Combinatorial Pure Exploration with Bottleneck Reward Function

Yihan Du
IIIS, Tsinghua University
Beijing, China
duyh18@mails.tsinghua.edu.cn

Yuko Kuroki
The University of Tokyo / RIKEN
Tokyo, Japan
yukok@is.s.u-tokyo.ac.jp

Wei Chen
Microsoft Research
Beijing, China
weic@microsoft.com

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 [31, 36, 4, 2] is a classic model to solve the exploration-exploitation trade-off in online decision making. Pure exploration [3, 25, 7, 32] 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 [17, 25, 7, 30], top-k arm under matriod constraints [9] and multi-bandit best arm identification [18, 7].

The Combinatorial Pure Exploration (CPE) framework, firstly proposed by Chen et al. [11], encompasses a rich class of pure exploration problems [3, 25, 9]. 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

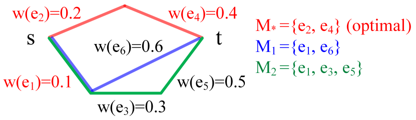
35th Conference on Neural Information Processing Systems (NeurIPS 2021).

with the maximum expected reward. While Chen et al. [11] 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]

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 [38], 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 [39], 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 [11, 10, 24], 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 [11, 12, 16] 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 the unnecessary e_3, e_5, e_6). Hence, our challenge falls on how to achieve such efficient sampling in an 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).



Base arm gap:

$\Delta_{e_2,e_1} = 0.2 - 0.1 = 0.1$

$\Delta_{e_4,e_1} = 0.4 - 0.1 = 0.3$

Super arm gap:

$\Delta_{M_*,M_1} = \min\{0.2,\ 0.4\} - \min\{0.1,\ 0.6\} = 0.1$

$\Delta_{M_*,M_2} = \min\{0.2,\ 0.4\} - \min\{0.1,\ 0.3,\ 0.5\} = 0.1$

Let $\Delta_{M_*,M_{\text{sub}}} \overset{\text{def}}{=} \Delta_{M_*,M_1} = \Delta_{M_*,M_2}$

Figure 1: Illustrating 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 the bottleneck identification task. BSAR achieves the state-of-the-art error probability and is the first to run efficiently on fixed-budget path instances,

[1] The algorithmic designs and analytical tools (e.g., symmetric difference and exchange set) in [11] all rely on the linear property and cannot be applied to nonlinear reward cases, e.g, the bottleneck reward problem.

compared to existing CPE algorithms. All our proposed algorithms run in *polynomial time*.[2] The experimental results demonstrate that our algorithms significantly outperform the baselines. Due to space limit, we defer all the proofs to Appendix.

1.1 Related Work

In the following we briefly review the related work in the CPE literature. Chen et al. [11] firstly propose the CPE model and only consider the linear reward function (CPE-L), and their results for CPE-L are further improved by [19, 10]. Huang et al. [24] 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. [12] 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. [29] 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. [16] 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 [13, 15, 14] 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 $1, 2, \dots, n$. 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, 11, 32, 35]. Let $\boldsymbol{w} = (w(1), \dots, w(n))^\top$ 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 $\texttt{MinW}(M, \boldsymbol{w}) = \min_{e \in M} w(e)$,[3] 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 [17, 11, 10, 12], we assume that M_* is unique, and this assumption can be removed in our extension to the PAC learning setting (see Section B.3).

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 [25, 7, 32, 10], 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.

[2] Here "polynomial time" refers to polynomial time in the number of base arms n (which is equal to the number of edges E in our considered instances such as s-t paths, matchings and spanning trees).

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

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

Algorithm 1 illustrates the proposed algorithm BLUCB for CPE-B in the FC setting. Here $\mathcal{S}(M_t)$ denotes 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 $\texttt{MinW}(M', \boldsymbol{w}) \leq \texttt{MinW}(M_t, \boldsymbol{w})$. 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* $\texttt{MaxOracle}(\mathcal{F}, \boldsymbol{v})$, which returns an optimal super arm from $\mathcal{F}$ with respect to $\boldsymbol{v}$, i.e., $\texttt{MaxOracle}(\mathcal{F}, \boldsymbol{v}) \in \text{argmax}_{M \in \mathcal{F}} \texttt{MinW}(M, \boldsymbol{v})$. For $\mathcal{F} = \mathcal{M}$ (Line 7), such an efficient oracle exists for many decision classes, such as the bottleneck shortest path [33], bottleneck bipartite matching [34] and minimum bottleneck spanning tree [8] 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 a 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 [11, 12, 16] 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 $\texttt{MinW}(M, \boldsymbol{w})$, 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 $N = \{e \mid e \notin M_*, w(e) < \texttt{OPT}\}$ and $\tilde{N} = \{e \mid e \notin M_*, w(e) \geq \texttt{OPT}\}$, 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

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 $k = 0, 1, \dots$, let $\texttt{BLUCB-Verify}_k$ be the sub-algorithm BLUCB-Verify with $\delta_k = \frac{\delta}{2^{k+1}}$
for $t = 1, 2, \dots$ do
    for each $k = 0, 1, \dots$ such that $t \bmod 2^k = 0$ do
        Start or resume $\texttt{BLUCB-Verify}_k$ with one sample, and then suspend $\texttt{BLUCB-Verify}_k$
        if $\texttt{BLUCB-Verify}_k$ returns an answer $M_{\text{out}}$, then return $M_{\text{out}}$
    end for
end for
```

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

