The Hardness Analysis of Thompson Sampling for Combinatorial Semi-bandits with Greedy Oracle

Anonymous Author(s)
Affiliation
Address
email

Abstract

Thompson sampling (TS) has attracted a lot of interest in the bandit area. It was introduced in the 1930s but has not been theoretically proven until recent years. All of its analysis in the combinatorial multi-armed bandit (CMAB) setting requires an exact oracle to provide optimal solutions with any input. However, such an oracle is usually not feasible since many combinatorial optimization problems are NP-hard and only approximation oracles are available. An example [30] has shown the failure of TS to learn with an approximation oracle. However, this oracle is uncommon and is designed only for a specific problem instance. It is still an open question whether the convergence analysis of TS can extend beyond the exact oracle in CMAB. In this paper, we study this question under the greedy oracle, which is a common (approximation) oracle with theoretical guarantees to solve many (offline) combinatorial optimization problems. We provide a problem-dependent regret lower bound of order $\Omega(\log T/\Delta^2)$ to quantify the hardness of TS to solve CMAB problems with greedy oracle, where T is the time horizon and Δ is some reward gap. We also provide an almost matching regret upper bound. These are the first theoretical results for TS to solve CMAB with a common approximation oracle and also give a clue on the hardness of the convergence of TS with other approximation oracles.

1 Introduction

Stochastic multi-armed bandit (MAB) problem [20, 5, 3] is a classical online learning framework. It has been extensively studied in the literature and has a wide range of applications [13, 12, 20]. This problem is modeled by a T-round game between the learning agent and the environment. The environment contains an arm set and each arm is associated with a reward distribution. At each round t, the agent first selects an arm, then the environment generates a random reward for each arm from its reward distribution. The agent then obtains the reward of the selected arm. The objective of the agent is to accumulate as many expected rewards over T rounds, or equivalent to minimizing the cumulative expected regret, which is defined as the cumulative difference between the expected reward of the optimal arm and the selected arm over T rounds. To achieve this long-horizon goal, the learning agent has to face the dilemma of *exploration* and *exploitation* in each round. The former aims to try arms that have not been observed enough times to get a potentially higher reward, the latter focuses on the arm with the best observed performance so far to maintain a high profit. How to balance the tradeoff between exploration and exploitation is the main focus of the MAB algorithms.

One of the most popular bandit algorithms is the upper confidence bound (UCB) [5]. This algorithm aims to construct confidence sets for unknown expected rewards and selects arms according to their highest upper confidence bounds. The UCB-type algorithms have been widely studied and theoretically guaranteed with regret upper bound of order $O(\log T/\Delta)$, where Δ is the minimum gap

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

between the expected reward of the optimal arm and any suboptimal arms. Thompson sampling (TS) [3, 26] is another popular method to solve MAB problems. It is a randomized algorithm based on Bayesian ideas, which maintains an iteratively updated prior distribution for each arm and chooses arms according to their probabilities of being the best one. The TS algorithm was introduced in the 1930s [28], but its theoretical analysis is open until recent years [15, 2]. It was proven that the regret upper bound of the TS algorithm is of the same order of $O(\log T/\Delta)$ in MAB problems [3]. Benefitting from the advantages of easier implementation and better empirical performance compared to UCB, the TS-type algorithms attract more attention in recent years.

Despite its importance, the MAB framework may fail to model many real applications since the agent's action is usually not a single arm but a combination of several arms. This motivates the study on the combinatorial MAB (CMAB) problem [11, 9, 17, 31, 7, 29, 30, 24]. In the CMAB framework, the agent selects a combination of base arms as an action to play in each round. All outcomes of these selected arms are then revealed to the agent, which scheme is called *semi-bandit feedback* and is widely studied in the literature [7, 29, 32, 30, 14, 24]. Such CMAB framework could cover many real application scenarios including probabilistic maximum coverage (PMC) [6], online influence maximization (OIM) [29], multiple-play MAB (MP-MAB) [17], minimum spanning tree (MST), and so forth.

TS-type algorithms have recently attracted a lot of interest in CMAB problems [17, 30, 14, 24]. All of these works need an exact oracle to provide the optimal solution with sampled parameters as input in each round. However, such oracles are usually not feasible since many offline combinatorial optimization problems, such as the offline problem of PMC and OIM [16], are NP-hard and only approximation oracles are available. With an example [30] illustrating the non-convergent regret of TS with an artificial approximation oracle designed for specific problem instance, whether TS could work well in CMAB problems with common approximation oracles is still an open problem.

One of the most common oracles for offline combinatorial optimization problems with a theoretical guarantee is the greedy algorithm. It sequentially finds the current optimal arm, or the optimal collection of multiple arms according to the structural correlations, to maximize the current total expected reward. When the termination condition is met, it will return the set of all arms found in previous steps as the solution. The termination condition is usually formulated by a constraint on the number of steps. For example, in the PMC, OIM, and MP-MAB problems, such a process is limited to continue K steps. The greedy algorithm could provide approximate solutions for offline problems of PMC [8] and OIM [16], and exact optimal solutions for offline problems of MP-MAB [17] and MST [18]. In general, as long as the expected reward in a problem satisfies the monotonicity and submodularity on the action set, the greedy algorithm could serve as an offline oracle to provide approximate solutions [23].

