Combinatorial Pure Exploration with Bottleneck Reward Function

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this paper, we study the Combinatorial Pure Exploration problem with the Bottleneck reward function (CPE-B) under the fixed-confidence (FC) and fixed-budget (FB) settings. In CPE-B, given a set of base arms and a collection of subsets of base arms (super arms) following a certain combinatorial constraint, a learner sequentially plays a base arm and observes its random reward, with the objective of finding the optimal super arm with the maximum bottleneck value, defined as the minimum expected reward of the base arms contained in the super arm. CPE-B captures a variety of practical scenarios such as network routing in communication networks, and its *unique challenges* fall on how to utilize the bottleneck property to save samples and achieve the statistical optimality. None of the existing CPE studies (most of them assume linear rewards) can be adapted to solve such challenges, and thus we develop brand-new techniques to handle them. For the FC setting, we propose novel algorithms with optimal sample complexity for a broad family of instances and establish a matching lower bound to demonstrate the optimality (within a logarithmic factor). For the FB setting, we design an algorithm which achieves the state-of-the-art error probability guarantee and is the first to run efficiently on fixed-budget path instances, compared to existing CPE algorithms. Our experimental results on the top-k, path and matching instances validate the empirical superiority of the proposed algorithms over their baselines.

1 Introduction

The Multi-Armed Bandit (MAB) problem [30, 35, 4, 2] is a classic model to solve the exploration-exploitation trade-off in online decision making. Pure exploration [3, 24, 6, 31] is an important variant of the MAB problem, which aims to identify the best arm under a given confidence or a given sample budget. There are various works studying pure exploration, such as top-k arm identification [16, 24, 6, 29], top-k arm under matriod constraints [8] and multi-bandit best arm identification [17, 6].

The Combinatorial Pure Exploration (CPE) framework, firstly proposed by Chen et al. [10], encompasses a rich class of pure exploration problems [3, 24, 8]. In CPE, there are a set of base arms, each associated with an unknown reward distribution. A subset of base arms is called a super arm, which follows a certain combinatorial structure. At each timestep, a learner plays a base arm and observes a random reward sampled from its distribution, with the objective to identify the optimal super arm with the maximum expected reward. While Chen et al. [10] provide this general CPE framework, their algorithms and analytical techniques only work under the linear reward function and cannot be applied to other nonlinear reward cases.[1]

[1] The algorithmic designs and analytical tools (e.g., symmetric difference and exchange set) in [10] all rely on the linear property and cannot be applied to nonlinear reward cases, e.g, the bottleneck reward problem.

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

However, in many real-world scenarios, the expected reward function is not necessarily linear. One of the common and important cases is the *bottleneck reward* function, i.e., the expected reward of a super arm is the minimum expected reward of the base arms contained in it. For example, in communication networks [5], the transmission speed of a path is usually determined by the link with the lowest rate, and a learner samples the links in order to find the optimal transmission path which maximizes its bottleneck link rate. In traffic scheduling [37], a scheduling system collects the information of road segments in order to plan an efficient route which optimizes its most congested (bottleneck) road segment. In neural architecture search [38], the overall efficiency of a network architecture is usually constrained by its worst module, and an agent samples the available modules with the objective to identify the best network architecture in combinatorial search space.

In this paper, we study the Combinatorial Pure Exploration with the Bottleneck reward function (CPE-B) which aims to identify the optimal super arm with the maximum bottleneck value by querying the base arm rewards, where the bottleneck value of a super arm is defined as the minimum expected reward of its containing base arms. We consider two popular settings in pure exploration, i.e, *fixed-confidence (FC)*, where given confidence parameter δ, the learner aims to identify the optimal super arm with probability $1-\delta$ and minimize the number of used samples (sample complexity), and *fixed-budget (FB)*, where the learner needs to use a given sample budget to find the optimal super arm and minimize the error probability.

Challenges of CPE-B. Compared to prior CPE works [10, 9, 23], our CPE-B aims at utilizing the bottleneck property to save samples and achieve the statistical optimality. It faces with two *unique challenges*, i.e., how to (i) achieve the tight *base-arm-gap dependent* sample complexity and (ii) avoid the dependence on *unnecessary base arms* in the results, while running in polynomial time. We use a simple example in Figure 1 to illustrate our challenges. In Figure 1, there are six edges (base arms) and three s-t paths (super arms), and the base arm reward $w(e_i)$, base arm gap Δ_{e_i,e_j} and super arm gap $\Delta_{M_*,M_{\text{sub}}}$ are as shown in the figure. In order to identify the optimal path, all we need is to pull e_1, e_2, e_4 to determine that e_1 is worse than e_2 and e_4, and e_3, e_5, e_6 are useless for revealing the sub-optimality of M_1 and M_2. In this case, the optimal sample complexity should be $O((\frac{2}{\Delta^2_{e_2,e_1}} + \frac{1}{\Delta^2_{e_4,e_1}}) \ln \delta^{-1})$, which depends on the tight base arm gaps and only includes the critical base arms (e_1, e_2, e_4). However, if one naively adapts existing CPE algorithms [10, 13, 15] to work with bottleneck reward function, an inferior sample complexity of $O(\sum_{e_i, i\in[6]} \frac{1}{\Delta^2_{M_*,M_{\text{sub}}}} \ln \delta^{-1})$ is incurred, which depends on the loose super arm gaps and contains a summation over all base arms (including unnecessary e_3, e_5, e_6). Hence, our challenge falls on how to achieve such efficient sampling in online environment, where we do not know which are critical base arms e_1, e_2, e_4 but want to gather just enough information to identify the optimal super arm. We remark that, none of existing CPE studies can be applied to solve the unique challenges of CPE-B, and thus we develop brand-new techniques to handle them and attain the optimal results (up to a logarithmic factor).

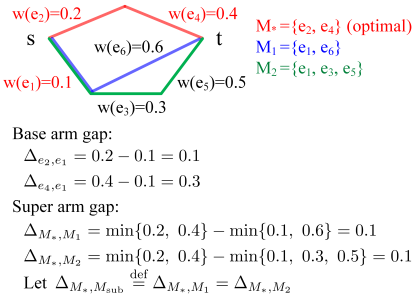


Figure 1: Illustration example.

Contributions. For CPE-B in the FC setting, (i) we first develop a novel algorithm `BLUCB`, which employs a bottleneck-adaptive sample strategy and achieves the tight base-arm-gap dependent sample complexity. (ii) We further propose an improved algorithm `BLUCB-Parallel` in high confidence regime, which adopts an efficient "bottleneck-searching" offline procedure and a novel "check-near-bottleneck" stopping condition. The sample complexity of `BLUCB-Parallel` drops the dependence on unnecessary base arms and achieves the optimality (within a logarithmic factor) under small enough δ. (iii) A matching sample complexity lower bound for the FC setting is also provided, which demonstrates the optimality of our algorithms. For the FB setting, (iv) we propose a novel algorithm `BSAR` with a special acceptance scheme for bottleneck identification task. `BSAR` achieves the state-of-the-art error probability and is the first to run efficiently on fixed-budget path instances, compared to existing CPE algorithms. All our proposed algorithms run in *polynomial time*. The experimental results demonstrate that our algorithms significantly outperform the baselines. Due to space limit, we defer all the proofs to the supplementary material.

