC_TS_ED_Agent` in our code.

4.4 OC-TS Empirical

Instead of drawing from the Dirichlet distribution to get the probability of a particular assignment of $Pa(Y)$ as in Sachidananda and Brunskill [2], we simply use the empirical probability, using the history of runs to calculate the probability of a particular assignment. If $\#_t\{S\}$ denotes the number of times event S occurred upto timestep t and Z_k denotes the k^{th} assignment of $Pa(Y)$, then for $A \in \mathcal{A}$ we have

$$P_t(Pa(Y) = Z_k \mid A) \approx \frac{\#_t\{Pa(Y) = Z_k \mid A\}}{\sum_{j=1}^{2^{|Pa(Y)|}} \#_t\{Pa(Y) = Z_j \mid A\}}$$

Initially, all assignments are considered to be equally probable; hence, $\#_0\{\cdot\} = 1$. These counts are updated as sampled values of $\mathcal{X}_c$ are observed. We still treat each $X \in \mathcal{X}$ as a potential parent of Y (since we wanted to separately assess the contribution of each modification); hence, $Pa(Y) = \mathcal{X}$. The remaining part of the original OC-TS algorithm is kept unchanged. We provide the implementation of this agent via `OC_TS_Empirical_Agent` in our code.

4.5 Graph ϵ_t-greedy

We propose a simple, novel algorithm combining ideas from Bayesian graph inference and ϵ_t-greedy algorithm for multi-armed bandits. This algorithm assumes that the causal graph structure is available beforehand (but its distribution remains unknown), which is consistent with our setting. It is a model-based strategy for finding the optimal intervention, which estimates the graph distribution using the history of sampled values of $\mathcal{X}_c$ during exploration and then uses this estimate to evaluate the expected reward for each action during exploitation. We describe the algorithm in detail below.

For each timestep t, our objective is to choose $a^* \in \mathcal{A}$ such that $a^* = \arg\max_{a \in \mathcal{A}} \mathbb{E}[Y = 1 \mid a]$. Since Y is a Bernoulli random variable, $\mathbb{E}[Y = 1 \mid a] = P(Y = 1 \mid a)$. Since each action $a \in \mathcal{A}$ generates a ***different*** mutilated graph, this probability can be naively computed by maintaining a separate count table for each intervention, and using that of a for computing $P(Y = 1 \mid a)$. This method is suboptimal, both in terms of memory as well as the manner in which $\mathcal{X}_c$ values are used. It is not difficult to realise that even *different* mutilated graphs share a common structure for nodes that are descendants of the intervened variables, and hence it makes sense to couple such nodes together for efficient usage of both memory and $\mathcal{X}_c$ values. We do so by exploiting a property of Bayesian graphs, which states that a node is conditionally independent of all other nodes given its parents.

For every node in the original (non-intervened) causal graph, we maintain two counts of occurrences for every possible assignment of its parents, one each for when the node value is 0 and 1 (i.e. $2 \times 2^{|Pa(v)|}$ values for $v \in \mathcal{X} \cup \{Y\}$). These counts represent the events $\{v = 0 \mid Pa(v) = Z_k\}_{k=1}^{2^{|Pa(v)|}}$ and $\{v = 1 \mid Pa(v) = Z_k\}_{k=1}^{2^{|Pa(v)|}}$ respectively. Now, for every intervention on the true model we will have sampled values for each node and its parents. Using these, we update the corresponding conditional probability counts of each node **except** the counts corresponding to the intervened variable, since its value was not sampled but rather given as input and hence doesn't give any signal of its conditional probability given its parents. Such a modeling design works and is useful since the local (i.e. node given its parents) structure *and* distribution remain unchanged across all mutilated graphs for all non-intervened nodes, and hence can be shared. As exploration continues, the model converges to the causal model.