```
Input: $\mathcal{M}$, $\delta^V \in (0, 0.01)$ and MaxOracle.
$\kappa \leftarrow 0.01$
$\hat{M}_*, \hat{B}_{\text{sub}} \leftarrow \texttt{BLUCB-Explore}(\mathcal{M}, \kappa, \texttt{MaxOracle})$
Initialize: play each $e \in [n]$ once, and update empirical means $\hat{\boldsymbol{w}}_{n+1}$ and $T_{n+1}$
for $t = n+1, n+2, \dots$ 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
```

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

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

Now we present the sample complexity upper bound of BLUCB.

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*

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

Base-arm-gap dependent sample complexity. Owing to the bottleneck-adaptive sample strategy, the reward gap Δ^{C}_e (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 [11, 12, 16] 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 bottleneck 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

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

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 [26, 10]. `BLUCB-Parallel` (Algorithm 2) simultaneously simulates multiple `BLUCB-Verify`$_k$ (Algorithm 3) with confidence $\delta^V_k = \delta/2^{k+1}$ for $k \in \mathbb{N}$. `BLUCB-Verify`$_k$ first calls `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^V_k 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 `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. `BottleneckSearch`$(\mathcal{M}, M_{\text{ex}}, \boldsymbol{v})$ (Line 8) serves as an efficient 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 efficient offline subroutine solves challenge (ii) on computation complexity (see Section B.2.1 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 $w(e) \leq \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M_{\text{sub}}, \boldsymbol{w}))$ 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., $\forall e \in \hat{B}_{\text{sub},t}, \bar{w}_t(e) \leq \frac{1}{2}(\texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t) + \underline{w}_t(e))$, using the definition of $\texttt{BottleneckSearch}$, 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

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

where inequality (a) comes from $\bar{w}_t(e') \leq \frac{1}{2}(\texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t) + \underline{w}_t(e'))$ and $\underline{w}_t(e')) = \texttt{MinW}(M', \underline{\boldsymbol{w}}_t)$. Hence, we can defer that $\texttt{MinW}(M', \boldsymbol{w}) \leq w(e') \leq \frac{1}{2}(\texttt{MinW}(M_t, \boldsymbol{w}) + \texttt{MinW}(M', \boldsymbol{w}))$ 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 $\texttt{BLUCB-Parallel}$ in Theorem 2.

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

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

Results without dependence on $\tilde{N}$ in the dominant term. Let $H_V = \sum_{e \in M_* \cup N} \frac{R^2}{(\Delta^{\text{C}}_e)^2}$ and $H_E = \sum_{e \in [n]} \frac{R^2}{(\Delta^{\text{C}}_e)^2}$ denote the verification and exploration hardness, respectively. Compared to $\texttt{BLUCB}$ (Theorem 1), the sample complexity of $\texttt{BLUCB-Parallel}$ removes the redundant dependence on $\tilde{N}$ in the $\ln \delta^{-1}$ term, which guarantees better performance when $\ln \delta^{-1} \geq \frac{H_E}{H_E - H_V}$, i.e., $\delta \leq \exp(-\frac{H_E}{H_E - H_V})$. This sample complexity matches the lower bound (within a logarithmic factor) under small enough δ. For the example in Figure 1, $\texttt{BLUCB-Parallel}$ 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_*.

The condition $\delta < 0.01$ in Theorem 2 is due to that the used explore-verify-parallel framework [26, 10] needs a small δ to guarantee that $\texttt{BLUCB-Parallel}$ can maintain the same order of sample complexity as its sub-algorithm $\texttt{BLUCB-Verify}_k$. Prior pure exploration works [26, 10] also have such condition on δ.

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 $\texttt{MaxOracle}$ and $\texttt{BottleneckSearch}$ only spend $O(E)$ and $O(E^2(E+V))$ time, respectively. See Section E 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*

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

This lower bound demonstrates that the sample complexity of $\texttt{BLUCB-Parallel}$ (Theorem 2) is optimal (within a logarithmic factor) under small enough δ, since its $\ln \delta^{-1}$ (dominant) term does

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

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

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 Appendix C for details). The condition $\delta < 0.1$ comes from the lower bound analysis, which ensures that the binary entropy of finding a correct or wrong answer can be lower bounded by $\ln \delta^{-1}$. Existing pure exploration works [11, 10] also have such condition on δ in their lower bounds.

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 $\boldsymbol{v}$ 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., 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 Appendix D 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 polynomial 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 to disturb the following calculation of empirical bottleneck values (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 CSAR [11], they force the whole set A_t inside the calculated super arms in the oracle, i.e., replacing Lines 12,14 with AR-Oracle$(A_t, R_t \cup \{e\}, \hat{\boldsymbol{w}}_t)$ and 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,[4] 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 $e \in M_*$, $\Delta^{\mathrm{B}}_e = \texttt{OPT} - \max_{M \in \mathcal{M}: e \notin M} \texttt{MinW}(M, \boldsymbol{w})$, and for $e \notin M_*$, $\Delta^{\mathrm{B}}_e = \texttt{OPT} - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w})$. Let $\Delta^{\mathrm{B}}_{(1)}, \dots, \Delta^{\mathrm{B}}_{(n)}$ be the permutation of $\Delta^{\mathrm{B}}_1, \dots, \Delta^{\mathrm{B}}_n$ such that $\Delta^{\mathrm{B}}_{(1)} \leq \dots \leq \Delta^{\mathrm{B}}_{(n)}$, and the fixed-budget problem hardness is defined as $H^{\mathrm{B}} = \max_{i \in [n]} \frac{i}{(\Delta^{\mathrm{B}}_{(i)})^2}$. Let $\tilde{\log}(n) = \sum_{i=1}^{n} \frac{1}{i}$.

Theorem 4 (Fixed-budget Upper Bound)**.** *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*

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

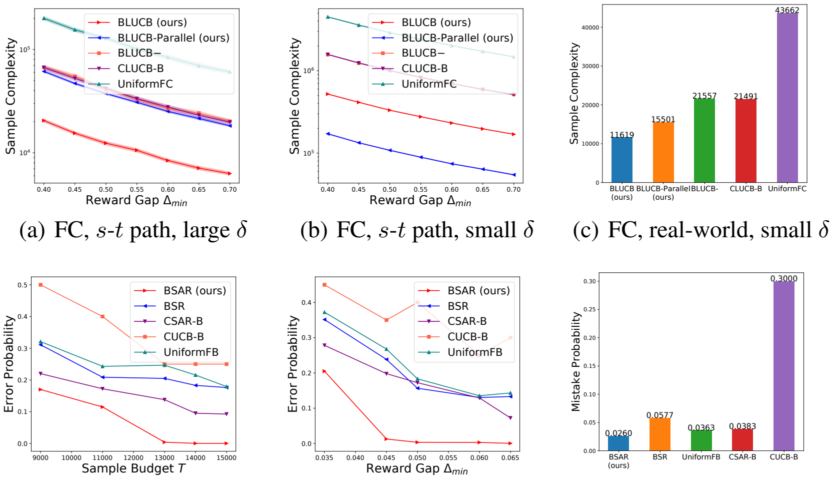
Compared to the uniform sampling algorithm, which plays all base arms equally and has $O(n \exp(-\frac{T}{R^2 n \Delta_{\min}^{-2}}))$ error probability with $\Delta_{\min} = \texttt{OPT} - \max_{M \neq M_*} \texttt{MinW}(M, \boldsymbol{w})$, Theorem 4 achieves a significantly better correctness guarantee (when $\Delta^B_e > \Delta_{\min}$ for most $e \in [n]$). In addition, when our CPE-B problem reduces to conventional K-armed pure exploration problem [7], Theorem 4 matches existing state-of-the-art result in [7]. To our best knowledge, the lower bound for the fixed-budget setting in the CPE literature [11, 24, 29, 16] remains open.

Our error probability analysis falls on taking advantage of the bottleneck property to handle the disturbance from the accepted arm set (which are not pulled sufficiently) and guaranteeing the estimation accuracy of bottleneck rewards. The differences between our analysis and prior analysis for CSAR [11] are highlighted as follows: (i) Prior analysis [11] relies on the linear property to cancel out the common part between two super arms when calculating their reward gap, in order to avoid the disturbance of accepted arms. In contrast, to achieve this goal, we utilize the special acceptance scheme of BSAR to exclude all accepted arms in the calculation of bottleneck rewards, which effectively addresses the perturbation of inaccurate estimation on accepted arms. (ii) Prior analysis [11] mainly uses the "exchange sets" technique, which only works for the linear reward function and leads to the dependence on the parameter of decision class structures. Instead, our analysis exploits the bottleneck property to establish confidence intervals in the base arm level, and effectively avoids the dependence on the parameter of decision class structures.

6 Experiments

In this section, we conduct experiments for CPE-B in FC/FB settings on synthetic and real-world datasets. The synthetic dataset consists of the s-t path and matching instances. For the s-t path instance, the number of edges (base arms) $n = 85$, and the expected reward of edges $w(e) = [0, 10.5]$ ($e \in [n]$). The minimum reward gap of any two edges (which is also the minimum gap of bottleneck values between two super arms) is denoted by $\Delta_{\min} \in [0.4, 0.7]$. For the matching instances, we use a 5×3 complete bipartite graph, where $n = 15$, $w(e) = [0.1, 1.08]$ and $\Delta_{\min} \in [0.03, 0.07]$. We change $\Delta_{\min}$ to generate a series of instances with different hardness (plotted points in Figures 2(a),2(b),2(e)). In terms of the real-world dataset, we use the data of American airports and the number of available seats of flights in 2002, provided by the International Air Transportation Association database (www.iata.org) [6]. Here we regard an airport as a vertex and a direct flight connecting two airports as an edge (base arm), and also consider the number of available seats of a flight as the expected reward of an edge. Our objective is to find an air route connecting the starting and destination airports which maximizes the minimum number of available seats among its passing

[4] Finding a s-t path which contains a given edge set is NP-hard. See Appendix D.3 for its proof.



(a) FC, s-t path, large δ

(b) FC, s-t path, small δ

(c) FC, real-world, small δ

(d) FB, matching

(e) FB, matching

(f) FB, real-world

Figure 2: Experiments for CPE-B in the FC/FB setting on synthetic and real-world datasets.

flights. In this instance, $n = 9$ and $w(e) \in [0.62, 1.84]$. We present the detailed graphs with specific values of $w(e)$ for the s-t path, matching and real-world air route instances in Appendix A.

In the FC setting, we set a large $\delta = 0.005$ and a small $\delta = \exp(-1000)$, and perform 50 independent runs to plot average sample complexity with 95% confidence intervals. In the FB setting, we set sample budget $T \in [6000, 15000]$, and perform 3000 independent runs to show the error probability across runs. For all experiments, the random reward of each edge $e \in [n]$ is i.i.d. drawn from Gaussian distribution $\mathcal{N}(w(e), 1)$.

Experiments for the FC setting. We compare our BLUCB/BLUCB-Parallel with three baselines. BLUCB− is an ablation variant of BLUCB, which replaces the sample strategy (Lines 12-14) with the one that uniformly samples a base arm in critical super arms. CLUCB-B [11] is the state-of-the-art fixed-confidence CPE-L algorithm run with bottleneck reward function. UniformFC is a fixed-confidence uniform sampling algorithm. As shown in Figures 2(a)-2(c), BLUCB and BLUCB-Parallel achieve better performance than 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 to avoid playing unnecessary base arms, which matches our theoretical results.

Experiments for the FB setting. Our BSAR is compared with four baselines. As an ablation variant of BSAR, BSR removes the special acceptance scheme of BSAR. CSAR-B [11] is the state-of-the-art fixed-budget CPE-L algorithm implemented with bottleneck reward function. CUCB-B [14] is a regret minimization algorithm allowing nonlinear reward functions, and in pure exploration experiments we let it return the empirical best super arm after T (sample budget) timesteps. UniformFB is a fixed-budget uniform sampling algorithm. One sees from Figures 2(d)-2(f) that, BSAR achieves significantly better error probability than all the baselines, which demonstrates that its special acceptance scheme effectively guarantees the correctness for the 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.

Acknowledgments and Disclosure of Funding

The work of Yihan Du is supported in part by the Technology and Innovation Major Project of the Ministry of Science and Technology of China under Grant 2020AAA0108400 and 2020AAA0108403. Yuko Kuroki is supported by Microsoft Research Asia and JST ACT-X 1124477.

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.

[6] Alain Barrat, Marc Barthelemy, Romualdo Pastor-Satorras, and Alessandro Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences*, 101(11):3747–3752, 2004.

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

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

[19] 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.

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

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

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

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

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

[29] 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.

[30] 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.

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

[32] 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.

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

[34] 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.

[35] 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.

[36] 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.

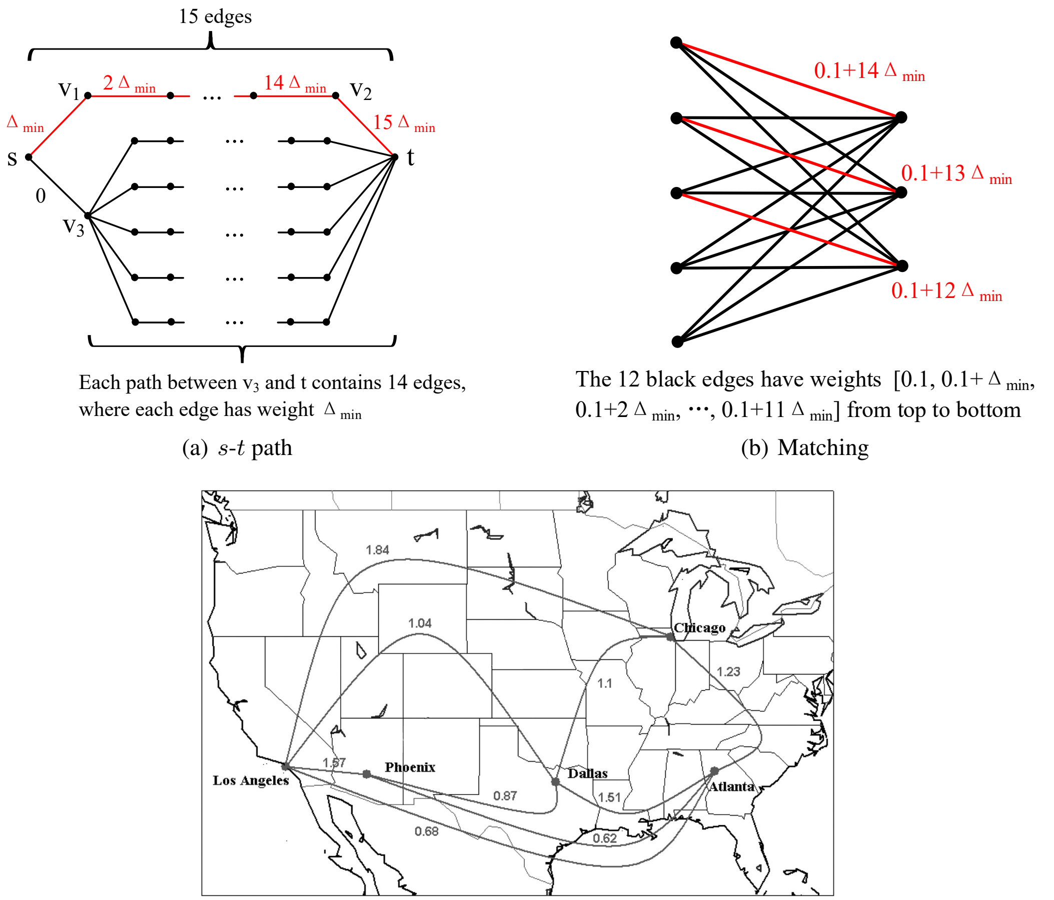
[37] Hoang Tuy, Saied Ghannadan, Athanasios Migdalas, and Peter Värbrand. A strongly polynomial algorithm for a concave production-transportation problem with a fixed number of nonlinear variables. *Mathematical Programming*, 72(3):229–258, 1996.

[38] 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.

[39] 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.

Appendix

A More Details of Experimental Setups


15 edges
Δ min
s
0
v1
2 Δ min
…
14 Δ min
v2
15 Δ min
t
v3
Each path between v3 and t contains 14 edges, where each edge has weight Δ min
0.1+14 Δ min
0.1+13 Δ min
0.1+12 Δ min
The 12 black edges have weights [0.1, 0.1+ Δ min, 0.1+2 Δ min, ⋯, 0.1+11 Δ min] from top to bottom
1.84
1.04
Chicago
1.23
1.1
Los Angeles
1.57
Phoenix
Dallas
Atlanta
0.87
1.51
0.68
0.62

(a) s-t path

(b) Matching

(c) Real-world air route

Figure 3: Graphs of the s-t path, matching and real-world air route instances in our experiments.

In this section, we supplement more details of graphs and the expected rewards of edges (base arms) for the s-t path, matching and real-world air route instances in our experiments.

Figure 3(a) shows the graph of s-t path instance. The red path contains 15 edges with weights $[\Delta_{\min}, 2\Delta_{\min}, \dots, 15\Delta_{\min}]$ and is the optimal s-t path with the maximum bottleneck value. There are 5 paths connecting v_3 and t, and each of them contains 14 edges with weights $\Delta_{\min}$. In this instance, we set $\Delta_{\min} \in [0.4, 0.7]$.

As shown in Figure 3(b), the matching instance uses a 5×3 complete bipartite graph with $n = 15$ edges. The red matching is the optimal one, which contains three edges with weights $[0.1 + 14\Delta_{\min}, 0.1 + 13\Delta_{\min}, 0.1 + 12\Delta_{\min}]$. The remaining 12 black edges have weights $[0.1, 0.1 + \Delta_{\min}, \dots, 0.1 + 11\Delta_{\min}]$ from top to bottom. In this instance, $\Delta_{\min} \in [0.03, 0.07]$.