In this paper, we first formulate the CMAB problems with greedy oracle, which is general enough and could cover PMC, OIM, and many other problems. In this framework, the objective of the learning agent is to minimize the cumulative greedy regret, defined as the cumulative difference between the expected reward of the selected action and the expected reward of the greedy's solution in the real environment. Focusing on a specific PMC problem instance, we derive the hardness analysis of the TS algorithm with the greedy oracle to solve CMAB problems. Due to the mismatch between the estimation gaps that need to be eliminated by exploration and the actual regret the algorithm needs to pay for each such exploration, the TS algorithm with greedy oracle in this CMAB problem could not achieve as good theoretical performance as previous MAB algorithms. A problem-dependent regret lower bound of order $\Omega(\log T/\Delta^2)$ is provided to illustrate such hardness, where T is the time horizon and Δ is some reward gap. By carefully exploiting the property of the greedy oracle, we also provide an almost matching problem-dependent regret upper bound, which is tight on that PMC problem instance and also recovers the main order of TS when solving MAB problems [3]. These results are the first theoretical results of TS for CMAB problems with approximation oracle, which also indicates the hardness of the convergence of TS with other common approximation oracles.

2 Related Work

CMAB problems have been widely studied in the literature [11, 6, 7, 29, 30, 24, 32, 21, 14]. Here we mainly focus on the most relevant works. Gai *et al.* [11] first study the CMAB with linear reward, where the reward of an action is linear in the reward of base arms included in it. They introduce a

UCB-type algorithm to solve such problems and allow approximation algorithms to serve as offline oracles. Later, Chen *et al.* [6, 7, 29] generalize this framework by considering a larger class of rewards and the case with probabilistically triggered arms (CMAB-T). This framework only assumes the expected reward satisfies the monotonicity and Lipschitz condition. The combinatorial UCB (CUCB) algorithm is proposed to solve such general CMAB problems, which could work with any offline oracle with approximation guarantees. When only approximation oracles are available, the goal of the algorithm is relaxed to minimize the cumulative approximation regret, which is defined as the difference between the expected reward of the selected action and the scaled optimal solution. The CUCB algorithm achieves the regret upper bound of order $O(\log T/\Delta_{\min})$ [29], where $\Delta_{\min}$ is the minimum reward gap from the scaled optimal solution over all suboptimal actions.

TS-type algorithms have recently attracted a lot of interest in CMAB problems. Komiyama *et al.* [17] consider using the TS algorithm to solve the MP-MAB problem, where the agent needs to select K from m arms to maximize the sum of rewards over these K arms. They provide the optimal regret upper bound of order $O(\log T/\Delta_{K,K+1})$ for the TS algorithm to solve this problem, where $\Delta_{K,K+1}$ is the reward gap between the K-th and $(K+1)$-th optimal arm. Later, Wang *et al.* [30] consider using TS to solve more general CMAB problems where only the Lipschitz condition is assumed to be satisfied. Their regret upper bound of order $O(\log T/\Delta_{\min})$ matches the main order of the CUCB [29] in the same setting. The coefficient of this upper bound was recently improved by Perrault *et al.* [24], who study the same CMAB setting. Huyuk *et al.* [14] extend the analysis of [30] and consider using the TS algorithm to solve the problem of CMAB-T. However, the current regret upper bound is $O(1/p^*)$ worse than CUCB [29], where p^* is the minimum triggering probability.

Compared with UCB-type algorithms, all of the above TS-based works need an exact oracle to provide the optimal solution with sampled parameters in each round. However, the exact oracles are usually not feasible as many combinatorial optimization problems, such as the offline problem of OIM and PMC, are NP-hard [16]. Wang *et al.* [30] have constructed a problem instance and designed an approximation offline oracle for this problem instance. The analysis has shown that the TS algorithm suffers the linear regret of order $O(T)$ when working with such an approximation oracle. However, this oracle is uncommon and artificial, thus could not represent the performance of TS when working with common approximation oracles. It is still a significant problem to be answered that whether TS would perform well with approximation oracles.

The greedy algorithm is one of the most important methods with approximation guarantees to solve combinatorial optimization problems. When the parameters are known beforehand, we call it *offline* problems. Using greedy algorithm to solve offline combinatorial problems have been studied for decades, including the problem of shortest spanning subtree [18], shortest connection networks [25], set coverage [8], influence maximization [16], and general submodular optimization [23]. There is also a line of works consider using greedy to solve online problems [4, 19, 10, 27, 22], some of them require the exact reward function forms as prior knowledge. Among these works the most related to ours is [22], both aiming to solve a general class of online problems. Lin *et al.* [22] consider using the online greedy strategy to make decisions based on UCB-type estimators. The algorithm sequentially selects a unit of arms to maximize the current expected reward until no feasible unit could be selected in each round. In their setting, a unit conditioned on a set of previously selected units is regarded as an arm and the marginal reward of selecting this unit is the expected reward of this arm. Since the number of combinations of units is usually exponentially large, there is an exponential number of arms to explore, making the algorithm pay exponential memory cost in practical applications. Based on such a framework, the algorithm needs to observe the marginal reward after the decision of each step to update the estimate on the arm. However, such observation may be not available as many combinatorial problems treat the action composed of several units as a whole and select them together. Compared to this work, our framework only needs polynomial memory cost and does not require the observation of the marginal reward.

In this paper, we study the problem of CMAB with the common (approximation) greedy oracle and hope to answer the question of whether the TS algorithm could work well in this problem.