We now elaborate how these *local* counts can be used to evaluate $P(Y = 1 \mid a)$. The estimated conditional probability of any node v given a particular parent assignment Z_k can be trivially computed as $\frac{\#\{v=1|Pa(v)=Z_k\}}{\#\{v=0|Pa(v)=Z_k\}+\#\{v=1|Pa(v)=Z_k\}}$. We propose two ways in which this conditional probability can be used to evaluate $P(Y = 1 \mid a)$ during exploitation.

4.5.1 Repeated Forward Sampling

Once the conditional probabilities have been computed for each node, we have a complete estimate of the causal graph (both structure and distribution), say $\mathcal{G}_{\text{est}}$. For each action $a \in \mathcal{A}$, we sample M values of Y from the graph obtained by mutilating $\mathcal{G}_{\text{est}}$ for intervention a using forward sampling, and compute $P(Y = 1 \mid a)$ as follows:

$$P(Y = 1 \mid a) \approx \frac{\text{number of occurrences of } Y = 1}{M}$$

This method, although time-consuming, was very easy to implement and was mainly used as a proof-of-concept for our Graph ϵ_t-greedy algorithm. We provide its implementation via `E_graphAgent` with `switch=1` in our code.

4.5.2 Exact Inference

This is non-trivial, since we have to consider both assignments and interventions (which are very different; see the analysis section below). Algorithm 1 describes our method in more detail. Essentially, we start from the node corresponding to Y and traverse the causal graph bottom-up multiplying with per-node conditional probabilities and marginalizing over all valid parent assignments. Mathematically,

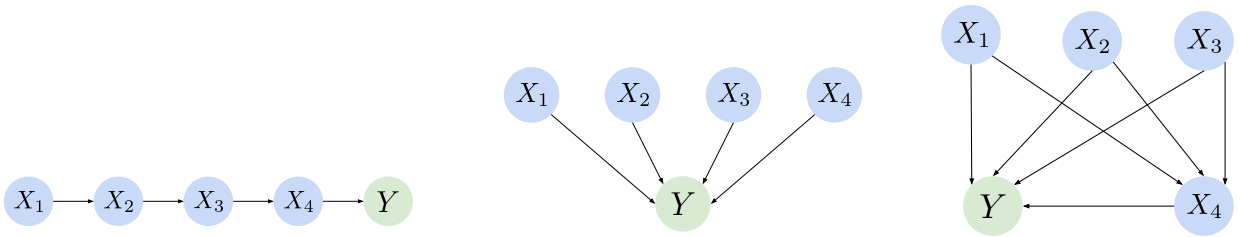
$$P(Y = 1 \mid a) = \sum_{k=1}^{2^{|Pa(Y)|}} P(Y = 1 \mid Pa(Y) = Z_k) \cdot P(Pa(Y) = Z_k \mid a)$$

The first term of the summation is available from the count tables of Y, and the second term is recursively computed as shown in Algorithm 1. We provide its implementation via `E_graphAgent` with `switch=0` in our code.

Algorithm 1 Calculate $\Pr(val \mid action)$

$X \leftarrow val$.keys()
if len$(X) == 0$ **then**
  return 1.0
end if
$var \leftarrow X[0]$
$value \leftarrow val[var]$
del $val[var]$
if $var \in action$ **then**
  if $value == action[var]$ **then**
    return $\Pr(val \mid action)$
  else
    return 0.0
  end if
end if
$parentVar \leftarrow$ parent(var)
if len$(parentVar) == 0$ **then**
  $p \leftarrow$ prob$[var][value]$
  return $p \cdot \Pr(val \mid action)$
end if
$allAssign \leftarrow$ all_possible_assignments$(parentVar)$
$validAssign \leftarrow$ valid_assignments$(allAssign,\ val)$
$p \leftarrow 0.0$
for $parentAssignment$ *in* $validAssign$ **do**
  $pGivenParent \leftarrow$ prob$[var][parentAssignment][value]$
  $newVal = parentAssignment \cup val$
  $p \mathrel{+}= pGivenParent \cdot \Pr(newVal \mid action)$
end for
return p