Figure 3(c) illustrates the graph of real-world air route instance, which is originated from [6]. We regard an airport (e.g., Los Angeles) as a vertex and a direct flight connecting two airports (e.g., Los Angeles ↔ Chicago) as an edge. The number marked on each edge denotes the number of available seats of this flight, i.e., the expected reward of this edge. Our objective is to find an air route connecting Los Angeles and Atlanta, and the optimal route is [Los Angeles ↔ Chicago ↔ Atlanta].

B CPE-B in the Fixed-Confidence Setting

B.1 Proof for Algorithm `BLUCB`

In this subsection, we prove the sample complexity of Algorithm `BLUCB` (Theorem 1).

In order to prove Theorem 1, we first introduce the following Lemmas 1-5. For ease of notation, we define a function $\texttt{MinE}(M, \boldsymbol{v})$ to return the base arm with the minimum reward in M with respect to weight vector $\boldsymbol{v}$, i.e., $\texttt{MinE}(M, \boldsymbol{v}) \in \operatorname{argmin}_{e \in M} v(e)$.

Lemma 1 (Concentration)**.** *For any $t > 0$ and $e \in [n]$, defining the confidence radius $\mathrm{rad}_t(e) = R\sqrt{\frac{2\ln(\frac{4nt^3}{\delta})}{T_t(e)}}$ and the events*

$$\xi_t = \{\forall e \in [n],\ |w(e) - \hat{w}_t(e)| < \mathrm{rad}_t(e)\}$$

and

$$\xi = \bigcap_{t=1}^{\infty} \xi_t,$$

then, we have

$$\Pr[\xi] \geq 1 - \delta.$$

Proof. Since for any $e \in [n]$, the reward distribution of base arm e has an R-sub-Gaussian tail and the mean of $w(e)$, according to the Hoeffding's inequality, we have that for any $t > 0$ and $e \in [n]$,

$$\begin{aligned}\Pr\left[|w(e) - \hat{w}_t(e)| \geq R\sqrt{\frac{2\ln(\frac{4nt^3}{\delta})}{T_t(e)}}\right] =& \sum_{s=1}^{t-1} \Pr\left[|w(e) - \hat{w}_t(e)| \geq R\sqrt{\frac{2\ln(\frac{4nt^3}{\delta})}{T_t(e)}},\ T_t(e) = s\right] \\ \leq& \sum_{s=1}^{t-1} \frac{\delta}{2nt^3} \\ \leq& \frac{\delta}{2nt^2}\end{aligned}$$

Using a union bound over $e \in [n]$, we have

$$\Pr\left[\xi_t\right] \leq \frac{\delta}{2t^2}$$

and thus

$$\begin{aligned}\Pr\left[\xi\right] \geq& 1 - \sum_{t=1}^{\infty} \Pr[\neg \xi_t] \\ \geq& 1 - \sum_{t=1}^{\infty} \frac{\delta}{2t^2} \\ \geq& 1 - \delta\end{aligned}$$

□

Lemma 2. *Assume that event ξ occurs. Then, if algorithm `BLUCB` (Algorithm 1) terminates at round t, we have $M_t = M_*$.*

Proof. According to the stop condition (Line 9 of Algorithm 1), when algorithm `BLUCB` terminates at round t, we have that for any $M \in \mathcal{M} \setminus \mathcal{S}(M_t)$,

$$\texttt{MinW}(M_t, \boldsymbol{w}) \geq \texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t) \geq \texttt{MinW}(M, \bar{\boldsymbol{w}}_t) \geq \texttt{MinW}(M, \boldsymbol{w}).$$

For any $M \in \mathcal{S}(M_t)$, according to the property of the bottleneck reward function, we have

$$\texttt{MinW}(M_t, \boldsymbol{w}) \geq \texttt{MinW}(M, \boldsymbol{w}).$$

Thus, we have $\texttt{MinW}(M_t, \boldsymbol{w}) \geq \texttt{MinW}(M, \boldsymbol{w})$ for any $M \neq M_t$ and according to the unique assumption of M_*, we obtain $M_t = M_*$. □

Lemma 3. *Assume that event ξ occurs. For any $e \in M_*$, if* $\mathrm{rad}_t(e) < \frac{\Delta^{\mathrm{C}}_e}{4} = \frac{1}{4}(w(e) - \max_{M \neq M_*} \texttt{MinW}(M, \boldsymbol{w}))$*, then, base arm e will not be pulled at round t, i.e., $p_t \neq e$.*

Proof. Suppose that for some $e \in M_*$, $\mathrm{rad}_t(e) < \frac{\Delta^{\mathrm{C}}_e}{4} = \frac{1}{4}(w(e) - \max_{M \neq M_*} \texttt{MinW}(M, \boldsymbol{w}))$ and $p_t = e$. According to the selection strategy of p_t, we have that $\mathrm{rad}_t(c_t) < \frac{\Delta^{\mathrm{C}}_e}{4}$ and $\mathrm{rad}_t(d_t) < \frac{\Delta^{\mathrm{C}}_e}{4}$.

Case (i): If e is selected from M_*, then one of M_t and $\tilde{M}_t$ is M_* such that $e = \texttt{MinE}(M_*, \underline{\boldsymbol{w}}_t)$, and the other is a sub-optimal super arm M'. Let $e' = \texttt{MinE}(M', \underline{\boldsymbol{w}}_t)$. $\underline{w}(e') \leq \underline{w}(\texttt{MinE}(M', \boldsymbol{w})) \leq w(\texttt{MinE}(M', \boldsymbol{w})) = \texttt{MinW}(M', \boldsymbol{w})$. $\{e, e'\} = \{c_t, d_t\}$. Then,

$$\begin{aligned}\underline{w}(e) - \bar{w}(e') \geq & w(e) - \underline{w}(e') - 2\mathrm{rad}_t(e) - 2\mathrm{rad}_t(e') \\ > & w(e) - \texttt{MinW}(M', \boldsymbol{w}) - \Delta^{\mathrm{C}}_e \\ \geq & 0.\end{aligned}$$

Then, we have

$$\texttt{MinW}(M_*, \underline{\boldsymbol{w}}_t) = \underline{w}(e) > \bar{w}(e') \geq \texttt{MinW}(M', \bar{\boldsymbol{w}}_t),$$

and algorithm $\texttt{BLUCB}$ must have stopped, which gives a contradiction.

Case (ii): If e is selected from a sub-optimal super arm M, then one of M_t and $\tilde{M}_t$ is M such that $e = \texttt{MinE}(M, \underline{\boldsymbol{w}}_t)$. Since $e \in M_*$, we have $w(e) \geq \texttt{MinW}(M_*, \boldsymbol{w}) > \texttt{MinW}(M, \boldsymbol{w})$ and thus $w(e) - \texttt{MinW}(M, \boldsymbol{w}) = w(e) - w(\texttt{MinE}(M, \boldsymbol{w})) > 0$. Then,

$$\begin{aligned}\underline{w}(e) - \underline{w}(\texttt{MinE}(M, \boldsymbol{w})) \geq & w(e) - 2\mathrm{rad}_t(e) - \texttt{MinW}(M, \boldsymbol{w}) \\ > & w(e) - \texttt{MinW}(M, \boldsymbol{w}) - \frac{\Delta^{\mathrm{C}}_e}{2} \\ > & 0,\end{aligned}$$

which contradicts $e = \texttt{MinE}(M, \underline{\boldsymbol{w}}_t)$. □

Lemma 4. *Assume that event ξ occurs. For any $e \notin M_*, w(e) \geq \texttt{MinW}(M_*, \boldsymbol{w})$, if* $\mathrm{rad}_t(e) < \frac{\Delta^{\mathrm{C}}_e}{2} = \frac{1}{2}(w(e) - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w}))$*, then, base arm e will not be pulled at round t, i.e., $p_t \neq e$.*

Proof. Suppose that for some $e \notin M_*, w(e) \geq \texttt{MinW}(M_*, \boldsymbol{w})$, $\mathrm{rad}_t(e) < \frac{\Delta^{\mathrm{C}}_e}{2} = \frac{1}{2}(w(e) - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w}))$ and $p_t = e$. According to the selection strategy of p_t, we have that $\mathrm{rad}_t(c_t) < \frac{\Delta^{\mathrm{C}}_e}{2}$ and $\mathrm{rad}_t(d_t) < \frac{\Delta^{\mathrm{C}}_e}{2}$.

Since $e \notin M_*$, e is selected from a sub-optimal super arm M. One of M_t and $\tilde{M}_t$ is M such that $e = \texttt{MinE}(M, \underline{\boldsymbol{w}}_t)$. Since $w(e) \geq \texttt{MinW}(M_*, \boldsymbol{w}) > \texttt{MinW}(M, \boldsymbol{w})$, we have $w(e) - \texttt{MinW}(M, \boldsymbol{w}) = w(e) - w(\texttt{MinE}(M, \boldsymbol{w})) > 0$. Then,

$$\begin{aligned}\underline{w}(e) - \underline{w}(\texttt{MinE}(M, \boldsymbol{w})) \geq & w(e) - 2\mathrm{rad}_t(e) - \texttt{MinW}(M, \boldsymbol{w}) \\ > & w(e) - \texttt{MinW}(M, \boldsymbol{w}) - \Delta^{\mathrm{C}}_e \\ \geq & 0,\end{aligned}$$

which contradicts $e = \texttt{MinE}(M, \underline{\boldsymbol{w}}_t)$. □

Lemma 5. *Assume that event ξ occurs. For any $e \notin M_*, w(e) < \texttt{MinW}(M_*, \boldsymbol{w})$, if* $\mathrm{rad}_t(e) < \frac{\Delta^{\mathrm{C}}_e}{4} = \frac{1}{4}(\texttt{MinW}(M_*, \boldsymbol{w}) - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w}))$*, then, base arm e will not be pulled at round t, i.e., $p_t \neq e$.*

Proof. Suppose that for some $e \notin M_*, w(e) < \texttt{MinW}(M_*, \boldsymbol{w})$, $\mathrm{rad}_t(e) < \frac{\Delta^{\mathrm{C}}_e}{4} = \frac{1}{4}(\texttt{MinW}(M_*, \boldsymbol{w}) - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w}))$ and $p_t = e$. According to the selection strategy of p_t, we have that $\mathrm{rad}_t(c_t) < \frac{\Delta^{\mathrm{C}}_e}{4}$ and $\mathrm{rad}_t(d_t) < \frac{\Delta^{\mathrm{C}}_e}{4}$.

Case (i): If one of M_t and $\tilde{M}_t$ is M_*, then the other is a sub-optimal super arm M such that $e = \texttt{MinE}(M, \underline{\boldsymbol{w}}_t)$. Let $f = \texttt{MinE}(M_*, \underline{\boldsymbol{w}}_t)$. $\{e, f\} = \{c_t, d_t\}$. Then, we have

$$\underline{w}(f) - \bar{w}(e) \geq w(f) - 2\mathrm{rad}_t(f) - \underline{w}(e) - 2\mathrm{rad}_t(e)$$

$$\begin{aligned}
&>w(f) - \underline{w}(e) - \Delta_e^{\mathrm{C}} \\
&\geq \texttt{MinW}(M_*, \boldsymbol{w}) - \underline{w}(\texttt{MinE}(M, \boldsymbol{w})) - \Delta_e^{\mathrm{C}} \\
&\geq \texttt{MinW}(M_*, \boldsymbol{w}) - \texttt{MinW}(M, \boldsymbol{w}) - \Delta_e^{\mathrm{C}} \\
&\geq 0.
\end{aligned}$$

Thus,

$$\texttt{MinW}(M_*, \boldsymbol{w}) \geq \underline{w}(\texttt{MinE}(M_*, \boldsymbol{w})) \geq \underline{w}(f) > \bar{w}(e) \geq \texttt{MinW}(M, \bar{\boldsymbol{w}}_t)$$

and algorithm $\texttt{BLUCB}$ must have stopped, which gives a contradiction.

Case (ii): If neither M_t nor $\tilde{M}_t$ is M_* and $e = c_t$, i.e., $e = \texttt{MinE}(M_t, \underline{\boldsymbol{w}}_t)$, we have

$$\bar{w}(d_t) \geq \texttt{MinW}(\tilde{M}_t, \bar{\boldsymbol{w}}_t) \geq \texttt{MinW}(M_*, \bar{\boldsymbol{w}}_t) \geq \texttt{MinW}(M_*, \boldsymbol{w})$$

and

$$\underline{w}(d_t) = \texttt{MinW}(\tilde{M}_t, \underline{\boldsymbol{w}}_t) \leq \texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t) \leq \texttt{MinW}(M_t, \boldsymbol{w}),$$

and thus

$$\begin{aligned}
2\text{rad}_t(d_t) =& \bar{w}(d_t) - \underline{w}(d_t) \\
\geq& \texttt{MinW}(M_*, \boldsymbol{w}) - \texttt{MinW}(M_t, \boldsymbol{w}),
\end{aligned}$$

which contradicts $\text{rad}_t(d_t) < \frac{\Delta_e^{\mathrm{C}}}{4} < \frac{\Delta_e^{\mathrm{C}}}{2} \leq \frac{\texttt{MinW}(M_*, \boldsymbol{w}) - \texttt{MinW}(M_t, \boldsymbol{w})}{2}$.

Case (iii): If neither M_t nor $\tilde{M}_t$ is M_* and $e = d_t$, i.e., $e = \texttt{MinE}(\tilde{M}_t, \underline{\boldsymbol{w}}_t)$. Let $c(M_*, \tilde{M}_t) = \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(\tilde{M}_t, \boldsymbol{w}))$. If $c(M_*, \tilde{M}_t) < w(e) < \texttt{MinW}(M_*, \boldsymbol{w})$, we have

$$\begin{aligned}
\underline{w}(e) \geq& w(e) - 2\text{rad}_t(e) \\
>& c(M_*, \tilde{M}_t) - \frac{\Delta_e^{\mathrm{C}}}{2} \\
\geq& \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(\tilde{M}_t, \boldsymbol{w})) - \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) - \texttt{MinW}(\tilde{M}_t, \boldsymbol{w})) \\
=& \texttt{MinW}(\tilde{M}_t, \boldsymbol{w}) \\
\geq& \underline{w}(\texttt{MinE}(\tilde{M}_t, \boldsymbol{w})),
\end{aligned}$$

which contradicts $e = \texttt{MinE}(\tilde{M}_t, \underline{\boldsymbol{w}}_t)$.

If $\texttt{MinW}(\tilde{M}_t, \boldsymbol{w}) \leq w(e) \leq c(M_*, \tilde{M}_t)$, we have

$$\begin{aligned}
\texttt{MinW}(\tilde{M}_t, \bar{\boldsymbol{w}}_t) \leq& \bar{w}(e) \\
\leq& w(e) + 2\text{rad}_t(e) \\
<& c(M_*, \tilde{M}_t) + \frac{\Delta_e^{\mathrm{C}}}{2} \\
\leq& \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(\tilde{M}_t, \boldsymbol{w})) + \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) - \texttt{MinW}(\tilde{M}_t, \boldsymbol{w})) \\
=& \texttt{MinW}(M_*, \boldsymbol{w}) \\
\leq& \texttt{MinW}(M_*, \bar{\boldsymbol{w}}_t).
\end{aligned}$$

In addition, from the uniqueness of M_*, we have $M_* \notin \mathcal{S}(\tilde{M}_t)$. Thus, the inequality $\texttt{MinW}(\tilde{M}_t, \bar{\boldsymbol{w}}_t) < \texttt{MinW}(M_*, \bar{\boldsymbol{w}}_t)$ violates the optimality of $\tilde{M}_t$ with respect to $\bar{\boldsymbol{w}}_t$. □

Next, we prove Theorem 1.

Proof. For any $e \in [n]$, let $T(e)$ denote the number of samples for base arm e, and t_e denote the last timestep at which e is pulled. Then, we have $T_{t_e} = T(e) - 1$. Let T denote the total number of samples. According to Lemmas 3-5, we have

$$R\sqrt{\frac{2\ln(\frac{4nt_e^3}{\delta})}{T(e) - 1}} \geq \frac{1}{4}\Delta_e^{\mathrm{C}}$$

Thus, we obtain

$$T(e) \leq \frac{32R^2}{(\Delta_e^{\mathrm{C}})^2} \ln\left(\frac{4nt_e^3}{\delta}\right) + 1 \leq \frac{32R^2}{(\Delta_e^{\mathrm{C}})^2} \ln\left(\frac{4nT^3}{\delta}\right) + 1$$

Summing over $e \in [n]$, we have

$$T \leq \sum_{e\in[n]} \frac{32R^2}{(\Delta_e^{\mathrm{C}})^2} \ln\left(\frac{4nT^3}{\delta}\right) + n \leq \sum_{e\in[n]} \frac{96R^2}{(\Delta_e^{\mathrm{C}})^2} \ln\left(\frac{2nT}{\delta}\right) + n,$$

where $\sum_{e\in[n]} \frac{R^2}{(\Delta_e^{\mathrm{C}})^2} \geq n$. Then, applying Lemma 20, we have

$$\begin{aligned} T \leq & \sum_{e\in[n]} \frac{576R^2}{(\Delta_e^{\mathrm{C}})^2} \ln\left(\frac{2n^2}{\delta} \sum_{e\in[n]} \frac{96R^2}{(\Delta_e^{\mathrm{C}})^2}\right) + n \\ =& O\left(\sum_{e\in[n]} \frac{R^2}{(\Delta_e^{\mathrm{C}})^2} \ln\left(\sum_{e\in[n]} \frac{R^2n^2}{(\Delta_e^{\mathrm{C}})^2\delta}\right) + n\right) \\ =& O\left(\sum_{e\in[n]} \frac{R^2}{(\Delta_e^{\mathrm{C}})^2} \ln\left(\sum_{e\in[n]} \frac{R^2n}{(\Delta_e^{\mathrm{C}})^2\delta}\right)\right) \end{aligned}$$

□

B.2 Details for the Improved Algorithm BLUCB-Parallel

In this subsection, we describe algorithm BLUCB-Parallel and its sub-algorithms in details, present the pseudo-code of the offline subroutine BottleneckSearch in Algorithm 6 and give the proofs of theoretical results.

B.2.1 Detailed Algorithm Description

BLUCB-Parallel simulates multiple sub-algorithms BLUCB-Verify (Algorithm 3) with different confidence parameters in parallel. For $k \in \mathbb{N}$, BLUCB-Verify$_k$ denotes the sub-algorithm BLUCB-Verify with confidence parameter $\delta_k^V = \frac{\delta}{2^{k+1}}$ At each timestep t, we start or resume sub-algorithms BLUCB-Verify$_k$ such that t is divisible by 2^k with only one sample, and then suspend these sub-algorithms. Such parallel simulation is performed until there exist some sub-algorithm which terminates and returns the answer M_{out}, and then we output M_{out} as the answer of BLUCB-Parallel.

Algorithm BLUCB-Verify (Algorithm 3) calls Algorithm BLUCB-Explore (Algorithm 4) as its preparation procedure. Algorithm BLUCB-Explore uses a big constant confidence parameter κ to guesses an optimal super arm and an advice set $\hat{B}_{\text{sub}}$ which contains the bottleneck base arms for the sub-optimal super arms, and then algorithm BLUCB-Verify verifies the correctness of the answer provided by BLUCB-Explore with the given confidence parameter δ^V.

BLUCB-Explore first calculates an super arm M_t with the maximum pessimistic bottleneck value, and then uses a subroutine BottleneckSearch (Algorithm 6) to find the set of bottleneck base arms $\hat{B}_{\text{sub},t}$ from all super arms in $\mathcal{M} \setminus \mathcal{S}(M_t)$ with respect to the lower reward confidence bound $\underline{\boldsymbol{w}}_t$. Then, we check whether for any base arm $e \in \hat{B}_{\text{sub},t}$, the optimistic reward of e is lower than a half of its pessimistic reward plus the pessimistic bottleneck value of M_t. If this stopping condition holds, we simply return M_t as the hypothesized optimal super arm and $\hat{B}_{\text{sub},t}$ as the advice set. Otherwise, we find the bottleneck c_t from M_t with respect to $\underline{\boldsymbol{w}}_t$, and collect the set of base arms $\hat{B}'_{\text{sub},t}$ which violate the stopping condition from $\hat{B}_{\text{sub},t}$. Let $P_t^E \overset{\text{def}}{=} \hat{B}'_{\text{sub},t} \cup \{c_t\}$ denote the sampling set of BLUCB-Explore. We plays the base arm with the maximum confidence radius in P_t^E.

BLUCB-Verify first calculates the super arm $\tilde{M}_t$ with the maximum optimistic bottleneck value in $\mathcal{M} \setminus \mathcal{S}(\hat{M}_*)$, and checks whether the pessimistic bottleneck value of $\hat{M}_*$ is higher than the optimistic bottleneck value of $\tilde{M}_t$. If so, we can determine that the answer $\hat{M}_*$ is correct, and simply stop

Algorithm 6 `BottleneckSearch`, offline subroutine of `BLUCB-Explore`