1.1 Related Work

In the following we briefly review the related work in the CPE literature. Chen et al. [10] firstly propose the CPE model and only consider the linear reward function (CPE-L), and their results for CPE-L are further improved by [18, 9]. Huang et al. [23] investigate the continuous and separable reward functions (CPE-CS), but their algorithm only runs efficiently on simple cardinality constraint instances. All these works consider directly sampling base arms and getting their feedback. There are also several CPE studies which consider other forms of sampling and feedback. Chen et al. [13] propose the CPE for dueling bandit setting, where at each timestep the learner pulls a duel between two base arms and observes their comparison outcome. Kuroki et al. [28] study an online densest subgraph problem, where the decision is a subgraph and the feedback is the reward sum of the edges in the chosen subgraph (i.e., full-bandit feedback). Du et al. [15] investigate CPE with the full-bandit or partial linear feedback. All of the above studies consider the pure exploration setting, while in combinatorial bandits there are other works [11, 14, 12] studying the regret minimization setting (CMAB). In CMAB, the learner plays a super arm and observes the rewards from all base arms contained in it, with goal of minimizing the regret, which is significantly different from our setting. Note that none of the above studies covers our CPE-B problem or can be adapted to solve the unique challenges of CPE-B, and thus CPE-B demands a new investigation.

2 Problem Formulation

In this section, we give the formal formulation of CPE-B. In this problem, a learner is given n base arms numbered by $1, 2, \ldots, n$. Each base arm $e \in [n]$ is associated with an *unknown* reward distribution with the mean of $w(e)$ and an R-sub-Gaussian tail, which is a standard assumption in bandits [1, 10, 31, 34]. Let $\boldsymbol{w} = (w(1), \ldots, w(n))^\top$ be the expected reward vector of base arms. The learner is also given a decision class $\mathcal{M} \subseteq 2^{[n]}$, which is a collection of super arms (subsets of base arms) and generated from a certain combinatorial structure, such as s-t paths, maximum cardinality matchings, and spanning trees. For each super arm $M \in \mathcal{M}$, we define its expected reward (also called *bottleneck value*) as $\texttt{MinW}(M, \boldsymbol{w}) = \min_{e \in M} w(e)$,[2] i.e., the minimum expected reward of its constituent base arms, which is so called *bottleneck reward function*. Let $M_* = \operatorname{argmax}_{M \in \mathcal{M}} \texttt{MinW}(M, \boldsymbol{w})$ be the optimal super arm with the maximum bottleneck value, and $\texttt{OPT} = \texttt{MinW}(M_*, \boldsymbol{w})$ be the optimal value. Following the pure exploration literature [16, 10, 9, 13], we assume that M_* is unique, and this assumption can be removed in our extension to the PAC learning setting (see the supplementary material).

At each timestep, the learner plays (or samples) a base arm $p_t \in [n]$ and observes a random reward sampled from its reward distribution, where the sample is independent among different timestep t. The learner's objective is to identify the optimal super arm M_* from $\mathcal{M}$.

For this identification task, we study two common metrics in pure exploration [24, 6, 31, 9], i.e., fixed-confidence (FC) and fixed-budget (FB) settings. In the FC setting, given a confidence parameter $\delta \in (0, 1)$, the learner needs to identify M_* with probability at least $1 - \delta$ and minimize the *sample complexity*, i.e., the number of samples used. In the FB setting, the learner is given a fixed sample budget T, and needs to identify M_* within T samples and minimize the *error probability*, i.e., the probability of returning a wrong answer.

3 Algorithms for the Fixed-Confidence Setting

In this section, we first propose a simple algorithm `BLUCB` for the FC setting, which adopts a novel bottleneck-adaptive sample strategy to obtain the tight base-arm-gap dependent sample complexity. We further develop an improvement `BLUCB-Parallel` in high confidence regime, whose sample complexity drops the dependence on unnecessary base arms for small enough δ. Both algorithms achieve the optimal sample complexity for a family of instances (within a logarithmic factor).

3.1 Algorithm `BLUCB` with Base-arm-gap Dependent Results

[2] In general, the second input of function `MinW` can be any vector: for any $M \in \mathcal{M}$ and $\boldsymbol{v} \in \mathbb{R}^n$, $\texttt{MinW}(M, \boldsymbol{v}) = \min_{e \in M} v(e)$.

Algorithm 1 BLUCB, algorithm for CPE-B in the FC setting

1: **Input:** $\mathcal{M}$, $\delta \in (0,1)$ and MaxOracle.
2: Initialize: play each $e \in [n]$ once, and update empirical means $\hat{\boldsymbol{w}}_{n+1}$ and T_{n+1}
3: **for** $t = n+1, n+2, \ldots$ **do**
4: $\quad \mathrm{rad}_t(e) \leftarrow \sqrt{2\ln(\frac{4nt^3}{\delta})/T_t(e)},\ \forall e \in [n]$
5: $\quad \underline{w}_t(e) \leftarrow \hat{w}_t(e) - \mathrm{rad}_t(e),\ \forall e \in [n]$
6: $\quad \bar{w}_t(e) \leftarrow \hat{w}_t(e) + \mathrm{rad}_t(e),\ \forall e \in [n]$
7: $\quad M_t \leftarrow \mathtt{MaxOracle}(\mathcal{M}, \underline{\boldsymbol{w}}_t)$
8: $\quad \tilde{M}_t \leftarrow \mathtt{MaxOracle}(\mathcal{M} \setminus \mathcal{S}(M_t), \bar{\boldsymbol{w}}_t)$
9: $\quad$ **if** $\mathtt{MinW}(M_t, \underline{\boldsymbol{w}}_t) \geq \mathtt{MinW}(\tilde{M}_t, \bar{\boldsymbol{w}}_t)$ **then**
10: $\quad\quad$ **return** M_t
11: $\quad$ **end if**
12: $\quad c_t \leftarrow \mathrm{argmin}_{e \in M_t} \underline{w}_t(e)$
13: $\quad d_t \leftarrow \mathrm{argmin}_{e \in \tilde{M}_t} \underline{w}_t(e)$
14: $\quad p_t \leftarrow \mathrm{argmax}_{e \in \{c_t, d_t\}} \mathrm{rad}_t(e)$
15: $\quad$ Play p_t, and observe the reward
16: $\quad$ Update empirical means $\hat{w}_{t+1}(p_t)$
17: $\quad$ Update the number of samples $T_{t+1}(p_t)$
18: **end for**

Algorithm 1 illustrates the proposed algorithm BLUCB for CPE-B in the FC setting. Here $\mathcal{S}(M_t)$ denote the set of all supersets of super arm M_t (Line 8). Since the bottleneck reward function is monotonically decreasing, for any $M' \in \mathcal{S}(M_t)$, we have $\mathtt{MinW}(M', \boldsymbol{w}) \leq \mathtt{MinW}(M_t, \boldsymbol{w})$. Hence, to verify the optimality of M_t, we only need to compare M_t against super arms in $\mathcal{M} \setminus \mathcal{S}(M_t)$, and this property will also be used in the later algorithm BLUCB-Parallel.