In this algorithm, `all_possible_assignments` returns a list of all $2^{|Pa(v)|}$ parent assignments for given node v, and `valid_assignments` filters those assignments out from this list which are not compatible with the values in the *val* dictionary. $P(Y = 1 \mid a)$ is computed by invoking this algorithm with *val* as $\{Y : 1\}$ and *action* as a (which is also of the form $\{v : 0\}$ or $\{v : 1\}$ for $v \in \mathcal{X} \cup \{Y\}$).



(a) Linear (b) Disjoint or Independent (c) Random

Figure 1: Representative models from each topology.

5 Experiments

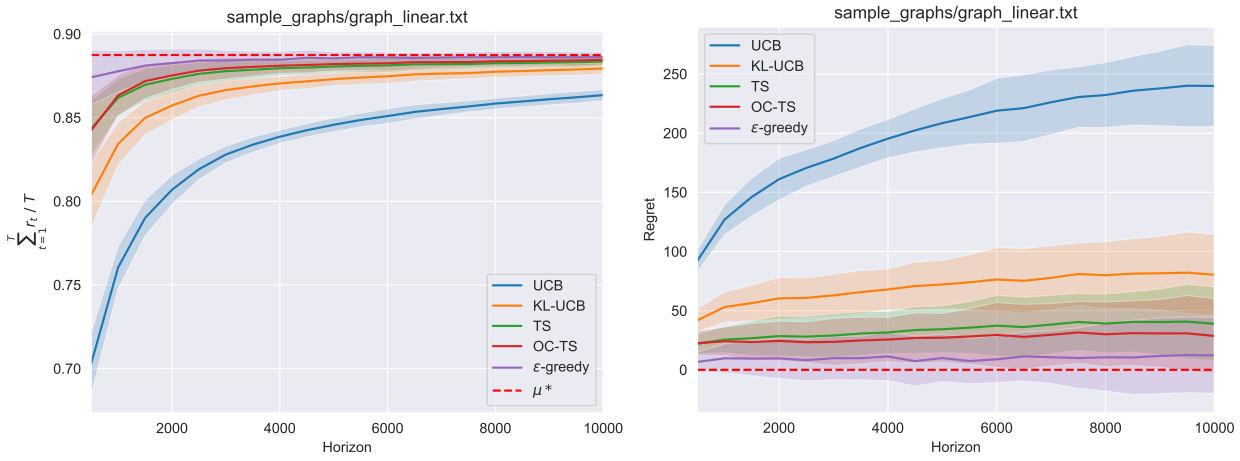
5.1 Graph topologies

We have considered three types of Bayesian graph models (described below) with different topologies representing the causal dependencies amongst Bernoulli variables to show that our proposed modifications/algorithms perform better than existing ones. Figure 1 shows the representative model from each topology chosen for our experiments.

- **Linear.** Every variable excluding one root variable has exactly one parent. The idea here is that the agent must learn that the best action is the one in which the closest possible ancestor is intervened (to either a value of 0 or 1 depending on the distribution).
- **Disjoint** or **Independent.** Here all variables (except the reward variable) are pairwise independent and are parents of the reward variable. The best action in this case is the one in which the marginalised probability over non-intervened variables is maximum.
- **Random.** It refers to a completely random Bayesian graph obtained by iteratively adding variables and randomly choosing whether each previously added variable is its parent or not.