```
Input: $\mathcal{M}$, $M_{\text{ex}}$, $\boldsymbol{v}$ and existence oracle ExistOracle($\mathcal{F}, e$): check if there exists a feasible super arm $M \in \mathcal{F}$ such that $e \in M$, and return $M$ if there exists and $\perp$ otherwise.
$A_{\text{out}} \leftarrow \varnothing$
for $e \in [n]$ do
    Remove all base arms with rewards lower than $v(e)$ from $\mathcal{M}$, and obtain $\mathcal{M}_{\geq v(e)}$
    if $e \in M_{\text{ex}}$ and $v(e) =$ MinW($M_{\text{ex}}, \boldsymbol{v}$) then
        for each $e_0 \in M_{\text{ex}} \setminus \{e\}$ do
            Remove $e_0$ from $\mathcal{M}_{\geq v(e)}$, and obtain $\mathcal{M}_{\geq v(e), -e_0}$
            if ExistOracle($\mathcal{M}_{\geq v(e), -e_0}, e$) $\neq \perp$, then $A_{\text{out}} \leftarrow A_{\text{out}} \cup \{e\}$ and break
        end for
    else if ExistOracle($\mathcal{M}_{\geq v(e)}, e$) $\neq \perp$ then
        $A_{\text{out}} \leftarrow A_{\text{out}} \cup \{e\}$
    end if
end for
return $A_{\text{out}}$
```

and return $\hat{M}_*$. Otherwise, we find the bottleneck c_t from M_t with respect to the lower reward confidence bound $\underline{\boldsymbol{w}}_t$, and collect the set of base arms F_t whose optimistic rewards are higher than the pessimistic bottleneck value of $\hat{M}_*$ from $\hat{B}_{\text{sub}}$. Let $P_t^V \overset{\text{def}}{=} F_t \cup \{c_t\}$ denote the sampling set of `BLUCB-Verify`. We samples the base arm with the maximum confidence radius in P_t^V.

Now, we discuss the skillful subroutine `BottleneckSearch`, which is formally defined as follows.

Definition 2 (`BottleneckSearch`)**.** *We define* `BottleneckSearch`$(\mathcal{M}, M, \boldsymbol{v})$ *as an algorithm that takes decision class $\mathcal{M}$, super arm $M \in \mathcal{M}$ and weight vector $\boldsymbol{v} \in \mathbb{R}^d$ as inputs and returns a set of base arms A_{out} that satisfies (i) for any $M' \in \mathcal{M} \setminus \mathcal{S}(M)$, there exists a base arm $e \in A_{\text{out}} \cap M'$ such that $\boldsymbol{v}(e) =$* `MinW`$(M', \boldsymbol{v})$*, and (ii) for any $e \in A_{\text{out}}$, there exists a super arm $M' \in \mathcal{M} \setminus \mathcal{S}(M)$ such that $e \in M'$ and $\boldsymbol{v}(e) =$* `MinW`$(M', \boldsymbol{v})$*.*

Algorithm 6 illustrates the implementation procedure of `BottleneckSearch`. We access the subroutine `BottleneckSearch` an efficient existence oracle `ExistOracle`$(\mathcal{F}, e)$, which returns a feasible super arm that contains base arm e from decision class $\mathcal{F}$ if there exists, and otherwise returns $\perp$, i.e., `ExistOracle`$(\mathcal{F}, e) \in \{M \in \mathcal{F} : e \in M\}$. Such efficient oracles exist for a wide family of decision classes. For example, for s-t paths, this problem can be reduced to the well-studied 2-vertex connectivity problem [20], which is polynomially tractable (see Section E.1 for the proof of reduction). For maximum cardinality matchings, we just need to remove e and its two end vertices, and then find a feasible maximum cardinality matching in the remaining graph. For spanning trees, we can just merge the vertices of e and find a feasible spanning tree in the remaining graph, which can also be solved efficiently.

In the subroutine `BottleneckSearch`, we enumerate all the base arms to collect the bottlenecks. For each enumerated base arm e, we first remove all base arms with the rewards lower than $w(e)$ from $\mathcal{M}$ and obtain a new decision class $\mathcal{M}_{\geq w(e)}$, in which the super arms only contain the base arms with the rewards at least $w(e)$. Then, we call `ExistOracle` to check whether there exists a feasible super arm M_e that contains e in $\mathcal{M}_{\geq w(e)}$. If there exist, then we obtain that e is the bottleneck of M_e with respect to $\boldsymbol{v}$, i.e., $e \in M_e$ and $w(e) =$ `MinW`$(M_e, \boldsymbol{v})$, and add e to the output set A_{out}. Otherwise, we can determine that e is not the bottleneck for any super arm in $\mathcal{M} \setminus \mathcal{S}(M_t)$ with respect to $\boldsymbol{v}$. However, for the particular e such that $e \in M_t$ and $w(e) =$ `MinW`$(M_t, \boldsymbol{v})$, directly calling `ExistOracle` can obtain the output M_t, which should be excluded. We solve this problem by repeatedly removing each base arm in $M_t \setminus \{e\}$ and then calling `ExistOracle` on the new decision classes, which can check whether e is some super arm's bottleneck apart from $\mathcal{M} \setminus \mathcal{S}(M_t)$.

B.2.2 Proof for Algorithm `BLUCB-Parallel`

Below we give the theoretical results for the proposed algorithms.

For algorithm `BLUCB-Explore`, define the events $\xi_{0,t} = \{\forall e \in [n], \ |w(e) - \hat{w}_t(e)| < \text{rad}_t(e)\}$ and $\xi_0 = \bigcap_{t=1}^{\infty} \xi_{0,t}$. Then, similar to Lemma 1, we have $\Pr[\xi_0] \geq 1 - \kappa$. For algorithm `BLUCB-Verify`, define the events $\xi_t = \{\forall e \in [n], \ |w(e) - \hat{w}_t(e)| < \text{rad}_t(e)\}$ and $\xi = \bigcap_{t=1}^{\infty} \xi_t$. Then, applying

Lemma 1, we have $\Pr[\xi] \ge 1 - \delta^V$. Let $M_{\text{second}} = \operatorname{argmax}_{M \in \mathcal{M} \setminus \{M_*\}} \texttt{MinW}(M, w)$ denote the second best super arm.

Lemma 6 (Correctness of `BLUCB-Explore`)**.** *For algorithm* `BLUCB-Explore`*, assume that event ξ_0 occurs. Then, if algorithm* `BLUCB-Explore` *(Algorithm 4) terminates at round t, we have that (i) $M_t = M_*$, (ii) for any $M \in \mathcal{M} \setminus \mathcal{S}(M_*)$, there exists a base arm $e \in \hat{B}_{\text{sub},t} \cap M$ satisfying $w(e) \le \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M, \boldsymbol{w}))$, i.e., $\Delta^{\mathrm{C}}_{\texttt{MinW}(M_*,\boldsymbol{w}),e} \ge \frac{1}{2}\Delta^{\mathrm{C}}_{M_*,M}$ and (iii) for any $e \in \hat{B}_{\text{sub},t}$, there exists a sub-optimal super arm $M \in \mathcal{M} \setminus \mathcal{S}(M_*)$ such that $e \in M$ and $w(e) \le \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M)) \le \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M_{\text{second}}, \boldsymbol{w}))$.*

Proof. According to the stop condition (Lines 9 of Algorithm 4) and the definition of `BottleneckSearch` (Definition 2), when algorithm `BLUCB-Explore` (Algorithm 4) terminates at round t, we have that for any $\mathcal{M} \setminus \mathcal{S}(M_t)$, there exists a base arm $e \in \hat{B}_{\text{sub},t} \cap M$ satisfying

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

and thus,

$$w(e) \le \frac{1}{2}(\texttt{MinW}(M_t, \boldsymbol{w}) + \texttt{MinW}(M, \boldsymbol{w})).$$

Then, we can obtain $M_t = M_*$. Otherwise, we cannot find any base arm $e \in M_*$ satisfying $w(e) \le \frac{1}{2}(\texttt{MinW}(M_t, \boldsymbol{w}) + \texttt{MinW}(M_*, \boldsymbol{w})) < \texttt{MinW}(M_*, \boldsymbol{w})$, where M_t is a sub-optimal super arm. Thus, we have (i) and (ii).

Now we prove (iii). According to the stop condition (Lines 9 of Algorithm 4), the definition of `BottleneckSearch` (Definition 2) and $M_t = M_*$, we have that for any $e \in \hat{B}_{\text{sub},t}$, there exists a sub-optimal super arm $\mathcal{M} \setminus \mathcal{S}(M_*)$ such that $e \in M$ and $\underline{w}_t(e) = \texttt{MinW}(M, \underline{\boldsymbol{w}}_t)$, and thus

$$\bar{w}_t(e) \le \frac{1}{2}(\texttt{MinW}(M_*, \underline{\boldsymbol{w}}_t) + \underline{w}_t(e)) = \frac{1}{2}(\texttt{MinW}(M_*, \underline{\boldsymbol{w}}_t) + \texttt{MinW}(M, \underline{\boldsymbol{w}}_t)),$$

Then, for any $e \in \hat{B}_{\text{sub},t}$,

$$w(e) \le \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M)) \le \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M_{\text{second}}, \boldsymbol{w})),$$

which completes the proof. □

Lemma 7. *For algorithm* `BLUCB-Explore`*, assume that event ξ_0 occurs. For any two base arms $e_1, e_2 \in [n]$ s.t. $w(e_1) < w(e_2)$, if $\mathrm{rad}_t(e_1) < \frac{1}{6}\Delta^{\mathrm{C}}_{e_2,e_1}$ and $\mathrm{rad}_t(e_2) < \frac{1}{6}\Delta^{\mathrm{C}}_{e_2,e_1}$, we have $\bar{w}_t(e_1) < \frac{1}{2}(\underline{w}_t(e_1) + \underline{w}_t(e_2))$.*

Proof. if $\mathrm{rad}_t(e_1) < \frac{1}{6}\Delta^{\mathrm{C}}_{e_2,e_1}$ and $\frac{1}{6}\Delta^{\mathrm{C}}_{e_2,e_1}$, we have

$$\begin{aligned}
\bar{w}_t(e_1) - \frac{1}{2}(\underline{w}_t(e_1) + \underline{w}_t(e_2)) &= \underline{w}_t(e_1) + 2\mathrm{rad}_t(e_1) - \frac{1}{2}(\underline{w}_t(e_1) + \underline{w}_t(e_2)) \\
&= \frac{1}{2}\underline{w}_t(e_1) - \frac{1}{2}\underline{w}_t(e_2) + 2\mathrm{rad}_t(e_1) \\
&\le \frac{1}{2}w_t(e_1) - \frac{1}{2}(w_t(e_2) - 2\mathrm{rad}_t(e_2)) + 2\mathrm{rad}_t(e_1) \\
&= \frac{1}{2}w_t(e_1) - \frac{1}{2}w_t(e_2) + \mathrm{rad}_t(e_2) + 2\mathrm{rad}_t(e_1) \\
&< \frac{1}{2}w_t(e_1) - \frac{1}{2}w_t(e_2) + \frac{1}{2}\Delta^{\mathrm{C}}_{e_2,e_1} \\
&= 0.
\end{aligned}$$

□

Lemma 8. *For algorithm* `BLUCB-Explore`*, assume that event ξ_0 occurs. For any $e \in M_*$, if $\mathrm{rad}_t(e) < \frac{\Delta^{\mathrm{C}}_e}{12} = \frac{1}{12}(w(e) - \max_{M \neq M_*} \texttt{MinW}(M, \boldsymbol{w}))$, then, base arm e will not be pulled at round t, i.e., $p_t \neq e$.*

Proof. Suppose that for some $e \in M_*$, $\text{rad}_t(e) < \frac{\Delta^{\text{C}}_e}{12} = \frac{1}{12}(w(e) - \max_{M \neq M_*} \texttt{MinW}(M, \boldsymbol{w}))$ and $p_t = e$.

According to the selection strategy of p_t, we have that for any $i \in P^E_t$, $\text{rad}_t(i) \leq \text{rad}_t(e) < \frac{\Delta^{\text{C}}_e}{12}$. From the definition of `BottleneckSearch` and c_t, we have that for any $i \in P^E_t$, there exists a super arm $M \in \mathcal{M}$ such that $i \in M$ and $\underline{w}_t(i) = \texttt{MinW}(M, \underline{\boldsymbol{w}}_t)$.

Case (i): Suppose that e is selected from a sub-optimal super arm M, i.e., $e \in M$ and $\underline{w}_t(e) = \texttt{MinW}(M, \underline{\boldsymbol{w}}_t)$. Then, using $\text{rad}_t(e) < \frac{\Delta^{\text{C}}_e}{12} < \frac{\Delta^{\text{C}}_e}{2}$, we have

$$\begin{aligned}\underline{w}_t(e) \geq & w(e) - 2\text{rad}_t(e)\\ > & w(e) - (w(e) - \texttt{MinW}(M_{\text{second}}, \boldsymbol{w}))\\ = & \texttt{MinW}(M_{\text{second}}, \boldsymbol{w})\\ \geq & \texttt{MinW}(M, \boldsymbol{w})\\ \geq & \texttt{MinW}(M, \underline{\boldsymbol{w}}_t)\end{aligned}$$

which gives a contradiction.

Case (ii): Suppose that e is selected from M_*, i.e., $\underline{w}_t(e) = \texttt{MinW}(M_*, \underline{\boldsymbol{w}}_t)$. We can obtain $M_* = M_t$. Otherwise,

$$\begin{aligned}\texttt{MinW}(M_*, \underline{\boldsymbol{w}}_t) = & \underline{w}_t(e)\\ \geq & w(e) - 2\text{rad}_t(e)\\ > & w(e) - (w(e) - \texttt{MinW}(M_{\text{second}}, \boldsymbol{w}))\\ = & \texttt{MinW}(M_{\text{second}}, \boldsymbol{w})\\ \geq & \texttt{MinW}(M_t, \boldsymbol{w})\\ \geq & \texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t)\end{aligned}$$

Thus, We have $M_* = M_t$. From the definition of `BottleneckSearch`, for any $e' \in \hat{B}'_{\text{sub},t}$, there exists a super arm $M' \in \mathcal{M} \setminus \{M_*\}$ such that $e' \in M'$ and $\underline{w}_t(e') = \texttt{MinW}(M', \underline{\boldsymbol{w}}_t)$. If $w(e') \geq \frac{1}{2}(w(e) + \texttt{MinW}(M', \boldsymbol{w}))$, using $\text{rad}_t(e') < \frac{\Delta^{\text{C}}_e}{12} < \frac{\Delta^{\text{C}}_e}{4}$, we have

$$\begin{aligned}\underline{w}_t(e') \geq & w(e') - 2\text{rad}_t(e')\\ > & \frac{1}{2}(w(e) + \texttt{MinW}(M', \boldsymbol{w})) - \frac{1}{2}(w(e) - M_{\text{second}})\\ \geq & \frac{1}{2}(w(e) + \texttt{MinW}(M', \boldsymbol{w})) - \frac{1}{2}(w(e) - \texttt{MinW}(M', \boldsymbol{w}))\\ = & \texttt{MinW}(M', \boldsymbol{w}),\end{aligned}$$

which gives a contradiction.

If $w(e') < \frac{1}{2}(w(e) + \texttt{MinW}(M', \boldsymbol{w}))$, we have

$$\begin{aligned}\Delta^{\text{C}}_{e,e'} = & w(e) - w(e')\\ > & w(e) - \frac{1}{2}(w(e) + \texttt{MinW}(M', \boldsymbol{w}))\\ = & \frac{1}{2}(w(e) - \texttt{MinW}(M', \boldsymbol{w}))\end{aligned}$$

Since $\text{rad}_t(e) < \frac{\Delta^{\text{C}}_e}{12} = \frac{1}{12}(w(e) - \texttt{MinW}(M_{\text{second}})) \leq \frac{1}{12}(w(e) - \texttt{MinW}(M', \boldsymbol{w})) \leq \frac{1}{6}\Delta^{\text{C}}_{e,e'}$ and $\text{rad}_t(e') \leq \text{rad}_t(e) < \frac{1}{6}\Delta^{\text{C}}_{e,e'}$, according to Lemma 7, we have

$$\begin{aligned}\bar{w}_t(e') < & \frac{1}{2}(\underline{w}_t(e) + \underline{w}_t(e'))\\ = & \frac{1}{2}(\texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t) + \underline{w}_t(e')),\end{aligned}$$

which contradicts the definition of $\hat{B}'_{\text{sub},t}$. □

Lemma 9. *For algorithm* `BLUCB-Explore`*, assume that event* ξ_0 *occurs. For any* $e \notin M_*, w(e) \geq \texttt{MinW}(M_*, \boldsymbol{w})$*, if* $\text{rad}_t(e) < \frac{\Delta^{\text{C}}_e}{2} = \frac{1}{2}(w(e) - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w}))$*, then, base arm e will not be pulled at round t, i.e.,* $p_t \neq e$*.*

Proof. Suppose that for some $e \notin M_*, w(e) \geq \texttt{MinW}(M_*, \boldsymbol{w})$, $\text{rad}_t(e) < \frac{\Delta^{\text{C}}_e}{2} = \frac{1}{2}(w(e) - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w}))$ and $p_t = e$.