BLUCB is allowed to access an efficient *bottleneck maximization oracle* $\mathtt{MaxOracle}(\mathcal{F}, \boldsymbol{v})$, which returns an optimal super arm from $\mathcal{F}$ with respect to $\boldsymbol{v}$, i.e., $\mathtt{MaxOracle}(\mathcal{F}, \boldsymbol{v}) \in \mathrm{argmax}_{M \in \mathcal{F}} \mathtt{MinW}(M, \boldsymbol{v})$. For $\mathcal{F} = \mathcal{M}$ (Line 7), such an efficient oracle exists for many decision classes, such as the bottleneck shortest path [32], bottleneck bipartite matching [33] and minimum bottleneck spanning tree [7] algorithms. For $\mathcal{F} = \mathcal{M} \setminus \mathcal{S}(M_t)$ (Line 8), we can also efficiently find the best super arm (excluding the supersets of M_t) by repeatedly removing each base arm in M_t and calling the basic maximization oracle, and then selecting the one with the maximum bottleneck value.

We describe the procedure of BLUCB as follows: at each timestep t, we calculate the lower and upper confidence bounds of base arm rewards, denoted by $\underline{\boldsymbol{w}}_t$ and $\bar{\boldsymbol{w}}_t$, respectively. Then, we call MaxOracle to find the super arm M_t with the maximum pessimistic bottleneck value from $\mathcal{M}$ using $\underline{\boldsymbol{w}}_t$ (Line 7), and the super arm $\tilde{M}_t$ with the maximum optimistic bottleneck value from $\mathcal{M} \setminus \mathcal{S}(M_t)$ using $\bar{\boldsymbol{w}}_t$ (Line 8). M_t and $\tilde{M}_t$ are two critical super arms that determine when the algorithm should stop or not. If the pessimistic bottleneck value of M_t is higher than the optimistic bottleneck value of $\tilde{M}_t$ (Line 9), we can determine that M_t has the higher bottleneck value than any other super arm with high confidence, and then the algorithm can stop and output M_t. Otherwise, we select two base arms c_t and d_t with the minimum lower reward confidence bounds in M_t and $\tilde{M}_t$ respectively, and play the one with the larger confidence radius (Lines 12-14).

Bottleneck-adaptive sample strategy. The "select-minimum" sample strategy in Lines 12-14 comes from an *insight* for the bottleneck problem: to determine that M_t has the higher bottleneck value than $\tilde{M}_t$, it suffices to find a base arm from $\tilde{M}_t$ which is worse than any base arm (the bottleneck base arm) in M_t. To achieve this, base arms c_t and d_t, which have the most potential to be the bottlenecks of M_t and $\tilde{M}_t$, are the most necessary ones to be sampled. This bottleneck-adaptive sample strategy is crucial for BLUCB to achieve the tight base-arm-gap dependent sample complexity. In contrast, the sample strategy of prior CPE algorithms [10, 13, 15] treats all base arms in critical super arms (M_t and $\tilde{M}_t$) equally and does a uniform choice. If one naively adapts those algorithms with the current reward function $\mathtt{MinW}(M, \boldsymbol{w})$, a loose super-arm-gap dependent sample complexity is incurred.

To formally state the sample complexity of BLUCB, we introduce some notation and gap definition. Let $N = \{e \mid e \notin M_*, w(e) < \mathtt{OPT}\}$ and $\tilde{N} = \{e \mid e \notin M_*, w(e) \geq \mathtt{OPT}\}$, which stand for the necessary and *unnecessary* base arms contained in the sub-optimal super arms, respectively. We define the reward gap for the FC setting as

Definition 1 (Fixed-confidence Gap)**.**

$$\Delta_e^{\mathrm{C}} = \begin{cases} w(e) - \max_{M \neq M_*} \mathtt{MinW}(M, \boldsymbol{w}), \text{ if } e \in M_*, & \text{(a)} \\ w(e) - \max_{M \in \mathcal{M}: e \in M} \mathtt{MinW}(M, \boldsymbol{w}), \text{ if } e \in \tilde{N}, & \text{(b)} \\ \mathtt{OPT} - \max_{M \in \mathcal{M}: e \in M} \mathtt{MinW}(M, \boldsymbol{w}), \text{ if } e \in N. \end{cases}$$

Now we present the sample complexity upper bound of BLUCB.

Algorithm 2 `BLUCB-Parallel`, an improved algorithm for the FC setting under small δ

Input: $\delta \in (0, 0.01)$ and sub-algorithm `BLUCB-Verify`.
For $k \in \mathbb{N}$, let `BLUCB-Verify`$_k$ be the sub-algorithm `BLUCB-Verify` with $\delta_k = \frac{\delta}{2^{k+1}}$
for $t = 1, 2, \ldots$ **do**
  for each $k \in \mathbb{N}$ such that $t \bmod 2^k = 0$ **do**
    Start or resume `BLUCB-Verify`$_k$ with one sample, and then suspend `BLUCB-Verify`$_k$
    if `BLUCB-Verify`$_k$ returns an answer M_{out}, then **return** M_{out}
  end for
end for

Algorithm 3 `BLUCB-Verify`, sub-algorithm of `BLUCB-Parallel`

Input: $\mathcal{M}$, $\delta^V \in (0, 0.01)$ and `MaxOracle`.
$\kappa \leftarrow 0.01$
$\hat{M}_*, \hat{B}_{\text{sub}} \leftarrow$ `BLUCB-Explore`$(\mathcal{M}, \kappa, \texttt{MaxOracle})$
Initialize: play each $e \in [n]$ once, and update empirical means $\hat{\boldsymbol{w}}_{n+1}$ and T_{n+1}
for $t = n+1, n+2, \ldots$ **do**
  $\text{rad}_t(e) \leftarrow R\sqrt{2\ln(\frac{4nt^3}{\delta^V})/T_t(e)}, \forall e \in [n]$
  $\underline{w}_t(e) \leftarrow \hat{w}_t(e) - \text{rad}_t(e),\ \forall e \in [n]$
  $\bar{w}_t(e) \leftarrow \hat{w}_t(e) + \text{rad}_t(e),\ \forall e \in [n]$
  $\tilde{M}_t = \texttt{MaxOracle}(\mathcal{M} \setminus \mathcal{S}(\hat{M}_*), \bar{\boldsymbol{w}}_t)$
  if $\texttt{MinW}(\hat{M}_*, \underline{\boldsymbol{w}}_t) \geq \texttt{MinW}(\tilde{M}_t, \bar{\boldsymbol{w}}_t)$ **then**
    return $\hat{M}_*$
  end if
  $c_t \leftarrow \text{argmin}_{e \in \hat{M}_*} \underline{w}_t(e)$
  $F_t \leftarrow \{e \in \hat{B}_{\text{sub}} : \bar{w}_t(e) > \underline{w}_t(c_t)\}$
  $p_t \leftarrow \text{argmax}_{e \in F_t \cup \{c_t\}} \text{rad}_t(e)$
  Play p_t, and observe the reward
  Update empirical means $\hat{w}_{t+1}(p_t)$
  Update the number of samples $T_{t+1}(p_t)$
end for