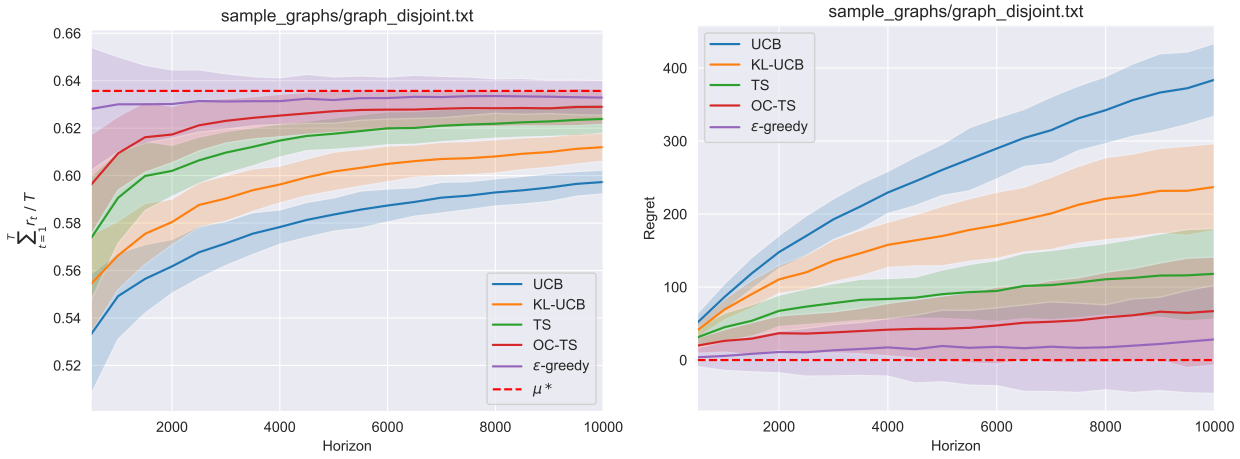
5.2 Graph ϵ_t-greedy vs. All

We present the results of our experiments here. For each topology, we compare the performance of 5 agents by plotting the estimated value of $P(Y = 1 \mid a^*)$ (which we call $\hat{\mu} = \frac{1}{T}\sum_{t=1}^{T} r_t$) and regret ($\mu^* T - \sum_{t=1}^{T} r_t$) as a function of horizon. These plots are shown in figures 2, 3 and 4. Each plot is obtained by averaging results over 20 random seeds; we also shade the regions of uncertainty corresponding to a deviation of $\pm\sigma$.



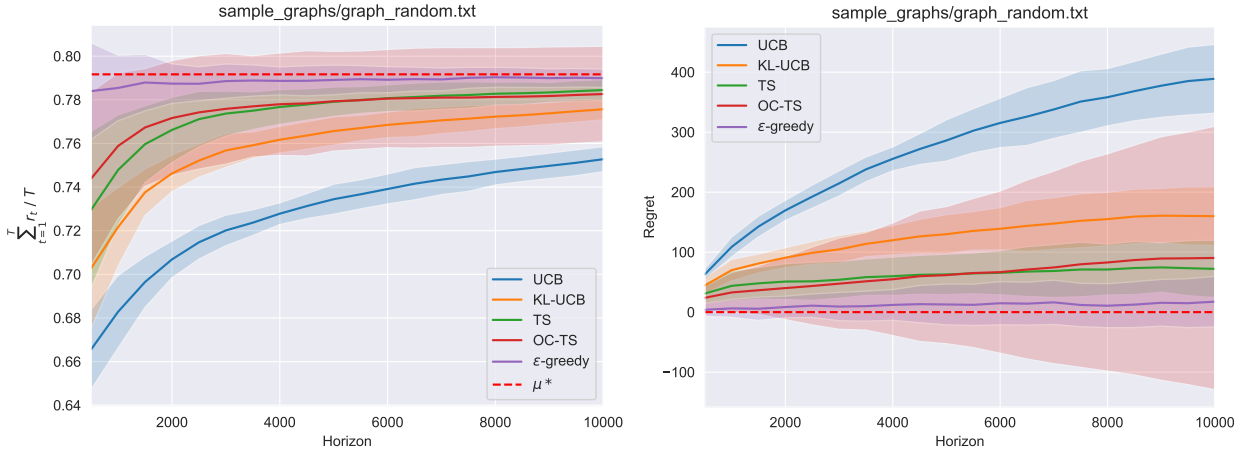
(a) $\hat{\mu}$ vs. Horizon (b) Regret vs. Horizon

Figure 2: Results for linear topology.