3 Setting

The combinatorial multi-armed bandit (CMAB) problem is formulated by a T-round learning game between the learning agent and the environment. The environment contains m base arms and the

arm set is denoted by $[m] = \{1, 2, \ldots, m\}$. Each arm $i \in [m]$ is associated with a distribution D_i on $[0, 1]$. We consider a combinatorial setting where the agent could select several base arms at a time. In many applications, different arms usually have structural correlations in the selection decision of the agent. For example, in the PMC and OIM problems, base arms (edges) starting from the same node must be selected together. Thus the base arm set $[m]$ can be further divided into n units, with each unit containing several base arms and a unit of arms will be selected together. Let $\mathcal{U}$ be the collection of all units and $|s|$ be the number of base arms contained in unit s for any $s \in \mathcal{U}$.

In each round $t = 1, 2, \ldots$, the learning agent selects an action $S_t \in \mathcal{S} = \{S \subseteq \mathcal{U} : |S| = K\}$ to play. Here $\mathcal{S}$ is the set of all candidate actions containing K units. For any action S, denote $\cup S = \{i \in s \text{ for some } s \in S\}$ as the set of base arms that belong to the units contained in S. The environment then draws a random output of all base arms $X_t = (X_{t,1}, X_{t,2}, \ldots, X_{t,m})$ from the distribution $D = D_1 \times D_2 \times \ldots \times D_m$. For any t, $X_{t,i}$ is independent and identically distributed on D_i with expectation μ_i, for any base arm i. Let $\mu = (\mu_i)_{i \in [m]}$ be the mean vector. We study the semi-bandit feedback [30, 7, 29] where the agent could observe feedback $Q_t = \{(i, X_{t,i}) \mid i \in \cup S_t\}$, namely the output of all base arms in units contained in S_t. Denote $\mathcal{H}_t = \{(S_\tau, Q_\tau) : 1 \leq \tau < t\}$ as the history of observations at time t. The agent finally obtains a corresponding reward $R_t = R(S_t, X_t)$ in this round, which is a function of action S_t and output X_t. We assume the expected reward satisfies the following two assumptions, which are standard in CMAB works [7, 30, 14, 24, 29].

Assumption 1. *The expected reward of an action S only depends on the S and the mean vector μ. That is to say, there exists a function r such that $\mathbb{E}[R_t] = \mathbb{E}_{X_t \sim D}[R(S_t, X_t)] = r(S_t, \mu)$.*

Assumption 2. *(Lipschitz continuity) There exists a constant B such that for any action S and mean vectors μ, μ', the reward of S under μ and μ' satisfies*

$$|r(S, \mu) - r(S, \mu')| \leq B \sum_{i \in \cup S} |\mu_i - \mu'_i| \ . \tag{1}$$

The above framework could cover many problems including OIM [29, 32, 21] on two-layer graphs, PMC, and MP-MAB [17] problems. In the OIM problem under independent cascade (IC) [29, 32] and linear threshold (LT) [21] models, given a two-layer graph $G = (V_1, V_2, E)$, where V_1 is the node set in the first layer, V_2 is the node set in the second layer and all edges have the start node in V_1 and end node in V_2. The candidate actions are node sets with size K in V_1. In this case, the base arm set $[m]$ is the edge set E and each unit contains outgoing edges from a single node. Thus the size of the unit set is $|\mathcal{U}| = n = |V_1|$. The expected reward of an action in this problem is the expected number of nodes finally influenced by it, which satisfies the above assumptions under both the IC [7] and LT [21] models. The PMC problem could be regarded as an OIM instance under the IC model on a two-layer graph thus could also be modeled by this framework. In the MP-MAB problem, given base arm set $[m]$, the action set is defined as the set of actions containing K base arms. To model the MP-MAB problem, each unit $s \in \mathcal{U}$ only contains a single base arm and the unit set is $\mathcal{U} = \{\{i\} : i \in [m]\}$. The expected reward of an action is defined as the sum of the mean values of all base arms contained in it, thus also satisfies the above assumptions.

Algorithm 1 `Greedy` algorithm

Input: base arm set $[m]$ and mean vector $\mu = (\mu_i)_{i \in [m]}$, unit set $\mathcal{U}$, action size K
Initialize: $S_g = \emptyset$
for $k = 1, 2, \cdots, K$ **do**
  $s_k = \text{argmax}_{s \in \mathcal{U} \setminus S_g} r(S_g \cup \{s\}, \mu)$
  $S_g = S_g \cup \{s_k\}$
end for
Output: S_g

When the mean vector of base arms is known beforehand, finding the optimal action containing K units is called the *offline* problem. However, the offline problems are usually NP-hard, and enumerating all actions to find the best one is not feasible as the number of actions is exponentially large. The `Greedy` algorithm (presented in Algorithm 1) is a common method to solve such offline problems, which is simple to implement and could provide approximate solutions for OIM [16] and PMC [8], and exact solutions for MP-MAB[17]. More specifically, when the reward function satisfies monotonicity and submodularity on the action set, the `Greedy` algorithm is a common method with

approximate guarantees [23]. Moreover, the greedy algorithm is also popular to serve as a heuristic method in real applications and could have good performance even without a theoretical guarantee.