Theorem 1 (Fixed-confidence Upper Bound)**.** *With probability at least $1 - \delta$, algorithm* `BLUCB` *(Algorithm 1) for CPE-B in the FC setting returns the optimal super arm with sample complexity*

$$O\left(\sum_{e \in [n]} \frac{R^2}{(\Delta_e^{\text{C}})^2} \ln\left(\sum_{e \in [n]} \frac{R^2 n}{(\Delta_e^{\text{C}})^2 \delta}\right)\right).$$

Base-arm-gap dependent sample complexity. Owing to the bottleneck-adaptive sample strategy, the reward gap Δ_e^{C} (Definition 1(a)(b)) is just defined as the difference between some critical bottleneck value and $w(e)$ itself, instead of the bottleneck gap between two super arms, and thus our result depends on the tight base-arm-level (instead of super-arm-level) gaps. For example, in Figure 1, `BLUCB` only spends $\tilde{O}((\frac{2}{\Delta^2_{e_2,e_1}} + \sum_{i=3,4,5,6} \frac{1}{\Delta^2_{e_i,e_1}}) \ln \delta^{-1})$ samples, while a naive adaptation of prior CPE algorithms [10, 13, 15] with the bottleneck reward function will cause a loose super-arm-gap dependent result $\tilde{O}(\sum_{e_i, i \in [6]} \frac{1}{\Delta^2_{M_*, M_{\text{sub}}}} \ln \delta^{-1})$. Regarding the optimality, Theorem 1 matches the lower bound (presented in Section 4) for some family of instances (up to a logarithmic factor). However, in general cases there still exists a gap on those needless base arms $\tilde{N}$ (e_3, e_5, e_6 in Figure 1), which are not contained in the lower bound. Next, we show how to bridge this gap.

3.2 Remove Dependence on Unnecessary Base Arms under Small δ

Challenges of avoiding unnecessary base arms. Under the bottlenck reward function, in each sub-optimal super arm M_{sub}, only the base arms with rewards lower than `OPT` (base arms in N) can determine the relationship of bottleneck values between M_* and M_{sub} (the bottleneck of M_{sub} is the most efficient choice to do this), and the others (base arms in $\tilde{N}$) are useless for revealing the sub-optimality of M_{sub}. Hence, to determine M_*, all we need is to sample the base arms in M_* and the *bottlenecks from all sub-optimal super arms*, denoted by B_{sub}, to see that each sub-optimal super arm contains at least one base arm that is worse than anyone in M_*. However, before sampling, (i) we do not know which is M_* that should be taken as the comparison benchmark, and in each M_{sub}, which base arm is its bottleneck (included in B_{sub}). Also, (ii) under combinatorial setting, how to efficiently collect B_{sub} from all sub-optimal super arms is another challenge.

To handle these challenges, we propose algorithm `BLUCB-Parallel` based on the explore-verify-parallel framework [25, 9]. `BLUCB-Parallel` (Algorithm 2) simultaneously simulates multiple `BLUCB-Verify`$_k$ (Algorithm 3) with confidence $\delta_k^V = \delta/2^{k+1}$ for $k \in \mathbb{N}$. `BLUCB-Verify`$_k$ first calls

Algorithm 4 `BLUCB-Explore`, sub-algorithm of `BLUCB-Verify`, the *key algorithm*

1: **Input:** $\mathcal{M}$, $\kappa = 0.01$ and `MaxOracle`.
2: Initialize: play each $e \in [n]$ once, and update empirical means $\hat{\boldsymbol{w}}_{n+1}$ and T_{n+1}
3: **for** $t = n+1, n+2, \ldots$ **do**
4: $\quad \text{rad}_t(e) \leftarrow R\sqrt{2\ln(\frac{4nt^3}{\kappa})/T_t(e)},\ \forall e \in [n]$
5: $\quad \underline{w}_t(e) \leftarrow \hat{w}_t(e) - \text{rad}_t(e),\ \forall e \in [n]$
6: $\quad \bar{w}_t(e) \leftarrow \hat{w}_t(e) + \text{rad}_t(e),\ \forall e \in [n]$
7: $\quad M_t \leftarrow \texttt{MaxOracle}(\mathcal{M}, \underline{\boldsymbol{w}}_t)$
8: $\quad \hat{B}_{\text{sub},t} \leftarrow \texttt{BottleneckSearch}(\mathcal{M}, M_t, \underline{\boldsymbol{w}}_t)$
9: $\quad$ **if** $\bar{w}_t(e) \le \frac{1}{2}(\texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t) + \underline{w}_t(e))$ for all $e \in \hat{B}_{\text{sub},t}$ **then**
10: $\quad\quad$ **return** $M_t, \hat{B}_{\text{sub},t}$
11: $\quad$ **end if**
12: $\quad c_t \leftarrow \text{argmin}_{e \in M_t} \underline{w}_t(e)$
13: $\quad \hat{B}'_{\text{sub},t} \leftarrow \{e \in \hat{B}_{\text{sub},t} : \bar{w}_t(e) > \frac{1}{2}(\texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t) + \underline{w}_t(e))\}$
14: $\quad p_t \leftarrow \text{argmax}_{e \in \hat{B}'_{\text{sub},t} \cup \{c_t\}} \text{rad}_t(e)$
15: $\quad$ Play p_t, and observe the reward
16: $\quad$ Update empirical means $\hat{w}_{t+1}(p_t)$
17: $\quad$ Update the number of samples $T_{t+1}(p_t)$
18: **end for**

`BLUCB-Explore` (Algorithm 4) to guess an optimal super arm $\hat{M}_*$ and collect a *near bottleneck set* $\hat{B}_{\text{sub}}$ with *constant confidence* κ, and then uses the required confidence δ_k^V to verify the correctness of $\hat{M}_*$ by only sampling base arms in $\hat{M}_*$ and $\hat{B}_{\text{sub}}$. Through parallel simulations, `BLUCB-Parallel` guarantees the $1-\delta$ correctness.