According to the selection strategy of p_t, we have that for any $i \in P^E_t$, $\text{rad}_t(i) \leq \text{rad}_t(e) < \frac{\Delta^{\text{C}}_e}{12}$. From the definition of `BottleneckSearch` and c_t, we have that for any $i \in P^E_t$, there exists a super arm $M \in \mathcal{M}$ such that $i \in M$ and $\underline{w}_t(i) = \texttt{MinW}(M, \underline{\boldsymbol{w}}_t)$.

Since $e \notin M_*$, e is selected from a sub-optimal super arm M', i.e., $e \in M'$ and $\underline{w}_t(e) = \texttt{MinW}(M', \underline{\boldsymbol{w}}_t)$. Then,

$$\begin{aligned}
\underline{w}_t(e) \geq & w(e) - 2\text{rad}_t(e) \\
> & w(e) - (w(e) - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w})) \\
= & \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w}) \\
\geq & \texttt{MinW}(M', \boldsymbol{w}) \\
\geq & \texttt{MinW}(M', \underline{\boldsymbol{w}}_t)
\end{aligned}$$

which contradicts $\underline{w}_t(e) = \texttt{MinW}(M', \underline{\boldsymbol{w}}_t)$. □

Lemma 10. *For algorithm* `BLUCB-Explore`*, assume that event* ξ_0 *occurs. For any* $e \notin M_*, w(e) < \texttt{MinW}(M_*, \boldsymbol{w})$*, if* $\text{rad}_t(e) < \frac{\Delta^{\text{C}}_e}{12} = \frac{1}{12}(\texttt{MinW}(M_*, \boldsymbol{w}) - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w}))$*, then, base arm e will not be pulled at round t, i.e.,* $p_t \neq e$*.*

Proof. Suppose that for some $e \notin M_*, w(e) < \texttt{MinW}(M_*, \boldsymbol{w})$, $\text{rad}_t(e) < \frac{\Delta^{\text{C}}_e}{12} = \frac{1}{12}(\texttt{MinW}(M_*, \boldsymbol{w}) - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w}))$ and $p_t = e$.

According to the selection strategy of p_t, we have that for any $i \in P^E_t$, $\text{rad}_t(i) \leq \text{rad}_t(e) < \frac{\Delta^{\text{C}}_e}{12}$. From the definition of `BottleneckSearch` and c_t, we have that for any $i \in P^E_t$, there exists a super arm $M \in \mathcal{M}$ such that $i \in M$ and $\underline{w}_t(i) = \texttt{MinW}(M, \underline{\boldsymbol{w}}_t)$. Thus, there exists a sub-optimal super arm $M' \in \mathcal{M}$ such that $e \in M'$ and $\underline{w}_t(e) = \texttt{MinW}(M', \underline{\boldsymbol{w}}_t)$.

Case (i) Suppose that $w(e) \geq \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M', \boldsymbol{w}))$. Using $\text{rad}_t(e) < \frac{\Delta^{\text{C}}_e}{12} < \frac{\Delta^{\text{C}}_e}{4}$, we have

$$\begin{aligned}
\underline{w}_t(e) \geq & w(e) - 2\text{rad}_t(e) \\
> & \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M', \boldsymbol{w})) - \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w})) \\
= & \frac{1}{2}\texttt{MinW}(M', \boldsymbol{w}) + \frac{1}{2} \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w}) \\
\geq & \texttt{MinW}(M', \boldsymbol{w}),
\end{aligned}$$

which contradicts $\underline{w}_t(e) = \texttt{MinW}(M', \underline{\boldsymbol{w}}_t)$.

Case (ii) Suppose that $w(e) < \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M', \boldsymbol{w}))$. According to the definition of `BottleneckSearch` (Definition 2) and c_t, there exists a base arm $\tilde{e} \in \hat{B}_{\text{sub},t} \cap \{c_t\}$ satisfying $\tilde{e} \in M_*$ and $\underline{w}_t(\tilde{e}) = \texttt{MinW}(M_*, \underline{\boldsymbol{w}}_t)$.

First, we prove $\tilde{e} \in P^E_t$ and thus $\text{rad}_t(\tilde{e}) \leq \text{rad}_t(e) < \frac{\Delta^{\text{C}}_e}{12}$. If $M_* = M_t$, then $\tilde{e} = c_t$ and the claim holds. If $M_* \neq M_t$, then $\tilde{e} \in \hat{B}_{\text{sub},t}$. We can obtain that $\tilde{e}$ will be put into $\hat{B}'_{\text{sub},t} \subseteq P^E_t$. Otherwise, we have

$$w(\tilde{e}) \leq \bar{w}_t(\tilde{e}) \leq \frac{1}{2}(\texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t) + \underline{w}_t(\tilde{e})) \leq \frac{1}{2}(\texttt{MinW}(M_t, \boldsymbol{w}) + w(\tilde{e})).$$

Since $w(\tilde{e}) \geq \texttt{MinW}(M_*, \boldsymbol{w})$ and $\texttt{MinW}(M_t, \boldsymbol{w}) < \texttt{MinW}(M_*, \boldsymbol{w})$, the above inequality cannot hold. Thus, we obtain that $\tilde{e}$ will be put into $\hat{B}'_{\text{sub},t} \subseteq P^E_t$ and thus $\text{rad}_t(\tilde{e}) \leq \text{rad}_t(e) < \frac{\Delta^{\text{C}}_e}{12}$.

Next, we discuss the following two cases: (a) $e = c_t$ and (b) $e \neq c_t$.

(a) if $e = c_t$, then $M_t \neq M_*$. Using $\mathrm{rad}_t(\tilde{e}) \leq \mathrm{rad}_t(e) < \frac{\Delta^{\mathrm{C}}_e}{12} < \frac{\Delta^{\mathrm{C}}_e}{2}$, we have

$$\begin{aligned}
\texttt{MinW}(M_*, \underline{\boldsymbol{w}}_t) =& \underline{w}_t(\tilde{e}) \\
\geq& w(e) - 2\mathrm{rad}_t(e) \\
>& w(e) - (\texttt{MinW}(M_*, \boldsymbol{w}) - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w})) \\
\geq& \texttt{MinW}(M_*, \boldsymbol{w}) - (\texttt{MinW}(M_*, \boldsymbol{w}) - \texttt{MinW}(M_t, \boldsymbol{w})) \\
\geq& \texttt{MinW}(M_t, \boldsymbol{w}) \\
\geq& \texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t),
\end{aligned}$$

which contradicts the definition of M_t.

(b) if $e \neq c_t$, i.e., $e \in \hat{B}'_{\mathrm{sub},t}$ we have

$$\begin{aligned}
\Delta^{\mathrm{C}}_{\tilde{e},e} =& w(\tilde{e}) - w(e) \\
>& \texttt{MinW}(M_*, \boldsymbol{w}) - \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M', \boldsymbol{w})) \\
=& \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) - \texttt{MinW}(M', \boldsymbol{w}))
\end{aligned}$$

Since $\mathrm{rad}_t(e) < \frac{\Delta^{\mathrm{C}}_e}{12} = \frac{1}{12}(\texttt{MinW}(M_*, \boldsymbol{w}) - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w})) \leq \frac{1}{12}(\texttt{MinW}(M_*, \boldsymbol{w}) - \texttt{MinW}(M', \boldsymbol{w})) < \frac{1}{6}\Delta^{\mathrm{C}}_{\tilde{e},e}$ and $\mathrm{rad}_t(\tilde{e}) \leq \mathrm{rad}_t(e) < \frac{1}{6}\Delta^{\mathrm{C}}_{\tilde{e},e}$, according to Lemma 7, we have

$$\begin{aligned}
\bar{w}_t(e) <& \frac{1}{2}(\underline{w}_t(\tilde{e}) + \underline{w}_t(e)) \\
=& \frac{1}{2}(\texttt{MinW}(M_*, \underline{\boldsymbol{w}}_t) + \underline{w}_t(e)) \\
\leq& \frac{1}{2}(\texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t) + \underline{w}_t(e'))
\end{aligned}$$

which contradicts the definition of $\hat{B}'_{\mathrm{sub},t}$. □

Theorem 5 (Sample Complexity of `BLUCB-Explore`). *With probability at least $1 - \kappa$, the* `BLUCB-Explore` *algorithm (Algorithm 4) will return M_* with sample complexity*

$$O\left(\sum_{e \in [n]} \frac{R^2}{(\Delta^{\mathrm{C}}_e)^2} \ln \left(\sum_{e \in [n]} \frac{R^2 n}{(\Delta^{\mathrm{C}}_e)^2 \kappa} \right) \right).$$

Proof. For any $e \in [n]$, let $T(e)$ denote the number of samples for base arm e, and t_e denote the last timestep at which e is pulled. Then, we have $T_{t_e} = T(e) - 1$. Let T denote the total number of samples. According to Lemmas 8-10, we have

$$R\sqrt{\frac{2\ln(\frac{4nt_e^3}{\kappa})}{T(e)-1}} \geq \frac{1}{12}\Delta^{\mathrm{C}}_e$$

Thus, we obtain

$$T(e) \leq \frac{288R^2}{(\Delta^{\mathrm{C}}_e)^2} \ln\left(\frac{4nt_e^3}{\kappa}\right) + 1 \leq \frac{288R^2}{(\Delta^{\mathrm{C}}_e)^2} \ln\left(\frac{4nT^3}{\kappa}\right) + 1$$

Summing over $e \in [n]$, we have

$$T \leq \sum_{e \in [n]} \frac{288R^2}{(\Delta^{\mathrm{C}}_e)^2} \ln\left(\frac{4nT^3}{\kappa}\right) + n \leq \sum_{e \in [n]} \frac{864R^2}{(\Delta^{\mathrm{C}}_e)^2} \ln\left(\frac{2nT}{\kappa}\right) + n,$$

where $\sum_{e \in [n]} \frac{R^2}{(\Delta^{\mathrm{C}}_e)^2} \geq n$. Then, applying Lemma 20, we have

$$T \leq \sum_{e \in [n]} \frac{5184R^2}{(\Delta^{\mathrm{C}}_e)^2} \ln\left(\frac{2n^2}{\kappa} \sum_{e \in [n]} \frac{864R^2}{(\Delta^{\mathrm{C}}_e)^2}\right) + n$$

$$
\begin{aligned}
=&O\left(\sum_{e\in[n]}\frac{R^2}{(\Delta^{\mathrm{C}}_e)^2}\ln\left(\sum_{e\in[n]}\frac{R^2n^2}{(\Delta^{\mathrm{C}}_e)^2\kappa}\right)+n\right)\\
=&O\left(\sum_{e\in[n]}\frac{R^2}{(\Delta^{\mathrm{C}}_e)^2}\ln\left(\sum_{e\in[n]}\frac{R^2n}{(\Delta^{\mathrm{C}}_e)^2\kappa}\right)\right).
\end{aligned}
$$

□

Lemma 11. *For algorithm* `BLUCB-Verify`*, assume that event* $\xi_0\cap\xi$ *occurs. For any* $e\in M_*$*, if* $\mathrm{rad}_t(e)<\frac{\Delta^{\mathrm{C}}_e}{8}=\frac{1}{8}(w(e)-\max_{M\neq M_*}\texttt{MinW}(M,\boldsymbol{w}))$*, then, base arm* e *will not be pulled at round* t*, i.e.,* $p_t\neq e$*.*

Proof. Suppose that for some $e\in M_*$, $\mathrm{rad}_t(e)<\frac{\Delta^{\mathrm{C}}_e}{8}=\frac{1}{8}(w(e)-\max_{M\neq M_*}\texttt{MinW}(M,\boldsymbol{w}))$ and $p_t=e$.

Since event ξ_0 occurs, according to Lemma 6, we have that (i) $\hat{M}_*=M_*$, (ii) for any $M\neq M_*$, there exists a base arm $e\in\hat{B}_{\mathrm{sub}}\cap M$ satisfying $w(e)\leq\frac{1}{2}(\texttt{MinW}(M_*,\boldsymbol{w})+\texttt{MinW}(M,\boldsymbol{w}))$, i.e., $\Delta^{\mathrm{C}}_{\texttt{MinW}(M_*,\boldsymbol{w}),e}\geq\frac{1}{2}\Delta^{\mathrm{C}}_{M_*,M}$ and (iii) for any $i\in\hat{B}_{\mathrm{sub}}$, there exists a sub-optimal super arm $M\neq M_*$ such that $i\in M$ and $w(i)\leq\frac{1}{2}(\texttt{MinW}(M_*,\boldsymbol{w})+\texttt{MinW}(M))\leq\frac{1}{2}(\texttt{MinW}(M_*,\boldsymbol{w})+\texttt{MinW}(M_{\mathrm{second}},\boldsymbol{w}))$. According to the selection strategy of p_t, we have that for any $i\in P^V_t$, $\mathrm{rad}_t(i)\leq\mathrm{rad}_t(e)<\frac{\Delta^{\mathrm{C}}_e}{8}$.

First, since for any $i\in\hat{B}_{\mathrm{sub}}$, $w(i)\leq\frac{1}{2}(\texttt{MinW}(M_*,\boldsymbol{w})+\texttt{MinW}(M_{\mathrm{second}},\boldsymbol{w}))<\texttt{MinW}(M_*,\boldsymbol{w})$ and $w(e)\geq\texttt{MinW}(M_*,\boldsymbol{w})$, we can obtain that $e=c_t$.

Then, for any $i\in F_t$, we have

$$
\begin{aligned}
\Delta^{\mathrm{C}}_{e,i}=&w(e)-w(i)\\
\geq&w(e)-\frac{1}{2}(\texttt{MinW}(M_*,\boldsymbol{w})+\texttt{MinW}(M_{\mathrm{second}},\boldsymbol{w}))\\
\geq&w(e)-\frac{1}{2}(w(e)+\texttt{MinW}(M_{\mathrm{second}},\boldsymbol{w}))\\
=&\frac{1}{2}(w(e)-\texttt{MinW}(M_{\mathrm{second}},\boldsymbol{w}))\\
=&\frac{1}{2}\Delta^{\mathrm{C}}_e
\end{aligned}
$$

and

$$
\begin{aligned}
\underline{w}_t(c_t)-\bar{w}_t(i)=&\underline{w}_t(e)-\bar{w}_t(i)\\
\geq&w(e)-2\mathrm{rad}_t(e)-(w(i)+2\mathrm{rad}_t(i))\\
=&\Delta^{\mathrm{C}}_{e,i}-2\mathrm{rad}_t(e)-2\mathrm{rad}_t(i)\\
>&\frac{1}{2}\Delta^{\mathrm{C}}_e-\frac{\Delta^{\mathrm{C}}_e}{4}-\frac{\Delta^{\mathrm{C}}_e}{4}\\
=&0,
\end{aligned}
$$

which implies $F_t=\varnothing$. Thus, for any $i\in\hat{B}_{\mathrm{sub}}$, $\underline{w}_t(c_t)\geq\bar{w}_t(i)$.

According to Lemma 6(ii), for any $M'\neq\hat{M}_*$, there exists a base arm $e'\in\hat{B}_{\mathrm{sub}}\cap M'$, we have

$$
\begin{aligned}
\texttt{MinW}(\hat{M}_*,\underline{\boldsymbol{w}}_t)=&\underline{w}_t(c_t)\\
\geq&\bar{w}_t(e')\\
\geq&\texttt{MinW}(M',\bar{\boldsymbol{w}}_t)
\end{aligned}
$$

which implies that algorithm 3 has already stopped. □

Lemma 12. *For algorithm* `BLUCB-Verify`*, assume that event* $\xi_0\cap\xi$ *occurs. For any* $e\notin M_*$*,* $w(e)<\texttt{MinW}(M_*,\boldsymbol{w})$*, if* $\mathrm{rad}_t(e)<\frac{\Delta^{\mathrm{C}}_e}{8}=\frac{1}{8}(\texttt{MinW}(M_*,\boldsymbol{w})-\max_{M\in\mathcal{M}:e\in M}\texttt{MinW}(M,\boldsymbol{w}))$*, then, base arm* e *will not be pulled at round* t*, i.e.,* $p_t\neq e$*.*

Proof. Suppose that for some $e \notin M_*, w(e) < \texttt{MinW}(M_*, \boldsymbol{w})$, $\text{rad}_t(e) < \frac{\Delta^{\texttt{C}}_e}{8} = \frac{1}{8}(\texttt{MinW}(M_*, \boldsymbol{w}) - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w}))$ and $p_t = e$.

Since event ξ_0 occurs, according to Lemma 6, we have that (i) $\hat{M}_* = M_*$, (ii) for any $M \neq M_*$, there exists a base arm $e \in \hat{B}_{\text{sub}} \cap M$ satisfying $w(e) \leq \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M, \boldsymbol{w}))$, i.e., $\Delta^{\texttt{C}}_{\texttt{MinW}(M_*, \boldsymbol{w}), e} \geq \frac{1}{2}\Delta^{\texttt{C}}_{M_*, M}$ and (iii) for any $i \in \hat{B}_{\text{sub}}$, there exists a sub-optimal super arm $M \neq M_*$ such that $i \in M$ and $w(i) \leq \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M)) \leq \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M_{\text{second}}, \boldsymbol{w}))$. According to the selection strategy of p_t, we have that for any $i \in P^V_t$, $\text{rad}_t(i) \leq \text{rad}_t(e) < \frac{\Delta^{\texttt{C}}_e}{8}$.

Since $e \notin M_* = \hat{M}_*$, we have that $e \in F_t$ and there exists a sub-optimal super arm M' such that $e \in M'$ and $w(e) \leq \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M'))$. Then, we have

$$\begin{aligned}
\underline{w}_t(c_t) - \bar{w}_t(e) \geq & w(c_t) - 2\text{rad}_t(c_t) - (w(e) + 2\text{rad}_t(e)) \\
\geq & \texttt{MinW}(M_*, \boldsymbol{w}) - \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) + \texttt{MinW}(M', \boldsymbol{w})) - 2\text{rad}_t(c_t) - 2\text{rad}_t(e) \\
> & \frac{1}{2}\Delta^{\texttt{C}}_{M_*, M'} - \frac{\Delta^{\texttt{C}}_e}{4} - \frac{\Delta^{\texttt{C}}_e}{4} \\
= & \frac{1}{2}\Delta^{\texttt{C}}_{M_*, M'} - \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) - \max_{M \in \mathcal{M}: e \in M} \texttt{MinW}(M, \boldsymbol{w})) \\
\geq & \frac{1}{2}\Delta^{\texttt{C}}_{M_*, M'} - \frac{1}{2}(\texttt{MinW}(M_*, \boldsymbol{w}) - \texttt{MinW}(M', \boldsymbol{w})) \\
= & 0,
\end{aligned}$$

which contradicts $e \in F_t$. □

Recall that $B = \{e \mid e \notin M_*, w(e) < \texttt{MinW}(M_*, \boldsymbol{w})\}$ and $B^c = \{e \mid e \notin M_*, w(e) \geq \texttt{MinW}(M_*, \boldsymbol{w})\}$.