(a) $\hat{\mu}$ vs. Horizon (b) Regret vs. Horizon

Figure 3: Results for disjoint topology.



(a) $\hat{\mu}$ vs. Horizon (b) Regret vs. Horizon

Figure 4: Results for random topology.

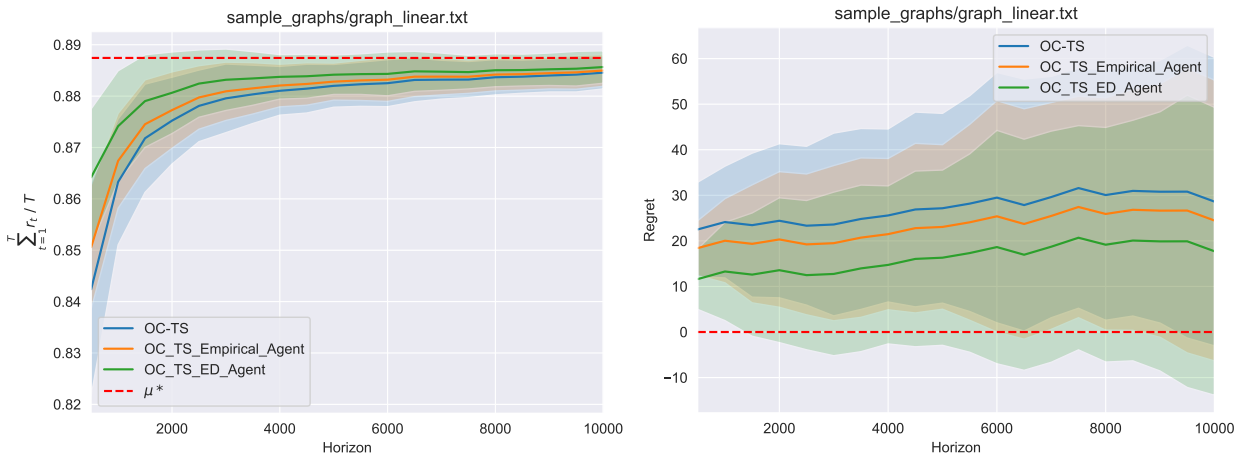
The value of ϵ_t was decayed according to the law $\epsilon_t = \frac{\epsilon_0}{1+\frac{t}{\alpha}}$ where $\epsilon_0 = 0.05,\ \alpha = 200$ (these values were empirically obtained by tuning for best performance). It can be easily observed that our proposed algorithm, Graph ϵ_t-greedy, performs better (lower regret and faster convergence to true μ^*) than OC-TS and other conventional multi-armed bandit algorithms across all three topologies.

5.3 Modifications to OC-TS

We compare our modification to OC-TS with the original algorithm on the causal graph corresponding to linear topology and provide results in Figure 5. We can observe that both the modifications (pruning the Dirichlet distribution and replacing it by an empirical estimate) perform better than the original algorithm; this was expected according to our reasoning in earlier sections.

5.4 Importance of Mutilation

In order to emphasize the importance of mutilating a graph while intervening, we construct a dummy algorithm along similar lines as our Graph ϵ_t-greedy as follows. Instead of a separate count table for each node, we maintain a single, global count table of size $2^{|\mathcal{X}|}$ shared by all nodes. For every intervention, it was updated with the sampled values of