The *key component* of this framework is `BLUCB-Explore` (Algorithm 4), which provides a hypothesized answer $\hat{M}_*$ and critical base arms $\hat{B}_{\text{sub}}$ for verification to accelerate its identification process. Below we first describe the procedure of `BLUCB-Explore`, and then explicate its two *innovative techniques*, i.e. offline subroutine and stopping condition, developed to handle the challenges (i),(ii). `BLUCB-Explore` employs the subroutine $\texttt{BottleneckSearch}(\mathcal{M}, M_{\text{ex}}, \boldsymbol{v})$ to return the set of bottleneck base arms from all super arms in $\mathcal{M} \setminus \mathcal{S}(M_{\text{ex}})$ with respect to weight vector $\boldsymbol{v}$. At each timestep, we first calculate the best super arm M_t under lower reward confidence bound $\underline{\boldsymbol{w}}_t$, and call `BottleneckSearch` to collect the bottlenecks $\hat{B}_{\text{sub},t}$ from all super arms in $\mathcal{M} \setminus \mathcal{S}(M_t)$ with respect to $\underline{\boldsymbol{w}}_t$ (Line 8). Then, we use a stopping condition (Line 9) to examine if M_t is correct and $\hat{B}_{\text{sub},t}$ is close enough to $\hat{B}_{\text{sub}}$ (with confidence κ). If so, M_t and $\hat{B}_{\text{sub},t}$ are eligible for verification and returned; otherwise, we play a base arm from M_t and $\hat{B}_{\text{sub},t}$, which is most necessary for achieving the stopping condition. In the following, we explicate the two innovative techniques in `BLUCB-Explore`.

Efficient "bottleneck-searching" offline subroutine. $\texttt{BottleneckSearch}(\mathcal{M}, M_{\text{ex}}, \boldsymbol{v})$ (Line 8) serves as a fast offline procedure to collect bottlenecks from all super arms in given decision class $\mathcal{M} \setminus \mathcal{S}(M_{\text{ex}})$ with respect to $\boldsymbol{v}$. To achieve efficiency, the main idea behind `BottleneckSearch` is to avoid enumerating super arms in the combinatorial space, but only enumerate base arms $e \in [n]$ to check if e is the bottleneck of some super arm in $\mathcal{M} \setminus \mathcal{S}(M_{\text{ex}})$. We achieve this by removing all base arms with rewards lower than $v(e)$ and examining whether there exists a feasible super arm M that contains e in the remaining decision class. If so, e is the bottleneck of M and added to the output (more procedures are designed to exclude $\mathcal{S}(M_{\text{ex}})$). This fast offline subroutine solves challenge (ii) on computation complexity (see the supplementary material for its pseudo-codes and details).

Delicate "check-near-bottleneck" stopping condition. The stopping condition (Line 9) aims to ensure the returned $\hat{B}_{\text{sub},t} = \hat{B}_{\text{sub}}$ to satisfy the following Property (1): for each sub-optimal super arm M_{sub}, some base arm e such that $w(e) \le \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M_{\text{sub}}, \boldsymbol{w}))$ is included in $\hat{B}_{\text{sub}}$, which implies that e is near to the actual bottleneck of M_{sub} within $\frac{1}{2}\Delta_{M_*, M_{\text{sub}}}$, and cannot be anyone in $\tilde{N}$. Property (1) is crucial for `BLUCB-Verify` to achieve the optimal sample complexity, since it guarantees that in verification using $\hat{B}_{\text{sub}}$ to verify M_* just costs the same order of samples as using B_{sub}, which matches the lower bound. In the following, we explain why this stopping condition can guarantee Property (1).

If the stopping condition (Line 9) holds, i.e., $\forall e \in \hat{B}_{\text{sub},t}, \bar{w}_t(e) \le \frac{1}{2}(\texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t) + \underline{w}_t(e))$, using the definition of `BottleneckSearch`, we have that for any $M' \in \mathcal{M} \setminus \mathcal{S}(M_t)$, its bottleneck e' with respect to $\underline{\boldsymbol{w}}_t$ is included in $\hat{B}_{\text{sub},t}$ and satisfies that

$$w(e') \le \bar{w}_t(e') \overset{\text{(a)}}{\le} \frac{1}{2}(\texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t) + \texttt{MinW}(M', \underline{\boldsymbol{w}}_t)) \le \frac{1}{2}(\texttt{MinW}(M_t, \boldsymbol{w}) + \texttt{MinW}(M', \boldsymbol{w})),$$

where inequality (a) comes from $\bar{w}_t(e') \le \frac{1}{2}(\texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t) + \underline{w}_t(e'))$ and $\underline{w}_t(e')) = \texttt{MinW}(M', \underline{\boldsymbol{w}}_t)$. Hence, we can defer that $\texttt{MinW}(M', \boldsymbol{w}) \le w(e') \le \frac{1}{2}(\texttt{MinW}(M_t, \boldsymbol{w}) + \texttt{MinW}(M', \boldsymbol{w}))$ for any $M' \in \mathcal{M} \setminus \mathcal{S}(M_t)$, and thus $M_t = M_*$ (with confidence κ). In addition, the returned $\hat{B}_{\text{sub},t}$ satisfies Property (1). This stopping condition offers knowledge of a hypothesized optimal super arm $\hat{M}_*$ and a near bottleneck set $\hat{B}_{\text{sub}}$ for verification, which solves the challenge (i) and enables the overall sample complexity to achieve the optimality for small enough δ. Note that these two techniques are new in the literature, which are specially designed for handling the unique challenges of CPE-B.

We formally state the sample complexity of `BLUCB-Parallel` in Theorem 2.

Theorem 2 (Improved Fixed-confidence Upper Bound)**.** *For any $\delta < 0.01$, with probability at least $1-\delta$, algorithm* `BLUCB-Parallel` *(Algorithm 2) for CPE-B in the FC setting returns M_* and takes the expected sample complexity*

$$O\left(\sum_{e \in M_* \cup N} \frac{R^2}{(\Delta_e^{\mathrm{C}})^2} \ln\left(\frac{1}{\delta} \sum_{e \in M_* \cup N} \frac{R^2 n}{(\Delta_e^{\mathrm{C}})^2}\right) + \sum_{e \in \tilde{N}} \frac{R^2}{(\Delta_e^{\mathrm{C}})^2} \ln\left(\sum_{e \in \tilde{N}} \frac{R^2 n}{(\Delta_e^{\mathrm{C}})^2}\right)\right).$$

Results without dependence on $\tilde{N}$ in the dominant term. Let $H_V = \sum_{e \in M_* \cup N} \frac{R^2}{(\Delta_e^{\mathrm{C}})^2}$ and $H_E = \sum_{e \in [n]} \frac{R^2}{(\Delta_e^{\mathrm{C}})^2}$ denote the verification and exploration hardness, respectively. Compared to `BLUCB` (Theorem 1), the sample complexity of `BLUCB-Parallel` removes the redundant dependence on $\tilde{N}$ in the $\ln \delta^{-1}$ term, which guarantees better performance when $\ln \delta^{-1} \ge \frac{H_E}{H_E - H_V}$, i.e., $\delta \le \exp(-\frac{H_E}{H_E - H_V})$. This sample complexity matches the lower bound (within a logarithmic factor) under small enough δ. For the example in Figure 1, `BLUCB-Parallel` only requires $\tilde{O}((\frac{2}{\Delta^2_{e_2,e_1}} + \frac{1}{\Delta^2_{e_4,e_1}}) \ln \delta^{-1})$ samples, which are just enough efforts (optimal) for identifying M_*.