We mainly study the CMAB problems with the `Greedy` oracle. With input $\mu = (\mu_i)_{i\in[m]}$, it sequentially selects K units to maximize the current expected reward. To simplify, we assume the `Greedy`'s solution $S_g(\mu)$, abbreviated as S_g, is unique, or equivalently the optimal unit in each step k (Line 4 in Algorithm 1) is unique. The general case with multiple solutions could also be solved and would be discussed later. The objective of the learning agent is to maximize the cumulative expected reward over T rounds, or equivalently to minimizing the cumulative expected regret with respect to the `Greedy`'s solution S_g, which we call cumulative *greedy regret* [22] defined by

$$R_g(T) = \mathbb{E}\left[\sum_{t=1}^{T} \max\left\{r(S_g,\mu) - r(S_t,\mu), 0\right\}\right], \tag{2}$$

where the expectation is taken from the randomness in observations and the online algorithm.

When `Greedy` is an α-approximation oracle, which means $r(S_g(\mu'),\mu') \geq \alpha \cdot r(S^*(\mu'),\mu')$ for any input μ', where $S^*(\mu')$ is the optimal action under μ'. In this case, the upper bound for the greedy regret also implies the upper bound for the α-approximation regret defined by the scaled optimal reward $\alpha \cdot r(S^*(\mu),\mu)$. The approximation regret is adopted in previous CMAB works based on UCB-type algorithms [7, 29, 32, 21], and is much weaker than greedy regret as it relaxes the requirements for online algorithms and only needs them to return solutions satisfying the relaxed approximation ratio.

4 Algorithm

In this section, we introduce the combinatorial Thompson sampling (CTS) algorithm with `Greedy` oracle (presented in Algorithm 2) for CMAB problems.

The CTS algorithm maintains a Beta distribution with parameters a_i and b_i for each base arm $i \in [m]$. At the beginning, it initializes $a_i = b_i = 1, \forall i \in [m]$ (Line 2). In each round t, the algorithm first samples a parameter candidate $\theta_{t,i}$ from Beta(a_i, b_i) representing the current estimate for μ_i (Line 4). Then the `Greedy` oracle outputs the solution S_t according to the input vector $\theta_t = (\theta_{t,1}, \theta_{t,2}, \cdots, \theta_{t,m})$ (Line 5). Based on the observation feedback, the algorithm then updates the corresponding Beta distributions for observed base arms (Line 7-10).

Algorithm 2 CTS algorithm with `Greedy` oracle

1: Input: base arm set $[m]$, unit set $\mathcal{U}$, action size K
2: Initialize: $\forall i \in [m], a_i = b_i = 1$
3: **for** $t = 1, 2, \cdots$ **do**
4: $\quad\forall i \in [m]$: Sample $\theta_{t,i} \sim$Beta(a_i, b_i). Denote $\theta_t = (\theta_{t,1}, \theta_{t,2}, \cdots, \theta_{t,m})$
5: $\quad$Select action $S_t = \texttt{Greedy}([m], \theta_t, \mathcal{U}, K)$ and receive the observation Q_t
6: $\quad$//Update
7: $\quad$**for** $(i, X_{t,i}) \in Q_t$ **do**
8: $\quad\quad$With probability $X_{t,i}$, $Y_{t,i} = 1$; with probability $1 - X_{t,i}$, $Y_{t,i} = 0$
9: $\quad\quad$Update $a_i = a_i + Y_{t,i}, b_i = b_i + (1 - Y_{t,i})$
10: $\quad$**end for**
11: **end for**

5 Lower Bound

We investigate the hardness of the CTS to solve CMAB problems with `Greedy` oracle by proving a problem-dependent regret lower bound.

First, we introduce some notations that will be used in the regret analysis. Recall S_g is the solution returned by the `Greedy` oracle when the input is μ. We denote it as $S_g = \{s_{g,1}, s_{g,2}, \ldots, s_{g,K}\}$, where $s_{g,k}$ is the k-th selected unit by `Greedy`. Further, define $S_{g,k} = \{s_{g,1}, s_{g,2}, \ldots, s_{g,k}\}$ as the sequence containing the first k units for any $k \in [K]$. Similarly, let $S_t = \{s_{t,1}, s_{t,2}, \ldots, s_{t,K}\}$ and

$S_{t,k} = \{s_{t,1}, s_{t,2}, \ldots, s_{t,k}\}$. Note $S_{g,0} = S_{t,0} = \emptyset$. The corresponding gaps are defined to measure the hardness of the task and the performance of the algorithm.

Definition 1. *(Gaps) For any unit $s \in \mathcal{U}$ and index $k \in [K]$ such that $s \notin S_{g,k-1}$, define the marginal reward gap*

$$\Delta_{s,k} = r(S_{g,k}, \mu) - r(S_{g,k-1} \cup \{s\}, \mu)$$

as the reward difference between $S_{g,k}$ and $S_{g,k-1} \cup \{s\}$. According to the design of the `Greedy` *algorithm, we have $\Delta_{s,k} > 0$ for any k such that $s \notin S_{g,k}$. And for any action $S \in \mathcal{S}$, define $\Delta_S = \max\{r(S_g, \mu) - r(S, \mu), 0\}$ as the reward difference from the* `Greedy`*'s solution S_g. Let*

$$\Delta_s^{\min} = \min_{S \in \mathcal{S}: s \in S} \Delta_S\,, \quad \Delta_s^{\max} = \max_{S \in \mathcal{S}: s \in S} \Delta_S$$

be the minimum and maximum reward gap of actions containing unit s, respectively. Denote $\Delta_{\max} = \max_{S \in \mathcal{S}} \Delta_S$ as the maximum reward gap over all suboptimal actions.