Theorem 6 (Sample Complexity of `BLUCB-Verify`)**.** *With probability at least* $1 - \kappa - \delta^V$*, the* `BLUCB-Verify` *algorithm (Algorithm 3) will return* M_* *with sample complexity*

$$O\left(\sum_{e \in B^c} \frac{R^2}{(\Delta^{\texttt{C}}_e)^2} \ln \left(\sum_{e \in B^c} \frac{R^2 n}{(\Delta^{\texttt{C}}_e)^2} \right) + \sum_{e \in M_* \cup B} \frac{R^2}{(\Delta^{\texttt{C}}_e)^2} \ln \left(\sum_{e \in M_* \cup B} \frac{R^2 n}{(\Delta^{\texttt{C}}_e)^2 \delta^V} \right) \right).$$

Proof. Assume that event $\xi_0 \cap \xi$ occurs. $\Pr[\xi_0 \cap \xi] \geq 1 - \kappa - \delta^V$.

First, we prove the correctness. According to Lemma 6, the hypothesized $\hat{M}_*$ outputted by the preparation procedure `BLUCB-Explore` is exactly the optimal super arm, and thus if `BLUCB-Verify` terminates, it returns the correct answer.

Next, we prove the sample complexity upper bound. According to Theorem 5, the preparation procedure `BLUCB-Explore` costs sample complexity

$$O\left(\sum_{e \in [n]} \frac{R^2}{(\Delta^{\texttt{C}}_e)^2} \ln \left(\sum_{e \in [n]} \frac{R^2 n}{(\Delta^{\texttt{C}}_e)^2 \kappa} \right) \right).$$

Then, we bound the sample complexity of the following verification part. Following the analysis procedure of Theorem 5 with Lemmas 11,12, we can obtain that the verification part cost sample complexity

$$\sum_{e \in M_* \cup B} \frac{R^2}{(\Delta^{\texttt{C}}_e)^2} \ln \left(\sum_{e \in M_* \cup B} \frac{R^2 n}{(\Delta^{\texttt{C}}_e)^2 \delta^V} \right).$$

Combining both parts, we obtain that the sample complexity is bounded by

$$O\left(\sum_{e \in B^c} \frac{R^2}{(\Delta^{\texttt{C}}_e)^2} \ln \left(\sum_{e \in B^c} \frac{R^2 n}{(\Delta^{\texttt{C}}_e)^2 \kappa} \right) + \sum_{e \in M_* \cup B} \frac{R^2}{(\Delta^{\texttt{C}}_e)^2} \ln \left(\sum_{e \in M_* \cup B} \frac{R^2 n}{(\Delta^{\texttt{C}}_e)^2 \delta^V} \right) \right)$$

$$=O\left(\sum_{e\in B^c}\frac{R^2}{(\Delta_e^{\mathrm{C}})^2}\ln\left(\sum_{e\in B^c}\frac{R^2 n}{(\Delta_e^{\mathrm{C}})^2}\right)+\sum_{e\in M_*\cup B}\frac{R^2}{(\Delta_e^{\mathrm{C}})^2}\ln\left(\sum_{e\in M_*\cup B}\frac{R^2 n}{(\Delta_e^{\mathrm{C}})^2\delta^V}\right)\right).$$

□

Lemma 13 (Correctness of `BLUCB-Verify`)**.** *With probability at least* $1-\delta$*, if algorithm* `BLUCB-Verify` *(Algorithm 3) terminates, it returns the optimal super arm* M_**.*

Proof. Assume that event ξ occurs, where $\Pr[\xi]\geq 1-\delta^V$. If algorithm `BLUCB-Verify` terminates by returning $\hat{M}_*$, we have that for any $M\in\mathcal{M}\setminus\mathcal{S}(\hat{M}_*)$,

$$\texttt{MinW}(\hat{M}_*,\boldsymbol{w})\geq\texttt{MinW}(\hat{M}_*,\underline{\boldsymbol{w}}_t)\geq\texttt{MinW}(M,\bar{\boldsymbol{w}}_t)\geq\texttt{MinW}(M,\boldsymbol{w}).$$

For any $M\in\mathcal{S}(\hat{M}_*)$, according to the property of the bottleneck reward function, we have

$$\texttt{MinW}(\hat{M}_*,\boldsymbol{w})\geq\texttt{MinW}(M,\boldsymbol{w}).$$

Thus, we have $\texttt{MinW}(\hat{M}_*,\boldsymbol{w})\geq\texttt{MinW}(M,\boldsymbol{w})$ for any $M\neq\hat{M}_*$ and according to the unique assumption of M_*, we obtain $\hat{M}_*=M_*$. In other words, algorithm `BLUCB-Verify` (Algorithm 3) will never return a wrong answer. □

Now, we prove Theorem 2.

Proof. Using Theorem 6, Lemma 13 and Lemma 4.8 in [10], we can obtain this theorem. □

B.3 PAC Learning

In this subsection, we further study the fixed-confidence CPE-B problem in the PAC learning setting, where the learner's objective is to identify a super arm M_{pac} such that $\texttt{MinW}(M_{\text{pac}},\boldsymbol{w})\geq\texttt{OPT}-\varepsilon$, and the uniqueness assumption of the optimal super arm is dropped. We propose two algorithms `BLUCB-PAC` and `BLUCB-Parallel-PAC` for the PAC learning setting, based on `BLUCB` and `BLUCB-Parallel`, respectively.

The PAC algorithms and their theoretical guarantees do not require the uniqueness assumption of the optimal super arm. Compared to the PAC lower bound, both proposed PAC algorithms achieve the optimal sample complexity for some family of instances. Similar to the exact case, when δ is small enough, the dominant term of sample complexity for algorithm `BLUCB-Parallel-PAC` does not depend on the reward gaps of unnecessary base arms, and thus `BLUCB-Parallel-PAC` achieves better theoretical guarantee than `BLUCB-PAC` and matches the lower bound for a broader family of instances.

B.3.1 Algorithm `BLUCB-PAC`

`BLUCB-PAC` simply replaces the stopping condition of `BLUCB` (Line 9 in Algorithm 1) with $\texttt{MinW}(\tilde{M}_t,\bar{\boldsymbol{w}}_t)-\texttt{MinW}(M_t,\underline{\boldsymbol{w}}_t)\leq\varepsilon$ to allow an ε deviation between the returned answer and the optimal one. The sample complexity of algorithm `BLUCB-PAC` is given as follows.

Theorem 7 (Fixed-confidence Upper Bound for PAC)**.** *With probability at least* $1-\delta$*, the* `BLUCB-PAC` *algorithm will return* M_{out} *such that* $\texttt{MinW}(M_{\text{out}},\boldsymbol{w})\geq\texttt{MinW}(M_*,\boldsymbol{w})-\varepsilon$*, with sample complexity*

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

Now we prove the sample complexity of algorithm `BLUCB-PAC` (Theorem 7).

Proof. First, we prove the correctness. When the stop condition of `BLUCB-PAC` is satisfied, we have that for any $M\neq M_t$,

$$\texttt{MinW}(M,\boldsymbol{w})-\texttt{MinW}(M_t,\boldsymbol{w})\leq\texttt{MinW}(M,\bar{\boldsymbol{w}}_t)-\texttt{MinW}(M_t,\underline{\boldsymbol{w}}_t)\leq\texttt{MinW}(\tilde{M}_t,\bar{\boldsymbol{w}}_t)-\texttt{MinW}(M_t,\underline{\boldsymbol{w}}_t)\leq\varepsilon.$$

If $M_t = M_*$, then the correctness holds. If $M_t \neq M_*$, the returned super arm M_t satisfies

$$\texttt{MinW}(M_*, \boldsymbol{w}) - \texttt{MinW}(M_t, \boldsymbol{w}) \leq \varepsilon,$$

which guarantees the correctness.

Next, we prove the sample complexity. When inheriting the proof of Theorem 1 for the baseline algorithm `BLUCB`, to prove Theorem 7 for PAC leaning, it suffices to prove that for any $e \in [n]$, if $\text{rad}_t(e) < \frac{\varepsilon}{2}$, base arm e will not be pulled at round t, i.e., $p_t \neq e$.

Suppose that $\text{rad}_t(e) < \frac{\varepsilon}{2}$ and $p_t = e$. According to the selection strategy of p_t, we have $\text{rad}_t(c_t) < \frac{\varepsilon}{2}$ and $\text{rad}_t(d_t) < \frac{\varepsilon}{2}$. According to the definition of d_t, we have

$$\begin{aligned}
\texttt{MinW}(\tilde{M}_t, \bar{\boldsymbol{w}}_t) - \texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t) \leq & \bar{w}(d_t) - \texttt{MinW}(\tilde{M}_t, \underline{\boldsymbol{w}}_t) \\
= & \bar{w}(d_t) - \underline{w}(d_t) \\
= & 2\text{rad}_t(d_t) \\
< & \varepsilon,
\end{aligned}$$

which contradicts the stop condition. □

B.3.2 Algorithm `BLUCB-Parallel-PAC`

`BLUCB-Parallel-PAC` is obtained by simply replacing the stopping condition of `BLUCB-Verify` (Line 10 in Algorithm 3) with $\texttt{MinW}(\tilde{M}_t, \bar{\boldsymbol{w}}_t) - \texttt{MinW}(M_t, \underline{\boldsymbol{w}}_t) \leq \varepsilon$.

Theorem 8 presents the sample complexity of algorithm `BLUCB-Parallel-PAC`.

Theorem 8 (Improved Fixed-confidence Upper Bound for PAC)**.** *For any $\delta < 0.01$, with probability at least $1 - \delta$, algorithm* `BLUCB-Parallel-PAC` *returns M_* and takes the expected sample complexity*

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

Proof. First, we prove the correctness. When the stop condition of `BLUCB-Verify` is satisfied, we have that for any $M \neq \hat{M}_*$,

$$\texttt{MinW}(M, \boldsymbol{w}) - \texttt{MinW}(\hat{M}_*, \boldsymbol{w}) \leq \texttt{MinW}(M, \bar{\boldsymbol{w}}_t) - \texttt{MinW}(\hat{M}_*, \underline{\boldsymbol{w}}_t) \leq \texttt{MinW}(\tilde{M}_t, \bar{\boldsymbol{w}}_t) - \texttt{MinW}(\hat{M}_*, \underline{\boldsymbol{w}}_t) \leq \varepsilon.$$

If $\hat{M}_* = M_*$, then the correctness holds. If $\hat{M}_* \neq M_*$, the returned super arm $\hat{M}_*$ satisfies

$$\texttt{MinW}(M_*, \boldsymbol{w}) - \texttt{MinW}(\hat{M}_*, \boldsymbol{w}) \leq \varepsilon,$$

which guarantees the correctness.

Next, we prove the sample complexity. We inherit the proofs of Theorems 2,6. Then, to prove Theorem 8 for PAC leaning, it suffices to prove that conditioning on $\xi_0 \cap \xi$, for any $e \in [n]$, if $\text{rad}_t(e) < \frac{\varepsilon}{4}$, base arm e will not be pulled at round t, i.e., $p_t \neq e$.

Suppose that $\text{rad}_t(e) < \frac{\varepsilon}{4}$ and $p_t = e$. According to the selection strategy of p_t, we have $\text{rad}_t(c_t) < \frac{\varepsilon}{4}$ and for any $e \in F_t$ $\text{rad}_t(e) < \frac{\varepsilon}{4}$. Using $F_t \subseteq \hat{B}_{\text{sub}}$ and the definition of $\hat{B}_{\text{sub}}$, we have that for any $e \in F_t$

$$\begin{aligned}
\bar{w}(e) - \texttt{MinW}(\hat{M}_*, \underline{\boldsymbol{w}}_t) \leq & w(e) + 2\text{rad}_t(e) - (w(c_t) - 2\text{rad}_t(c_t)) \\
< & \texttt{MinW}(\hat{M}_*, \boldsymbol{w}) + 2\text{rad}_t(e) - (\texttt{MinW}(\hat{M}_*, \boldsymbol{w}) - 2\text{rad}_t(c_t)) \\
< & 4 \cdot \frac{\varepsilon}{4} \\
= & \varepsilon
\end{aligned}$$

and thus

$$\texttt{MinW}(\tilde{M}_t, \bar{\boldsymbol{w}}_t) - \texttt{MinW}(\hat{M}_*, \underline{\boldsymbol{w}}_t) \leq \varepsilon,$$

which contradicts the stop condition. □

C Lower Bounds for the Fixed-Confidence Setting

In this section, we present the proof of lower bound for the exact fixed-confidence CPE-B problem. Then, we also provide a lower bound for the PAC fixed-confidence CPE-B problem and give its proof.

First, we prove the lower bound for the exact fixed-confidence CPE-B problem (Theorem 3). Notice that, the sample complexity of algorithm `BLUCB` also matches the lower bound within a logarithmic factor if we replace condition (iii) below with that each sub-optimal super arm only has a single base arm.

Proof. Consider an instance $\mathcal{I}$ of the fixed-confidence CPE-B problem such that: (i) the reward distribution of each base arm $e \in [n]$ is $\mathcal{N}(w(e), R)$; (ii) both M_* and the second best super arms are unique, and the second best super arm has no overlapped base arm with M_*; (iii) in each sub-optimal super arm, there is a single base arm with reward below $\texttt{MinW}(M_*, \boldsymbol{w})$.

Fix an arbitrary δ-correct algorithm $\mathbb{A}$. For an arbitrary base arm $e \in M_*$, we construct an instance $\mathcal{I}'$ by changing its reward distribution to $\mathcal{N}(w'(e), R)$ where $w'(e) = w(e) - 2\Delta^{\mathrm{C}}_e$. Recall that $M_{\text{second}} = \operatorname{argmax}_{M \neq M_*} \texttt{MinW}(M, \boldsymbol{w})$. For instance $\mathcal{I}'$, from the definition of Δ^{C}_e (Definition 1),

$$\begin{aligned} w'(e) =& w(e) - 2\Delta^{\mathrm{C}}_e \\ =& w(e) - (w(e) - \texttt{MinW}(M_{\text{second}}, \boldsymbol{w})) - \Delta^{\mathrm{C}}_e \\ =& \texttt{MinW}(M_{\text{second}}, \boldsymbol{w}) - \Delta^{\mathrm{C}}_e \\ <& \texttt{MinW}(M_{\text{second}}, \boldsymbol{w}) \end{aligned}$$

and $\texttt{MinW}(M_*, \boldsymbol{w}') = w'(e) < \texttt{MinW}(M_{\text{second}}, \boldsymbol{w})$. Thus, M_{second} becomes the optimal super arm.

Let T_e denote the number of samples drawn from base arm e when algorithm $\mathbb{A}$ runs on instance $\mathcal{I}$. Let $d(x, y) = x \ln(x/y) + (1-x)\ln[(1-x)/(1-y)]$ denote the binary relative entropy function. Define $\mathcal{H}$ as the event that algorithm $\mathbb{A}$ returns M_*. Since $\mathbb{A}$ is δ-correct, we have $\Pr_{\mathbb{A},\mathcal{I}}[\mathcal{H}] \geq 1 - \delta$ and $\Pr_{\mathbb{A},\mathcal{I}'}[\mathcal{H}] \leq \delta$. Thus, $d(\Pr_{\mathbb{A},\mathcal{I}}[\mathcal{H}], \Pr_{\mathbb{A},\mathcal{I}'}[\mathcal{H}]) \geq d(1-\delta, \delta)$. Using Lemma 1 in [27], we can obtain

$$\mathbb{E}[T_e] \mathrm{KL}(\mathcal{N}(w(e), R), \mathcal{N}(w'(e), R)) \geq d(1-\delta, \delta),$$

Since the reward distribution of each base arm is Gaussian distribution, we have $\mathrm{KL}(\mathcal{N}(w(e), R), \mathcal{N}(w(e'), R)) = \frac{1}{2R^2}(w(e) - w'(e))^2 = \frac{2}{R^2}(\Delta^{\mathrm{C}}_e)^2$. Since $\delta \in (0, 0.1)$, $d(1-\delta, \delta) \geq 0.4 \ln(1/\delta)$. Thus, we have

$$\frac{2}{R^2}(\Delta^{\mathrm{C}}_e)^2 \cdot \mathbb{E}[T_e] \geq 0.4 \ln(\frac{1}{\delta}).$$

Then,

$$\mathbb{E}[T_e] \geq 0.2 \frac{R^2}{(\Delta^{\mathrm{C}}_e)^2} \ln(\frac{1}{\delta}).$$

For an arbitrary base arm $e \notin M_*, w(e) < \texttt{MinW}(M_*, \boldsymbol{w})$, we can construct another instance $\mathcal{I}'$ by changing its reward distribution to $\mathcal{N}(w'(e), R)$ where $w'(e) = w(e) + 2\Delta^{\mathrm{C}}_e$. Let M_e denote the sub-optimal super arm that contains e.

For instance $\mathcal{I}'$, from the definition of Δ^{C}_e (Definition 1),

$$\begin{aligned} w'(e) =& w(e) + 2\Delta^{\mathrm{C}}_e \\ =& w(e) + (\texttt{MinW}(M_*, \boldsymbol{w}) - \texttt{MinW}(M_e)) + \Delta^{\mathrm{C}}_e \\ =& w(e) + (\texttt{MinW}(M_*, \boldsymbol{w}) - w(e)) + \Delta^{\mathrm{C}}_e \\ =& \texttt{MinW}(M_*, \boldsymbol{w}) + \Delta^{\mathrm{C}}_e \\ >& \texttt{MinW}(M_*, \boldsymbol{w}). \end{aligned}$$

Thus, M_e become the optimal super arm. Similarly, using Lemma 1 in [27] we can obtain

$$\frac{2}{R^2}(\Delta^{\mathrm{C}}_e)^2 \cdot \mathbb{E}[T_e] \geq 0.4 \ln(\frac{1}{\delta}).$$

Algorithm 7 AR-Oracle