Time Complexity. All our algorithms can run in polynomial time, and the running time mainly depends on the offline oracles. For example, on s-t path instances with E edges and V vertices, the used offline procedures `MaxOracle` and `BottleneckSearch` only spend $O(E)$ and $O(E^2(E+V))$ time, respectively. See the supplementary material for more time complexity analysis.

4 Lower Bound for the Fixed-Confidence Setting

In this section, we establish a matching sample complexity lower bound for CPE-B in the FC setting. To formally state our results, we first define the notion of *δ-correct algorithm* as follows. For any confidence parameter $\delta \in (0,1)$, we call an algorithm $\mathcal{A}$ a δ-correct algorithm if for the fixed-confidence CPE-B problem, $\mathcal{A}$ returns the optimal super arm with probability at least $1-\delta$.

Theorem 3 (Fixed-confidence Lower Bound)**.** *There exists a family of instances for the fixed-confidence CPE-B problem, for which given any $\delta \in (0, 0.1)$, any δ-correct algorithm has the expected sample complexity*

$$\Omega\left(\sum_{e \in M_* \cup N} \frac{R^2}{(\Delta_e^{\mathrm{C}})^2} \ln\left(\frac{1}{\delta}\right)\right).$$

This lower bound demonstrates that the sample complexity of `BLUCB-Parallel` (Theorem 2) is optimal (within a logarithmic factor) under small enough δ, since its $\ln \delta^{-1}$ (dominant) term does not depend on unnecessary base arms $\tilde{N}$ either. In addition, if we impose some constraint on the constructed instances, the sample complexity of `BLUCB` (Theorem 1) can also match the lower bound up to a logarithmic factor (see the supplementary material for details).

Notice that, both our lower and upper bounds depend on the tight base-arm-level (instead of super-arm-level) gaps, and capture the *bottleneck insight*: different base arms in one super arm play distinct roles in determining its (sub)-optimality and impose different influences on the problem hardness.

5 Algorithm for the Fixed-Budget Setting

For CPE-B in the FB setting, we design a novel algorithm `BSAR` that adopts a special acceptance scheme for bottleneck identification. We allow `BSAR` to access an efficient accept-reject oracle `AR-Oracle`, which takes an accepted base arm e or $\perp$, a rejected base arm set R and a weight vector

Algorithm 5 BSAR, algorithm for CPE-B in the FB setting

1: **Input:** budget T, $\mathcal{M}$, and $\texttt{AR}-\texttt{Oracle}$.
2: $\tilde{\log}(n) \leftarrow \sum_{i=1}^{n} \frac{1}{i}$. $\tilde{T}_0 \leftarrow 0$. $A_1, R_1 \leftarrow \varnothing$.
3: **for** $t = 1, \ldots, n$ **do**
4: $\quad \tilde{T}_t \leftarrow \left\lceil \frac{T-n}{\tilde{\log}(n)(n-t+1)} \right\rceil$
5: $\quad U_t \leftarrow [n] \setminus (A_t \cup R_t)$
6: $\quad$ Play each $e \in U_t$ for $\tilde{T}_t - \tilde{T}_{t-1}$ times
7: $\quad$ Update empirical mean $\hat{w}_t(e), \forall e \in U_t$
8: $\quad \hat{w}_t(e) \leftarrow \infty$ for all $e \in A_t$
9: $\quad M_t \leftarrow \texttt{AR-Oracle}(\perp, R_t, \hat{\boldsymbol{w}}_t)$
10: $\quad$ **for** each $e \in U_t$ **do**
11: $\quad\quad$ **if** $e \in M_t$ **then**
12: $\quad\quad\quad \tilde{M}_{t,e} \leftarrow \texttt{AR-Oracle}(\perp, R_t \cup \{e\}, \hat{\boldsymbol{w}}_t)$
13: $\quad\quad$ **else**
14: $\quad\quad\quad \tilde{M}_{t,e} \leftarrow \texttt{AR-Oracle}(e, R_t, \hat{\boldsymbol{w}}_t)$
15: $\quad\quad$ **end if**
16: $\quad\quad$ // AR-Oracle returns $\perp$ if the calculated feasible set is empty
17: $\quad$ **end for**
18: $\quad p_t \leftarrow \operatorname{argmax}_{e \in U_t} \texttt{MinW}(M_t, \hat{\boldsymbol{w}}_t) - \texttt{MinW}(\tilde{M}_{t,e}, \hat{\boldsymbol{w}}_t)$
19: $\quad$ // $\texttt{MinW}(\perp, \hat{\boldsymbol{w}}_t) = -\infty$
20: $\quad$ **if** $p_t \in M_t$ **then**
21: $\quad\quad A_{t+1} \leftarrow A_t \cup \{p_t\}, R_{t+1} \leftarrow R_t$
22: $\quad$ **else**
23: $\quad\quad A_{t+1} \leftarrow A_t, R_{t+1} \leftarrow R_t \cup \{p_t\}$
24: $\quad$ **end if**
25: **end for**
26: **return** A_{n+1}

$\boldsymbol{v}$ as inputs, and returns an optimal super arm from the decision class $\mathcal{M}(e, R) = \{M \in \mathcal{M} : e \in M, R \cap M = \varnothing\}$ with respect to $\boldsymbol{v}$, i.e., $\texttt{AR-Oracle} \in \operatorname{argmax}_{M \in \mathcal{M}(e,R)} \texttt{MinW}(M, \boldsymbol{w})$. If $\mathcal{M}(e, R)$ is empty, AR-Oracle simply returns $\perp$. Such an efficient oracle exists for many decision classes, e.g., paths, matchings and spanning trees (see the supplementary material for implementation details).

BSAR allocates the sample budget T to n phases adaptively, and maintains the accepted set A_t, rejected set R_t and undetermined set U_t. In each phase, we only sample base arms in U_t and set the empirical rewards of base arms in A_t to infinity (Line 8). Then, we call AR-Oracle to compute the empirical best super arm M_t. For each $e \in U_t$, we forbid R_t and constrain e inside/outside the calculated super arms and find the empirical best super arm $\tilde{M}_{t,e}$ from the restricted decision class (Lines 12,14). Then, we accept or reject the base arm p_t that maximizes the empirical reward gap between M_t and $\tilde{M}_{t,e}$, i.e., the one that is most likely to be in or out of M_* (Line 18).