We take the following CMAB problem (shown in Figure 1) as the instance to carry out the hardness analysis. Each edge in the graph is a base arm and the set of all outgoing edges from a single node forms a unit. The action size is set to $K = 2$. The mean outcome $\mu_{(u,v)}$ of each edge (u, v) is listed on the edges, where we assume $0 < \Delta \leq 0.04$. The expected reward of an action is computed as in the problem of PMC or the problem of OIM under the IC model [32, 29]. Specifically, after an action S is selected, each node u contained in S will independently make an attempt to influence each v of its outgoing neighbors for once with success probability $\mu_{(u,v)}$. When a node receives multiple influence attempts, it would be finally influenced if one of these attempts succeeds. For example, when u_1 and u_2 try to influence v_1, the probability that v_1 is influenced is $1 - (1 - \mu_{(u_1,v_1)})(1 - \mu_{(u_2,v_1)}) = 0.592$. The expected reward of an action is defined as the expected number of nodes it could influence. Take action $S = \{u_1, u_2\}$ as an example, the expected number of nodes S could influence is $r(S, \mu) = 0.592 + 0.3 = 0.892$. It is proved that such a problem satisfies the Lipschitz condition [7] and the `Greedy` oracle could provide an approximate solution with approximation ratio $(1 - \frac{1}{e})$ for any input [16]. For simplicity, we also assume the output of each base arm in each round is exactly its mean. Such assumption still satisfies the above properties and is also adopted in previous lower bound proofs [3] to simplify the analysis.

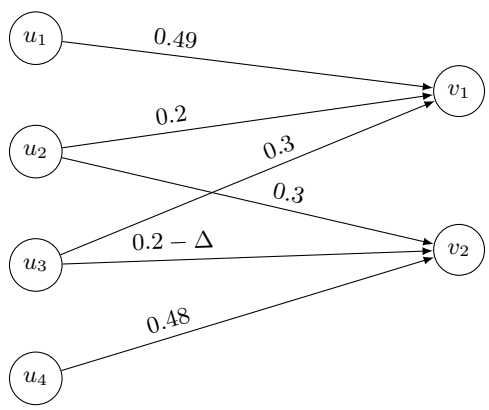

Figure 1: The underlying graph of the CMAB instance used to derive the hardness analysis.

For convenience, we first list the expected reward of each action in this problem in the following Table 1. We can find that the greedy solution is $S_g = \{u_2, u_1\}$ with $s_{g,1} = u_2$, $s_{g,2} = u_1$, and $r(S_g, \mu) = 0.892$, while the optimal action is $\{u_1, u_4\}$. The corresponding marginal reward gaps of each unit can be then computed as follows.

$$\begin{aligned}
&\Delta_{u_1,1} = 0.01\,; \\
&\Delta_{u_3,1} = \Delta, \qquad \Delta_{u_3,2} = 0.012 + 0.7\Delta\,; \\
&\Delta_{u_4,1} = 0.02, \quad \Delta_{u_4,2} = 0.056\,.
\end{aligned}$$

In the following, we mainly focus on unit u_3 and take it as an example to derive the hardness analysis. According to Table 1, all actions containing u_3 are suboptimal actions compared to S_g and $\Delta_{u_3}^{\min} =$

Action	Expected Reward	Action	Expected Reward
$\{u_1\}$	0.49	$\{u_1, u_2\}$	0.892
$\{u_2\}$	0.5	$\{u_1, u_3\}$	$0.843 - \Delta$
$\{u_3\}$	$0.5 - \Delta$	$\{u_1, u_4\}$	0.97
$\{u_4\}$	0.48	$\{u_2, u_3\}$	$0.88 - 0.7\Delta$
$\{u_3, u_4\}$	$0.884 - 0.52\Delta$	$\{u_2, u_4\}$	0.836

Table 1: The expected rewards of actions in the problem instance shown in Figure 1.

$\Delta_{\{u_3,u_4\}} = 0.52\Delta + 0.008$. Thus to avoid regret generation, the algorithm should avoid incorporating u_3 in the action S_t. Intuitively, u_3 should be explored at least $\Omega\left(\log T/\Delta^2_{u_3,1}\right) = \Omega\left(\log T/\Delta^2\right)$ times to be distinguished from $s_{g,1} = u_2$ and thus could avoid being selected as the first unit by `Greedy`. However, in each round of such exploration, the algorithm needs to pay a constant regret of at least $0.52\Delta + 0.008$. Thus the estimation gap Δ needs to be eliminated by exploration on the denominator of $\Omega\left(\log T/\Delta^2\right)$ could not be canceled by the actual regret paid in each exploration round. Such mismatch would cause the greedy regret at least of order $\Omega\left(\log T/\Delta^2\right)$.

We give the formal lower bound for both the expected number of selections of each unit and the cumulative greedy regret in the following Theorem 1.

Theorem 1. *(Lower bound) Using the CTS algorithm with Gaussian priors and* `Greedy` *oracle to solve the CMAB problem shown in Figure 1, when T is sufficiently large, we have*

$$\mathbb{E}\left[N_{T+1,s}\right] = \Omega\left(\frac{\log T}{\Delta^2_{s,1}}\right), \tag{3}$$

for any $s \neq s_{g,1} = u_2$, where $N_{T+1,s} = \sum_{t=1}^{T} \mathbb{1}\{s \in S_t\}$ is the number of rounds when s is contained in the selected action set S_t.