```
Input: decision class $\mathcal{M}$, accepted base arm $a$, set of the rejected base arms $R$ and weight vector $\boldsymbol{v}$.
Remove the base arms in $R$ from $\mathcal{M}$ and obtain a new decision class $\mathcal{M}_{-R}$.
Sort the remaining base arms by descending rewards and denote them by $e_{(1)}, \dots, e_{(n-|R|)}$
for $e = e_{(1)}, \dots, e_{(n-|R|)}$ do
    Remove all base arms with the rewards lower than $w(e)$ from $\mathcal{M}$ and obtain a new decision class $\mathcal{M}_{-R,\geq w(e)}$
    $M_{\text{out}} \leftarrow$ ExistOracle($\mathcal{M}_{-R,\geq w(e)}, a$)
    if $M_{\text{out}} \neq \perp$ then
        return $M_{\text{out}}$
    end if
end for
```

Then,

$$\mathbb{E}[T_e] \geq 0.2 \frac{R^2}{(\Delta_e^{\text{C}})^2} \ln(\frac{1}{\delta}).$$

Summing over all $e \in M_*$ and $e \notin M_*, w(e) < \texttt{MinW}(M_*, \boldsymbol{w})$, we can obtain that any δ-correct algorithm has sample complexity

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

□

Next, we present the lower bound for the PAC fixed-confidence CPE-B problem, where we can relax condition (ii) in the proof of the exact lower bound (Theorem 3). To formally state our result, we first introduce the notion of (δ, ε)-*correct algorithm* as follows. For any confidence parameter $\delta \in (0, 1)$ and accuracy parameter $\varepsilon > 0$, we call an algorithm $\mathcal{A}$ a (δ, ε)-correct algorithm if for the fixed-confidence CPE-B in PAC learning, $\mathcal{A}$ returns a super arm M_{pac} such that $\texttt{MinW}(M_{\text{pac}}, \boldsymbol{w}) \geq \texttt{MinW}(M_*, \boldsymbol{w}) - \varepsilon$ with probability at least $1 - \delta$.

Theorem 9 (Fixed-confidence Lower Bound for PAC)**.** *There exists a family of instances for the fixed-confidence CPE-B problem, where for any* $\delta \in (0, 0.1)$*, any* (δ, ε)*-correct algorithm has the expected sample complexity*

$$\Omega \left(\sum_{e \in M_* \cup B} \frac{R^2}{\max\{(\Delta_e^{\text{C}})^2, \varepsilon^2\}} \ln \left(\frac{1}{\delta} \right) \right).$$

Proof. Consider the instance $\mathcal{I}$ for the PAC fixed-confidence CPE-B problem, where $\varepsilon < \texttt{MinW}(M_*, \boldsymbol{w}) - \texttt{MinW}(M_{\text{second}}, \boldsymbol{w})$ (to guarantee that M_* is unique) and (i) the reward distribution of each base arm $e \in [n]$ is $\mathcal{N}(w(e), 1)$; (ii) the PAC solution M_{pac} is unique and the second best super arm has no overlapped base arm with M_{pac}; (iii) in each sub-optimal super arm, there is a single base arm with reward below $\texttt{MinW}(M_*, \boldsymbol{w})$. Then, following the proof procedure of Theorem 3, we can obtain Theorem 9. □

D CPE-B in the Fixed-Budget Setting

In this section, we present the implementation details of AR-Oracle and error probability proof for algorithm BSAR.

D.1 Implementation Details of AR-Oracle

First, we discuss AR-Oracle. Recall that $\texttt{AR-Oracle} \in \operatorname{argmax}_{M \in \mathcal{M}(e,R)} \texttt{MinW}(M, \boldsymbol{w})$, where $\mathcal{M}(e, R) = \{M \in \mathcal{M} : e \in M, R \cap M = \varnothing\}$. If $\mathcal{M}(e, R) = \varnothing$, $\texttt{AR-Oracle} = \perp$. Algorithm 7 gives the algorithm pseudo-code of AR-Oracle. As BottleneckSearch, AR-Oracle also uses the existence oracle ExistOracle to find a feasible super arm that contains some base arm from the given decision class if there exists, and otherwise return $\perp$. We explain the procedure of AR-Oracle as follows: we first remove all base arms in R from the decision class $\mathcal{M}$ to disable the super arms that contain the rejected base arms. Then, we enumerate the remaining base arms by descending rewards. For each enumerated base arm e, we remove the base arms with rewards lower than $w(e)$ and obtain a new decision class $\mathcal{M}_{-R,\geq w(e)}$, and then use ExistOracle to find a feasible super arm that contains the accepted base arm a from $\mathcal{M}_{-R,\geq w(e)}$. Once such a feasible super arm is found, the procedure terminates and returns this super arm. Since the enumeration of base arms is performed according to descending rewards and the computed decision class only contains base arms no worse than the enumerated one, AR-Oracle guarantees to return an optimal super arm from $\mathcal{M}(e, R)$.

As for the computational efficiency, the time complexity of AR-Oracle mainly depends on the step of finding a feasible super arm containing some base arm e. Fortunately, this existence problem can be solved in polynomial time for a wide family of decision classes. For example, for s-t paths, this problem can be reduced to the well-studied 2-vertex connectivity problem [20], which is polynomially tractable (see Section E.1 for the proof of reduction). For maximum cardinality matchings, we just need to remove e and its two end vertices, and then find a feasible maximum cardinality matching in the remaining graph; and for spanning trees, we can just merge the vertices of e and find a feasible spanning tree in the remaining graph. All of the above cases can be solved efficiently.

D.2 Proof for Algorithm BSAR

Below we present the proof of error probability for algorithm BSAR. To prove Theorem 4, we first introduce the lowing Lemmas 14-17.

Lemma 14. *For phase $t = 1, \dots, n$, define events*

$$\mathcal{E}_t = \left\{ \forall i \in [n] \setminus (A_t \cup R_t),\ |\hat{w}_t(i) - w(i)| < \frac{\Delta^{\mathrm{B}}_{(n+1-t)}}{8} \right\}.$$

and $\mathcal{E} \triangleq \bigcap_{t=1}^{n} \mathcal{E}_t$. Then, we have

$$\Pr\left[\mathcal{E}\right] \geq 1 - 2n^2 \exp\left(- \frac{(T-n)}{128 \widetilde{\log}(n) R^2 H^B} \right).$$

Proof. For any $t \in [n]$ and $e \in [n] \setminus (A_t \cup R_t)$, according to the Hoeffding's inequality,

$$\left\{ |\hat{w}_t(i) - w(i)| \geq \frac{\Delta^{\mathrm{B}}_{(n+1-t)}}{8} \right\} \leq 2 \exp\left(- \frac{\tilde{T}(\Delta^{\mathrm{B}}_{n-t+1})^2}{128 R^2} \right).$$

From the definition of $\tilde{T}$ and H^B, we have

$$\begin{aligned}
\Pr\left[|\hat{w}_t(i) - w(i)| \geq \frac{\Delta^{\mathrm{B}}_{(n+1-t)}}{8} \right] \leq & 2\exp\left(- \frac{\tilde{T}(\Delta^{\mathrm{B}}_{n-t+1})^2}{128R^2} \right) \\
\leq & 2\exp\left(- \frac{\frac{T-n}{\widetilde{\log}(n)(n-t+1)}(\Delta^{\mathrm{B}}_{n-t+1})^2}{128R^2} \right) \\
= & 2\exp\left(- \frac{(T-n)}{128\widetilde{\log}(n)R^2 \frac{n-t+1}{(\Delta^{\mathrm{B}}_{n-t+1})^2}} \right) \\
\leq & 2\exp\left(- \frac{(T-n)}{128\widetilde{\log}(n)R^2 H^B} \right)
\end{aligned}$$

By a union bound over $t \in [n]$ and $e \in [n] \setminus (A_t \cup R_t)$, we have

$$\Pr[\mathcal{E}] \geq 1 - n^2 \Pr\left[|\hat{w}_t(i) - w(i)| \geq \frac{\Delta^{\mathrm{B}}_{(n+1-t)}}{8} \right]$$

$$\geq 1 - 2n^2 \exp\left(- \frac{(T-n)}{128\tilde{\log}(n) R^2 H^B} \right).$$

□

Lemma 15. *Fix any phase $t > 0$. Assume that event $\mathcal{E}_t$ occurs and algorithm* `BSAR` *does not make any mistake before phase t, i.e., $A_t \subseteq M_*$ and $R_t \cap M_* = \varnothing$. Then, for any $e \in [n] \setminus (A_t \cup R_t)$ s.t. $\Delta^{\mathtt{B}}_e \geq \Delta^{\mathtt{B}}_{(n+1-t)}$, we have $e \in (M_* \cap M_t) \cup (\neg M_* \cap \neg M_t)$.*

Proof. Suppose that $e \in (M_* \cap \neg M_t) \cup (\neg M_* \cap M_t)$.

Case (I). If $e \in M_*, e \notin M_t$, then M_t is a sub-optimal super arm and $\Delta^{\mathtt{B}}_{M_*, M_t} \geq \Delta^{\mathtt{B}}_e \geq \Delta^{\mathtt{B}}_{(n+1-t)}$. Then, we have

$$\begin{aligned}
\mathtt{MinW}(M_*, \hat{\boldsymbol{w}}_t) - \mathtt{MinW}(M_t, \hat{\boldsymbol{w}}_t) >& \mathtt{MinW}(M_*, \boldsymbol{w}) - \frac{1}{8}\Delta^{\mathtt{B}}_{(n+1-t)} - (\mathtt{MinW}(M_t, \boldsymbol{w}) + \frac{1}{8}\Delta^{\mathtt{B}}_{(n+1-t)}) \\
=& \mathtt{MinW}(M_*, \boldsymbol{w}) - \mathtt{MinW}(M_t, \boldsymbol{w}) - \frac{1}{4}\Delta^{\mathtt{B}}_{(n+1-t)} \\
\geq& \Delta^{\mathtt{B}}_e - \frac{1}{4}\Delta^{\mathtt{B}}_{(n+1-t)} \\
\geq& \frac{3}{4}\Delta^{\mathtt{B}}_{(n+1-t)} \\
>& 0,
\end{aligned}$$

which contradicts the definition of M_t.

Case (II). If $e \in M_t, e \notin M_*$, then M_t is a sub-optimal super arm and $\Delta^{\mathtt{B}}_{M_*, M_t} \geq \Delta^{\mathtt{B}}_e \geq \Delta^{\mathtt{B}}_{(n+1-t)}$. Then, we have

$$\begin{aligned}
\mathtt{MinW}(M_*, \hat{\boldsymbol{w}}_t) - \mathtt{MinW}(M_t, \hat{\boldsymbol{w}}_t) >& \mathtt{MinW}(M_*, \boldsymbol{w}) - \frac{1}{8}\Delta^{\mathtt{B}}_{(n+1-t)} - (\mathtt{MinW}(M_t, \boldsymbol{w}) + \frac{1}{8}\Delta^{\mathtt{B}}_{(n+1-t)}) \\
=& \mathtt{MinW}(M_*, \boldsymbol{w}) - \mathtt{MinW}(M_t, \boldsymbol{w}) - \frac{1}{4}\Delta^{\mathtt{B}}_{(n+1-t)} \\
\geq& \Delta^{\mathtt{B}}_e - \frac{1}{4}\Delta^{\mathtt{B}}_{(n+1-t)} \\
\geq& \frac{3}{4}\Delta^{\mathtt{B}}_{(n+1-t)} \\
>& 0,
\end{aligned}$$

which contradicts the definition of M_t.