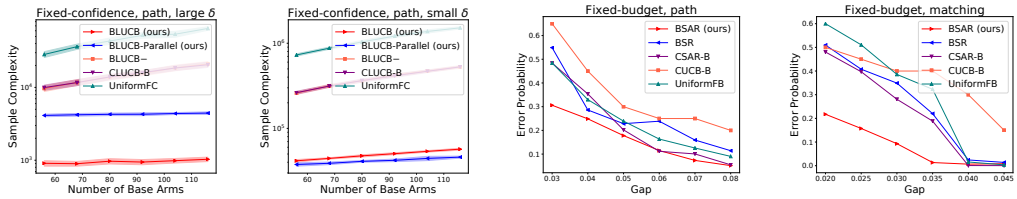
Special acceptance scheme for bottleneck and fast running time. The acceptance scheme $\hat{w}_t(e) \leftarrow \infty$ for all $e \in A_t$ (Line 8) is critical to the correctness and computation efficiency of BSAR. Since A_t and R_t are not pulled in phase t and their estimated rewards are not accurate enough, we need to avoid them disturbing the following calculation of empirical bottleneck values for super arms (Lines 9-18). By setting the empirical rewards of A_t to infinity, the estimation of bottleneck values for sub-optimal super arms M_{sub} avoids the disturbance of A_t, because each M_{sub} has at least one base arm with reward lower than OPT and this base arm will never be included in A_t (conditioned on high probability events). As for M_*, its empirical bottleneck value can be raised, but this only enlarges the empirical gap between M_* and M_{sub} and does not affect the correctness of the choice p_t (Line 18). Hence, this acceptance scheme guarantees the correctness of BSAR in bottleneck identification task.

Compared to existing CPE-L algorithm [10], they force the whole set A_t inside the calculated super arms in the oracle, i.e., replacing Lines 12,14 with $\texttt{AR-Oracle}(A_t, R_t \cup \{e\}, \hat{\boldsymbol{w}}_t)$ and $\texttt{AR-Oracle}(A_t \cup \{e\}, R_t, \hat{\boldsymbol{w}}_t)$, and deleting Line 8. Such acceptance strategy incurs *exponential-time* complexity on s-t path instances,[3] and *only works* for the linear reward function, where the common part A_t between two compared super arms can be canceled out. If one naively applies their acceptance strategy to our bottleneck problem, the common part A_t is possible to drag down (dominate) the empirical bottleneck values of all calculated super arms (Lines 9,12,14) and their empirical gaps will become all zeros (Line 18), which destroys the correctness of the choice p_t in theoretical analysis.

BSAR is the first to run in *polynomial time* on fixed-budget s-t path instances among existing CPE algorithms, owing to its skillful acceptance scheme and the simplified AR-Oracle (only work with one accepted base arm instead of A_t). Specifically, for E edges and V vertices, the time complexity of AR-Oracle is $O(E(E+V))$ and BSAR only spends $O(E^2(E+V))$ time in decision making.

Now we give the definitions of fixed-budget reward gap and problem hardness, and then formally state the error probability result of BSAR. For $e \in M_*$, $\Delta_e^{\text{B}} = \texttt{OPT} - \max_{M \in \mathcal{M}: e \notin M} \texttt{MinW}(M, \boldsymbol{w})$,

[3] Finding a s-t path which contains a given edge set is NP-hard. See the supplementary material for its proof.



(a) FC, path, large δ (b) FC, path, small δ (c) FB, path (d) FB, matching

Figure 2: Experiments for CPE-B in the FC/FB setting.

and for $e \notin M_*$, $\Delta_e^{\mathrm{B}} = \texttt{OPT} - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w})$. Let $\Delta_{(1)}^{\mathrm{B}}, \ldots, \Delta_{(n)}^{\mathrm{B}}$ be the permutation of $\Delta_1^{\mathrm{B}}, \ldots, \Delta_n^{\mathrm{B}}$ such that $\Delta_{(1)}^{\mathrm{B}} \leq \cdots \leq \Delta_{(n)}^{\mathrm{B}}$, and the fixed-budget problem hardness is defined as $H^{\mathrm{B}} = \max_{i \in [n]} \frac{i}{(\Delta_{(i)}^{\mathrm{B}})^2}$. Let $\tilde{\log}(n) = \sum_{i=1}^{n} \frac{1}{i}$.

Theorem 4. *For any $T > n$, algorithm* BSAR *(Algorithm 5) for CPE-B in the FB setting uses at most T samples and returns the optimal super arm with the error probability bounded by*

$$O\left(n^2 \exp\left(-\frac{T-n}{\tilde{\log}(n) R^2 H^B}\right)\right).$$

This error probability matches the state-of-the-art result [10] of existing CPE algorithms. In addition, compared to the uniform sampling algorithm, which plays all base arms equally and has $O(n \exp(-\frac{T}{R^2 n \Delta_{\min}^{-2}}))$ error probability with $\Delta_{\min} = \texttt{OPT} - \max_{M \neq M_*} \texttt{MinW}(M, \boldsymbol{w})$, our BSAR achieves a significantly better correctness guarantee.

6 Experiments

In this section, we conduct experiments for CPE-B in FC and FB settings on the top-k, s-t path and maximum cardinality matching instances. In the FC setting, we set a large $\delta = 0.005$ and a small $\delta = \exp(-1000)$, and perform 50 runs to plot the average sample complexity with 95% confidence intervals. In the FB setting, we perform 3000 runs and show the error probability across runs. Due to space limit, we defer the detailed setup description and more results to the supplementary material.

For the FC setting, we compare our BLUCB/BLUCB-Parallel with three baselines. BLUCB− replaces the sample strategy (Lines 12-14) of BLUCB with the one that uniformly plays a base arm in critical super arms. CLUCB-B simply replaces the linear reward function in prior CPE-L algorithm CLUCB [10] with bottleneck reward function. UniformFC is a fixed-confidence uniform sampling algorithm. As shown in Figures 2(a),2(b), BLUCB/BLUCB-Parallel significantly outperform the three baselines, which validates the statistical efficiency of our bottleneck-adaptive sample strategy. Under small δ, BLUCB-Parallel enjoys lower sample complexity than BLUCB due to its careful algorithmic design for avoiding unnecessary base arms, which matches our theoretical results.

For the FB setting, our BSAR is compared with four baseline algorithms. BSR removes the acceptance procedure of BSAR. CSAR-B simply replaces the linear reward calculation in prior CPE-L algorithm CSAR [10] with bottleneck reward calculation. CUCB-B follows the regret minimization algorithm CUCB [12] and returns the empirical best super arm after T (sample budget) timesteps. UniformFB is a fixed-budget uniform sampling algorithm. In Figures 2(c),2(d), BSAR achieves significantly lower error probability than all the baselines, which demonstrates that its special acceptance scheme effectively guarantees the correctness in bottleneck identification task.

7 Conclusion and Future Work

In this paper, we study the Combinatorial Pure Exploration with the Bottleneck reward function (CPE-B) problem in FC/FB settings. For the FC setting, we propose two novel algorithms, which achieve the optimal sample complexity for a broad family of instances (within a logarithmic factor), and establish a matching lower bound to demonstrate their optimality. For the FB setting, we propose an algorithm whose error probability matches the state-of-the-art result, and it is the first to run efficiently on fixed-budget path instances among existing CPE algorithms. The empirical evaluation also validates the superior performance of our algorithms. There are several interesting directions worth further research. One direction is to derive a lower bound for the FB setting, and another direction is to investigate the general nonlinear reward functions.

References

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *Advances in Neural Information Processing Systems*, 24:2312–2320, 2011.

[2] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 39–1, 2012.