Further, the cumulative greedy regret satisfies

$$R_g(T) = \Omega\left(\frac{\log T}{\Delta^2_{u_3,1}}\right) = \Omega\left(\frac{\log T}{\Delta^2}\right). \tag{4}$$

The proof of Theorem 1 follows directly the intuition of the above hardness analysis. The reason why we consider the Gaussian priors to derive the lower bound analysis is that we hope to use its concentration and anti-concentration inequalities. The analysis could directly apply to other types of prior distribution if similar inequalities could be provided. Due to the space limit, we include the detailed proof in Appendix B.

6 Upper Bound

By investigating the properties of the CTS algorithm and the `Greedy` oracle, we also provide a problem-dependent regret upper bound for Algorithm 2 to solve general CMAB problems.

Theorem 2. *(Upper bound) The cumulative greedy regret of the CTS algorithm with* `Greedy` *oracle can be upper bounded by*

$$\begin{aligned} R_g(T) \leq & \sum_{s \neq s_{g,1}} \max_{k: s \notin S_{g,k}} \frac{6B^2 |s|^2 \Delta_s^{\max} \log T}{\left(\Delta_{s,k} - 2B |\cup S_g| \varepsilon\right)^2} + \sum_{k \in [K]} \frac{C}{\varepsilon^2} \left(\frac{C'}{\varepsilon^4}\right)^{|s_{g,k}|} \Delta_{\max} \\ & + \left(|\cup S_g| \left(2 + \frac{8}{\varepsilon^2}\right) + 4m\right) \Delta_{\max} \qquad (5) \\ = & O\left(\sum_{s \neq s_{g,1}} \max_{k: s \notin S_{g,k}} \frac{B^2 |s|^2 \Delta_{\max} \log T}{\Delta^2_{s,k}}\right), \qquad (6) \end{aligned}$$

for any ε such that $\forall s \neq s_{g,1}$ and k satisfying $s \notin S_{g,k}$, $\Delta_{s,k} > 2B |\cup S_g| \varepsilon$, where B is the coefficient of the Lipschitz continuity condition, $|\cup S_g|$ is the number of base arms that belong to the units contained in S_g, C and C' are two universal constants.

Tightness of the upper bound. We now discuss the tightness of the above regret upper bound based on the problem instance shown in Figure 1. Specific to this problem, we have $|s| \leq 2$ for all s since each node has no more than 2 outgoing edges. And with the help of [21, Theorem 4], it could be easily checked that the coefficient of the Lipschitz condition in this problem is $B = 1$. When $0 < \Delta \leq 0.04$, we have $\Delta_{s,1} = \min_{k:s \notin S_{g,k}} \Delta_{s,k}$ for any $s \neq s_{g,1} = u_2$, and $\Delta_{\max} = \Delta_{\{u_2,u_4\}} = 0.056$ is a constant. Thus our regret upper bound in this problem instance is of order

$$O\left(\sum_{s \neq s_{g,1}} \max_{k:s \notin S_{g,k}} \frac{B^2 |s|^2 \Delta_{\max} \log T}{\Delta_{s,k}^2}\right) = O\left(\sum_{s \in \{u_1,u_3,u_4\}} \frac{\log T}{\Delta_{s,1}^2}\right) = O\left(\frac{\log T}{\Delta^2}\right),$$

where the last equality holds since $\Delta_{u_1,1} = 0.01, \Delta_{u_4,1} = 0.02$ are constants and $\Delta_{u_3,1} = \Delta$.

We can see our regret upper bound matches the lower bound of (4) in Theorem 1 only up to some constant factors in this specific problem instance.

Comparison with MAB. When each unit contains only one base arm and the action size is $K = 1$, our CMAB framework could model the MAB problem and the `Greedy` oracle could provide the exact optimal solution. Thus we can also compare our regret upper bound with the theoretical results of the TS algorithm in MAB problems. In the MAB problem, the expected reward of each action is exactly the mean of the base arm contained in this action. Thus the Lipschitz coefficient is just $B = 1$ and $|s| = 1$ for any unit s. The optimal action is $S_g = S_{g,1}$ with $|\cup S_g| = 1$. And for any unit $s \neq s_{g,1}$, we have $\Delta_s^{\max} = \Delta_{s,1}$. Thus, according to (5) of Theorem 2, the regret upper bound of Algorithm 2 in MAB problems is of order $O\left(\sum_{s \neq s_{g,1}} \frac{\log T}{\Delta_{s,1}}\right)$, which recovers the main order of the regret upper bound of TS for MAB problems [3].

Discussion on the marginal reward gap. Recall that the `Greedy` oracle could provide approximate solutions for problems whose expected reward satisfies monotonicity and submodularity on the action set. Formally, the submodularity means for any action $S \subseteq T$ and unit $s \notin T$, there is $r(S \cup \{s\}, \mu) - r(S, \mu) \geq r(T \cup \{s\}, \mu) - r(T, \mu)$, which characterizes the phenomenon of diminishing returns. One may concern that in these problems, due to the submodularity, the marginal reward gap $\Delta_{s,k}$ for larger k would become much smaller and the main order of the upper bound would blow up. We would clarify that the submodularity could not imply the relationships among marginal reward gaps $\Delta_{s,k}$ for different $k \in [K]$. The problem instance in Figure 1 satisfying submodularity [16] indicates that the marginal reward gap $\Delta_{s,k}$ does not necessarily decrease with the increase of k.