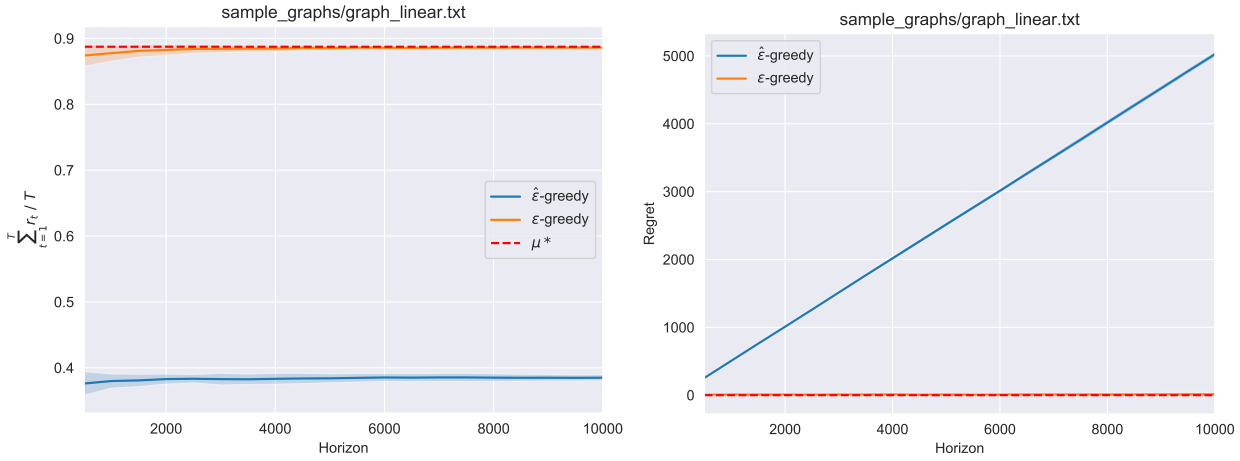
(a) $\hat{\mu}$ vs. Horizon

(b) Regret vs. Horizon

Figure 5: Results of the two OC-TS modifications for linear topology.

$\mathcal{X}_c$. All conditional probabilities were computed using this table during exploitation, and an ϵ_t-greedy strategy similar to ours was used for online learning. While updating the counts, this algorithm neglected the fact that all the actions are not equally probable in the Bayesian model and that the graph structure changed upon intervention. Maintaining a shared, global table is incorrect and would lead to wrong learning, which is evident from the results we obtained.

Figure 6 demonstrates this analysis, where $\hat{\epsilon}$-greedy is the dummy algorithm described above.



(a) $\hat{\mu}$ vs. Horizon

(b) Regret vs. Horizon

Figure 6: Results of un-mutilated vs. mutilated ϵ_t-greedy algorithms for linear topology.

6 Conclusion

We propose modifications to the OC-TS algorithm put forward by Sachidananda and Brunskill [2] and also suggest a novel graph-based algorithm which combines concepts from Bayesian graph inference and ϵ_t-greedy algorithm for conventional multi-armed bandits. By experimenting on and comparing with various agents and graph topologies, we demonstrate that our proposed algorithms perform much better. Though the number of experiments done by us are limited, these results can definitely become a good starting point for future research in the field. We hope that our proposed algorithms translate well to and benefit real-world applications of causal bandits.

Acknowledgement

We would like to thank Prof. Shivaram Kalyanakrishnan for teaching us Multi-Armed Bandits as a part of the course, CS747: Foundations of Intelligent and Learning Agents, and for giving us the opportunity to explore such agents further as a part of our course project.

References

[1] Finnian Lattimore, Tor Lattimore, and Mark D. Reid. Causal bandits: Learning good interventions via causal inference. In *NIPS*, 2016. URL https://papers.nips.cc/paper/6195-causal-bandits-learning-good-interventions-via-causal-inference.pdf.

[2] Vin Sachidananda and Emma Brunskill. Online learning for causal bandits. 2017. URL https://web.stanford.edu/class/cs234/past_projects/2017/2017_Sachidananda_Brunskill_Causal_Bandits_Paper.pdf.