[3] Jean-Yves Audibert, Sébastien Bubeck, and Remi Munos. Best arm identification in multi-armed bandits. In *Conference on Learning Theory*, pages 41–53, 2010.

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

[5] Ron Banner and Ariel Orda. Bottleneck routing games in communication networks. *IEEE Journal on Selected Areas in Communications*, 25(6):1173–1179, 2007. doi: 10.1109/JSAC.2007.070811.

[6] Séebastian Bubeck, Tengyao Wang, and Nitin Viswanathan. Multiple identifications in multi-armed bandits. In *International Conference on Machine Learning*, pages 258–265, 2013.

[7] Paolo M. Camerini. The min-max spanning tree problem and some extensions. *Information Processing Letters*, 7(1):10–14, 1978.

[8] Lijie Chen, Anupam Gupta, and Jian Li. Pure exploration of multi-armed bandit under matroid constraints. In *Conference on Learning Theory*, pages 647–669, 2016.

[9] Lijie Chen, Anupam Gupta, Jian Li, Mingda Qiao, and Ruosong Wang. Nearly optimal sampling algorithms for combinatorial pure exploration. In *Conference on Learning Theory*, pages 482–534, 2017.

[10] Shouyuan Chen, Tian Lin, Irwin King, Michael R Lyu, and Wei Chen. Combinatorial pure exploration of multi-armed bandits. In *Advances in Neural Information Processing Systems*, pages 379–387, 2014.

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

[12] 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*, 17 (1):1746–1778, 2016.

[13] Wei Chen, Yihan Du, Longbo Huang, and Haoyu Zhao. Combinatorial pure exploration for dueling bandit. In *International Conference on Machine Learning*, pages 1531–1541, 2020.

[14] Richard Combes, Mohammad Sadegh Talebi Mazraeh Shahi, Alexandre Proutiere, et al. Combinatorial bandits revisited. *Advances in Neural Information Processing Systems*, 28:2116–2124, 2015.

[15] Yihan Du, Yuko Kuroki, and Wei Chen. Combinatorial pure exploration with full-bandit or partial linear feedback. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[16] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 7:1079–1105, 2006.

[17] Victor Gabillon, Mohammad Ghavamzadeh, Alessandro Lazaric, and Sébastien Bubeck. Multi-bandit best arm identification. In *Advances in Neural Information Processing Systems*, 2011.

[18] Victor Gabillon, Alessandro Lazaric, Mohammad Ghavamzadeh, Ronald Ortner, and Peter Bartlett. Improved learning complexity in combinatorial pure exploration bandits. In *Artificial Intelligence and Statistics*, pages 1004–1012, 2016.

[19] Loukas Georgiadis, Giuseppe F Italiano, Luigi Laura, and Nikos Parotsidis. 2-vertex connectivity in directed graphs. *Information and Computation*, 261:248–264, 2018.

[20] A. V. Goldberg. Finding a maximum density subgraph. Technical report, University of California Berkeley, 1984.

[21] Yuri Gurevich and Saharon Shelah. Expected computation time for hamiltonian path problem. *SIAM Journal on Computing*, 16(3):486–502, 1987.

[22] Dorit S Hochbaum and Sung-Pil Hong. About strongly polynomial time algorithms for quadratic optimization over submodular constraints. *Mathematical programming*, 69(1):269–309, 1995.

[23] Weiran Huang, Jungseul Ok, Liang Li, and Wei Chen. Combinatorial pure exploration with continuous and separable reward functions and its applications. In *International Joint Conference on Artificial Intelligence*, pages 2291–2297, 2018.

[24] Shivaram Kalyanakrishnan, Ambuj Tewari, Peter Auer, and Peter Stone. PAC subset selection in stochastic multi-armed bandits. In *International Conference on Machine Learning*, pages 655–662, 2012.

[25] Zohar S Karnin. Verification based solution for structured mab problems. In *Advances in Neural Information Processing Systems*, pages 145–153, 2016.

[26] Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On the complexity of best-arm identification in multi-armed bandit models. *The Journal of Machine Learning Research*, 17(1): 1–42, 2016.

[27] Samir Khuller and Barna Saha. On finding dense subgraphs. In *International Colloquium on Automata, Languages, and Programming*, pages 597–608. Springer, 2009.

[28] Yuko Kuroki, Atsushi Miyauchi, Junya Honda, and Masashi Sugiyama. Online dense subgraph discovery via blurred-graph feedback. In *International Conference on Machine Learning*, pages 5522–5532, 2020.

[29] Yuko Kuroki, Liyuan Xu, Atsushi Miyauchi, Junya Honda, and Masashi Sugiyama. Polynomial-time algorithms for multiple-arm identification with full-bandit feedback. *Neural Computation*, 32(9):1733–1773, 2020.

[30] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

[31] Andrea Locatelli, Maurilio Gutzeit, and Alexandra Carpentier. An optimal algorithm for the thresholding bandit problem. In *International Conference on Machine Learning*, pages 1690–1698, 2016.

[32] M Peinhardt and V Kaibel. On the bottleneck shortest path problem. *Technical Report*, 2006.

[33] Abraham P Punnen and KPK Nair. Improved complexity bound for the maximum cardinality bottleneck bipartite matching problem. *Discrete Applied Mathematics*, 55(1):91–93, 1994.

[34] Chao Tao, Saúl Blanco, and Yuan Zhou. Best arm identification in linear bandits with linear dimension dependency. In *International Conference on Machine Learning*, pages 4877–4886, 2018.

[35] 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.

[36] Hoang Tuy, Saied Ghannadan, Athanasios Migdalas, and Peter Värbrand. A strongly polynomial algorithm for a concave production-transportation problem with a fixed number of nonlinear variables. *Mathematical Programming*, 72(3):229–258, 1996.

[37] Wenwei Yue, Changle Li, and Guoqiang Mao. Urban traffic bottleneck identification based on congestion propagation. In *2018 IEEE International Conference on Communications*, pages 1–6, 2018. doi: 10.1109/ICC.2018.8422108.

[38] Yiyi Zhou, Rongrong Ji, Xiaoshuai Sun, Gen Luo, Xiaopeng Hong, Jinsong Su, Xinghao Ding, and Ling Shao. K-armed bandit based multi-modal network architecture search for visual question answering. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1245–1254, 2020.

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]
   (b) Did you describe the limitations of your work? [Yes] We present them in our future work (Secion 7).
   (c) Did you discuss any potential negative societal impacts of your work? [No] This work is a theoretical study, and we believe that it does not have 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] We state the unique assumption of M_* in Secion 2.
   (b) Did you include complete proofs of all theoretical results? [Yes] We present all proofs in the supplementary material.
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)? [Yes] We include the code, data, and instructions in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [No] Our algorithms do not have any training procedure.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See our experimental results in Section 6 and supplemental material.
   (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)? [Yes] This work does not involve GPU computation. We include the hardcore information in the supplementary material.
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] This work is a theoretical study, which does not involve any asset issue.
   (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] This work is a theoretical study, which does not involve human subject issues.
   (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]