6.1 Proof Sketch

In this section, we briefly introduce the proof idea of Theorem 2. At any step $k \in [K]$, for any unit $s \notin S_{g,k}$, the selection of s in action S_t may force the `Greedy` oracle to choose a worse action in subsequential steps and making CTS suffer constant regret. A sufficient condition for generating zero regret in round t is that each unit $s_{t,k}$ selected in step k is actually $s_{g,k}$ for any $k \in [K]$. Thus to bound the cumulative greedy regret, we could sequentially analyze whether $s_{g,k}$ is appropriately selected at each step k.

Recall that under the framework of the CTS algorithm, the `Greedy` oracle sequentially selects $s_{t,k}$ for $k \in [K]$ based on θ_t sampled from the prior distributions in round t. Focusing on step $k = 1$, to ensure $s_{t,1} = s_{g,1}$, the algorithm needs to guarantee the accurate estimations $\theta_{t,s}$ for any unit $s \in \mathcal{U}$ such that $r(\{s\}, \theta_t) < r(S_{g,1}, \theta_t), \forall s \neq s_{g,1}$, where $\theta_{t,s}$ is the projection of θ_t on unit s. We could first suppose $s_{g,1}$ is already estimated well. Then based on this assumption, when all of the other units $s \neq s_{g,1}$ have been explored $O(\log T/\Delta_{s,1}^2)$ times and thus estimated accurately, $s_{g,1}$ would be selected at the first step with high probability. But if after any other unit $s \neq s_{g,1}$ has already been estimated well, $s_{g,1}$ is still not selected appropriately, we can conclude that the estimations for $s_{g,1}$ are not accurate enough. In this case, the Beta priors for $s_{g,1}$ tend to be uniformly distributed. When CTS sample $\theta_{t,s_{g,1}}$ from its Beta priors, with constant probability there would be $r(S_{g,1}, \theta_t) > r(\{s\}, \mu) \approx r(\{s\}, \theta_t)$ for any unit $s \neq s_{g,1}$. Thus after some rounds, $s_{g,1}$ would be selected enough times and also estimated accurately. Then in the following rounds, $s_{g,1}$ would be selected appropriately at the first step with high probability. Above all, the expected number of misselections at the first step could be bounded.

The above analysis can apply to other cases when $k = 2, 3, \ldots, K$. Based on the correct selections in the first $k-1$ steps, the misselection of $s_{t,k}$ also comes from the bad estimations for both $s_{g,k}$ and other units $s \notin S_{g,k}$. To distinguish $r(S_{g,k-1} \cup \{s\}, \theta_t)$ from $r(S_{g,k}, \theta_t)$, those units need to be explored at least $O(\log T/\Delta_{s,k}^2)$ times.

According to the above analysis, for each unit $s \neq s_{g,1}$, we define the exploration price as

$$L(s) = O\left(\max_{k: s \notin S_{g,k}} \frac{\log T}{\Delta_{s,k}^2}\right). \tag{7}$$

To avoid being incorrectly selected at some step k, each unit $s \notin S_{g,k}$ needs to be explored for at least $L(s)$ times. The sum of $L(s)$ over all units $s \neq s_{g,1}$ would lead to the main order of the regret upper bound in Theorem 2. Due to the space limit, the formal proof is presented in Appendix C.

6.2 Extension to Multiple-solution Case

We can also extend the analysis of the regret upper bound to the case where multiple solutions could be returned by `Greedy` with input μ, or equivalently the optimal unit in each step k (Line 4 in Algorithm 1) may be not unique. Let

$$\sigma_K = \left\{\{s_1, s_2, \ldots, s_K\} : s_1 \in \operatorname*{argmax}_s r(\{s\}, \mu), \ldots, s_K \in \operatorname*{argmax}_{s \notin \{s_1, \ldots, s_{K-1}\}} r(\{s_1, \ldots, s\}, \mu)\right\}$$

be the set of all actions that could be possibly returned by the `Greedy` oracle when the input is μ. Here we do not care how `Greedy` breaks the tie at each step and regard this process as a black box. In order to take into account the worst case where `Greedy` always return a solution with minimum reward compared to other possible solutions, we define $S_g \in \operatorname{argmin}_{S \in \sigma_K} r(S, \mu)$ as one of `Greedy`'s possible solutions with the minimum expected reward and consider the cumulative greedy regret defined in (2).

The proof idea of the regret upper bound of Algorithm 2 in this case is similar to the proof of Theorem 2. A sufficient condition for generating zero regret in round t is that the selected action S_t falls into the set σ_K. Thus to bound the cumulative greedy regret, we could sequentially analyze whether the unit $s_{t,k}$ selected in each step k is an optimal unit conditioned on the previously selected units $S_{t,k-1}$. For completeness, we include the regret upper bound as well as the detailed proof for this case in Appendix D.

7 Conclusion

In this paper, we aim to answer the question of whether the convergence analysis of TS could extend beyond exact oracles in the CMAB area. Taking the common offline (approximation) `Greedy` oracle as an example, we derive the hardness analysis of CTS for CMAB problems based on a constructed CMAB problem instance. When using CTS with `Greedy` oracle to solve this problem, we find that the algorithm needs to explore at least $\Omega(\log T/\Delta^2)$ rounds to distinguish the suboptimal units from the optimal unit at some step. However, at least constant regret needs to pay for each such exploration round. The mismatch between the gap to be distinguished and the actually paid regret forces the algorithm to pay the cumulative regret of order $\Omega(\log T/\Delta^2)$. We also provide an almost matching problem-dependent regret upper bound for the CTS algorithm with `Greedy` oracle to solve CMAB problems. The upper bound is tight on the constructed problem instance only up to some constant factors and also recovers the main order of TS when solving MAB problems.

An interesting future direction is to extend the current CMAB framework to the case with probabilistically triggered arms (CMAB-T). The CMAB-T framework could model the OIM problem on general social networks. As shown in [21], using UCB-type algorithms to solve such a problem may face great challenges on the computation efficiency. This problem is expected to be avoided by TS-type algorithms since TS would sample candidate parameters to escape the computation of complicated optimization problems. However, the current proof idea based on each selection step of the `Greedy` oracle (proof of Lemma 1) could not directly apply to this setting as different units may probabilistically trigger some common base arms. New proof techniques are required to derive the theoretical guarantee of the TS algorithm with the `Greedy` oracle in this framework.

References

[1] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. 1964.

[2] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Proceedings of the 25th Annual Conference on Learning Theory*, pages 1–26, 2012.

[3] Shipra Agrawal and Navin Goyal. Further optimal regret bounds for thompson sampling. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, pages 99–107, 2013.

[4] Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. Minimax policies for combinatorial prediction games. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 107–132, 2011.

[5] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[6] Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In *Proceedings of the 30th International Conference on Machine Learning*, pages 151–159, 2013.

[7] Wei Chen, Yajun Wang, Yang Yuan, and Qinshi Wang. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *The Journal of Machine Learning Research*, pages 1746–1778, 2016.

[8] Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[9] Richard Combes, Sadegh Talebi, Alexandre Proutière, and Marc Lelarge. Combinatorial bandits revisited. In *Advances in Neural Information Processing Systems*, pages 2116–2124, 2015.

[10] Victor Gabillon, Branislav Kveton, Zheng Wen, Brian Eriksson, and S Muthukrishnan. Adaptive submodular maximization in bandit setting. In *Advances in Neural Information Processing Systems*, pages 2697–2705, 2013.

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

[12] Dorota Głowacka. *Bandit algorithms in information retrieval*. Now Publishers, 2019.

[13] Dorota Głowacka. Bandit algorithms in recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 574–575, 2019.

[14] Alihan Huyuk and Cem Tekin. Analysis of thompson sampling for combinatorial multi-armed bandit with probabilistically triggered arms. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, pages 1322–1330, 2019.

[15] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: an asymptotically optimal finite-time analysis. In *Proceedings of the 23rd international conference on Algorithmic Learning Theory*, pages 199–213, 2012.

[16] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.

[17] Junpei Komiyama, Junya Honda, and Hiroshi Nakagawa. Optimal regret analysis of thompson sampling in stochastic multi-armed bandit problem with multiple plays. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, pages 1152–1161, 2015.

[18] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[19] Branislav Kveton, Zheng Wen, Azin Ashkan, Hoda Eydgahi, and Brian Eriksson. Matroid bandits: fast combinatorial optimization with learning. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, pages 420–429, 2014.

[20] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[21] Shuai Li, Fang Kong, Kejie Tang, Qizhi Li, and Wei Chen. Online influence maximization under linear threshold model. In *Advances in Neural Information Processing Systems*, 2020.

[22] Tian Lin, Jian Li, and Wei Chen. Stochastic online greedy learning with semi-bandit feedbacks. In *Advances in Neural Information Processing Systems*, pages 352–360, 2015.

[23] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[24] Pierre Perrault, Etienne Boursier, Vianney Perchet, and Michal Valko. Statistical efficiency of thompson sampling for combinatorial semi-bandits. In *Advances in Neural Information Processing Systems*, 2020.

[25] R. C. Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.

[26] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

[27] Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *Advances Neural Information Processing Systems*, pages 1577–1584, 2008.

[28] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[29] Qinshi Wang and Wei Chen. Improving regret bounds for combinatorial semi-bandits with probabilistically triggered arms and its applications. In *Advances in Neural Information Processing Systems*, pages 1161–1171, 2017.

[30] Siwei Wang and Wei Chen. Thompson sampling for combinatorial semi-bandits. In *Proceedings of the 35th International Conference on International Conference on Machine Learning*, pages 5114–5122, 2018. https://arxiv.org/abs/1803.04623.

[31] Zheng Wen, Branislav Kveton, and Azin Ashkan. Efficient learning in large-scale combinatorial semi-bandits. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, pages 1113–1122, 2015.

[32] Zheng Wen, Branislav Kveton, Michal Valko, and Sharan Vaswani. Online influence maximization under independent cascade model with semi-bandit feedback. In *Advances in Neural Information Processing Systems*, pages 1–24, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Please see Abstract and Section 1.
   (b) Did you describe the limitations of your work? [Yes] Please see the future work part in Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This work mainly focuses on the online learning theory, which does not have any potential negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [Yes] Please see Section 3.

(b) Did you include complete proofs of all theoretical results? [Yes] Please see all sections in Appendix.

3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [N/A]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]