Thus, the supposition does not hold and we obtain $e \in (M_* \cap M_t) \cup (\neg M_* \cap \neg M_t)$. □

Lemma 16. *Fix any phase $t > 0$. Assume that event $\mathcal{E}_t$ occurs and algorithm* `BSAR` *does not make any mistake before phase t, i.e., $A_t \subseteq M_*$ and $R_t \cap M_* = \varnothing$. Then, there exists some base arm $e \in [n] \setminus (A_t \cup R_t)$ s.t. $\Delta^{\mathtt{B}}_e \geq \Delta^{\mathtt{B}}_{(n+1-t)}$ and this base arm e satisfies*

$$\mathtt{MinW}(M_t, \hat{\boldsymbol{w}}_t) - \mathtt{MinW}(\tilde{M}_{t,e}, \hat{\boldsymbol{w}}_t) > \frac{3}{4}\Delta^{\mathtt{B}}_{(n+1-t)}.$$

Proof. Since $e \in [n] \setminus (A_t \cup R_t)$ and $\Delta^{\mathtt{B}}_e \geq \Delta^{\mathtt{B}}_{(n+1-t)}$, according to Lemma 15, we have $e \in (M_* \cap M_t) \cup (\neg M_* \cap \neg M_t)$.

Case (I). If $e \in (M_* \cap M_t)$, then $e \notin \tilde{M}_{t,e}$ (if $\tilde{M}_{t,e} = \perp$ then the lemma trivially holds) and $\Delta^{\mathtt{B}}_{M_*, \tilde{M}_{t,e}} \geq \Delta^{\mathtt{B}}_e$. We have

$$\begin{aligned}
\mathtt{MinW}(M_*, \hat{\boldsymbol{w}}_t) - \mathtt{MinW}(\tilde{M}_{t,e}, \hat{\boldsymbol{w}}_t) >& \mathtt{MinW}(M_*, \boldsymbol{w}) - \frac{1}{8}\Delta^{\mathtt{B}}_{(n+1-t)} - (\mathtt{MinW}(\tilde{M}_{t,e}, \boldsymbol{w}) + \frac{1}{8}\Delta^{\mathtt{B}}_{(n+1-t)}) \\
=& \mathtt{MinW}(M_*, \boldsymbol{w}) - \mathtt{MinW}(\tilde{M}_{t,e}, \boldsymbol{w}) - \frac{1}{4}\Delta^{\mathtt{B}}_{(n+1-t)} \\
\geq& \Delta^{\mathtt{C}}_e - \frac{1}{4}\Delta^{\mathtt{B}}_{(n+1-t)}
\end{aligned}$$

$$\geq \frac{3}{4}\Delta^{\texttt{B}}_{(n+1-t)}.$$

Case (II). If $e \in (\neg M_* \cap \neg M_t)$, then $e \in \tilde{M}_{t,e}$ (if $\tilde{M}_{t,e} = \perp$ then the lemma trivially holds) and $\Delta^{\texttt{B}}_{M_*,\tilde{M}_{t,e}} \geq \Delta^{\texttt{B}}_e$. We have

$$\begin{aligned}\texttt{MinW}(M_*, \hat{\boldsymbol{w}}_t) - \texttt{MinW}(\tilde{M}_{t,e}, \hat{\boldsymbol{w}}_t) >&\texttt{MinW}(M_*, \boldsymbol{w}) - \frac{1}{8}\Delta^{\texttt{B}}_{(n+1-t)} - (\texttt{MinW}(\tilde{M}_{t,e}, \boldsymbol{w}) + \frac{1}{8}\Delta^{\texttt{B}}_{(n+1-t)})\\ =&\texttt{MinW}(M_*, \boldsymbol{w}) - \texttt{MinW}(\tilde{M}_{t,e}, \boldsymbol{w}) - \frac{1}{4}\Delta^{\texttt{B}}_{(n+1-t)}\\ \geq&\Delta^{\texttt{C}}_e - \frac{1}{4}\Delta^{\texttt{B}}_{(n+1-t)}\\ \geq&\frac{3}{4}\Delta^{\texttt{B}}_{(n+1-t)}.\end{aligned}$$

Combining cases (I) and (II), we obtain the lemma. □

Lemma 17. *Fix any phase $t > 0$. Assume that event $\mathcal{E}_t$ occurs and algorithm* BSAR *does not make any mistake before phase t, i.e., $A_t \subseteq M_*$ and $R_t \cap M_* = \varnothing$. Then, for any $p \in [n] \setminus (A_t \cup R_t)$ s.t. $p \in (M_* \cap \neg M_t) \cup (\neg M_* \cap M_t)$, we have*

$$\texttt{MinW}(M_t, \hat{\boldsymbol{w}}_t) - \texttt{MinW}(\tilde{M}_{t,p}, \hat{\boldsymbol{w}}_t) < \frac{1}{4}\Delta^{\texttt{B}}_{(n+1-t)}$$

Proof. Since $p \in (M_* \cap \neg M_t) \cup (\neg M_* \cap M_t)$, then M_t is a sub-optimal super arm and $\Delta^{\texttt{B}}_{M_t,M_*} < 0$. We have

$$\begin{aligned}\texttt{MinW}(M_t, \hat{\boldsymbol{w}}_t) - \texttt{MinW}(M_*, \hat{\boldsymbol{w}}_t) <&\texttt{MinW}(M_t, \boldsymbol{w}) + \frac{1}{8}\Delta^{\texttt{B}}_{(n+1-t)} - (\texttt{MinW}(M_*, \boldsymbol{w}) - \frac{1}{8}\Delta^{\texttt{B}}_{(n+1-t)})\\ =&\texttt{MinW}(M_t, \boldsymbol{w}) - \texttt{MinW}(M_*, \boldsymbol{w}) + \frac{1}{4}\Delta^{\texttt{B}}_{(n+1-t)}\\ <&\frac{1}{4}\Delta^{\texttt{B}}_{(n+1-t)}.\end{aligned}$$

Since $p \in (M_* \cap \neg M_t) \cup (\neg M_* \cap M_t)$, according to the definition of $\tilde{M}_{t,p}$, we have $\texttt{MinW}(\tilde{M}_{t,p}, \hat{\boldsymbol{w}}_t) \geq \texttt{MinW}(M_*, \hat{\boldsymbol{w}}_t)$. Then, we have

$$\begin{aligned}\texttt{MinW}(M_t, \hat{\boldsymbol{w}}_t) - \texttt{MinW}(\tilde{M}_{t,p}, \hat{\boldsymbol{w}}_t) \leq&\texttt{MinW}(M_t, \hat{\boldsymbol{w}}_t) - \texttt{MinW}(M_*, \hat{\boldsymbol{w}}_t)\\ <&\frac{1}{4}\Delta^{\texttt{B}}_{(n+1-t)}.\end{aligned}$$

□

Now, we prove Theorem 4.

Proof. First, we prove that the number of samples for algorithm BSAR is bounded by T. Summing the number of samples for each phase, we have

$$\begin{aligned}\sum_{t=1}^{n} \tilde{T}_t \leq&\sum_{t=1}^{n} \left(\frac{T-n}{\tilde{\log}(n)(n-t+1)} + 1 \right)\\ =&\frac{T-n}{\tilde{\log}(n)}\tilde{\log}(n) + n\\ =&T.\end{aligned}$$

Next, we prove the mistake probability. According to Lemma 14, in order to prove Theorem 4, it suffices to prove that conditioning on $\mathcal{E}$, algorithm BSAR returns M_*.

Assuming that $\mathcal{E}$ occurs, we prove by induction. Fix a phase $t \in [n]$. Suppose that algorithm BSAR does not make any mistake before phase t, i.e., $A_t \subseteq M_*$ and $R_t \cap M_* = \varnothing$. We show that algorithm BSAR does not make any mistake in phase t either.

According to Lemma 16, there exists some base arm $e \in [n] \setminus (A_t \cup R_t)$ s.t. $\Delta^{\mathrm{B}}_e \geq \Delta^{\mathrm{B}}_{(n+1-t)}$ and this base arm e satisfies $\texttt{MinW}(M_t, \hat{\boldsymbol{w}}_t) - \texttt{MinW}(\tilde{M}_{t,e}, \hat{\boldsymbol{w}}_t) > \frac{3}{4}\Delta^{\mathrm{B}}_{(n+1-t)}$. Suppose that algorithm $\texttt{BSAR}$ makes a mistake in phase t, i.e., $p_t \in (M_* \cap \neg M_t) \cup (\neg M_* \cap M_t)$. According to Lemma 17, we have $\texttt{MinW}(M_t, \hat{\boldsymbol{w}}_t) - \texttt{MinW}(\tilde{M}_{t,p_t}, \hat{\boldsymbol{w}}_t) < \frac{1}{4}\Delta^{\mathrm{B}}_{(n+1-t)}$. Then,

$$\begin{aligned}\texttt{MinW}(M_t, \bar{\boldsymbol{w}}_t) - \texttt{MinW}(\tilde{M}_{t,e}, \bar{\boldsymbol{w}}_t) >& \frac{3}{4}\Delta^{\mathrm{B}}_{(n+1-t)} \\ >& \frac{1}{4}\Delta^{\mathrm{B}}_{(n+1-t)} \\ >& \texttt{MinW}(M_t, \bar{\boldsymbol{w}}_t) - \texttt{MinW}(\tilde{M}_{t,p_t}, \bar{\boldsymbol{w}}_t),\end{aligned}$$

which contradicts the selection strategy of p_t. Thus, $p_t \in (M_* \cap M_t) \cup (\neg M_* \cap \neg M_t)$, i.e., algorithm $\texttt{BSAR}$ does not make any mistake in phase t, which completes the proof. □

D.3 Exponential-time Complexity of the Accept-reject Oracle used in Prior Work [11]

The accept-reject oracle used in prior CPE-L work [11], which returns the optimal super arm with a given base arm set A_t contained in it, costs exponential running time on s-t path instances. This is because the problem $\mathcal{P}$ of finding an s-t path which contains a given edge set is NP-hard. In the following, we prove the NP-hardness of problem $\mathcal{P}$ by building a reduction from the Hamiltonian Path Problem [22] to $\mathcal{P}$.

Proof. Given any Hamiltonian Path instance $G = (V, E)$ with start and end nodes $s, t \in V$, we need to find an s-t path that passes through every vertex in V once (s-t Hamiltonian path). We construct a new graph G' as follows: for each vertex $u \in V \setminus \{s, t\}$, we split u into two vertices u_1, u_2 and add an "internal" edge (u_1, u_2). For each edge $(u, v) \in E$ such that $u, v \in V \setminus \{s, t\}$, we change the original (u, v) to two edges (u_1, v_2) and (u_2, v_1). For each edge $(s, u) \in E$ such that $u \in V \setminus \{s, t\}$, we change the original (s, u) to edge (s, u_1). For each edge $(u, t) \in E$ such that $u \in V \setminus \{s, t\}$, we change the original (u, t) to edge (u_2, t).

Then, the following two statements are equivalent: (i) there exists an s-t Hamiltonian path in G, and (ii) there exists an s-t path in G', which contains all internal edges (u_1, u_2) for $u \in V \setminus \{s, t\}$. If there is a polynomial-time oracle to find an s-t path which contains a given edge set, then this oracle can solve the given Hamiltonian path instance in polynomial time. However, the Hamiltonian Path Problem is NP-hard, and thus the problem of finding an s-t path which contains a given edge set is also NP-hard. □

E Time Complexity

In this paper, all our algorithms run in polynomial time. Since the running time of our algorithms mainly depends on their used offline procedures, here we present the time complexity of used offline procedures on three common decision classes, e.g., s-t paths, maximum cardinality matchings and spanning trees. Let E and V denote the numbers of edges and vertices in the graph, respectively.

	`MaxOracle`	`ExistOracle`	`BottleneckSearch`	`AR-Oracle`
s-t paths	$O(E)$	$O(E+V)$	$O(E^2(E+V))$	$O(E(E+V))$
matchings	$O(V\sqrt{VE})$	$O(E\sqrt{V})$	$O(E^3\sqrt{V})$	$O(E^2\sqrt{V})$
spanning trees	$O(E)$	$O(E)$	$O(E^3)$	$O(E^2)$

Table 1: Time complexity of the offline procedures used in our algorithms.

E.1 Reduction of `ExistOracle` to 2-vertex Connectivity

In this subsection, we show how to reduce *the problem of finding a s-t path that contains a given edge* (u, v) (`ExistOracle`) to *the* 2*-vertex connectivity problem [20]* as follows.

First, we formally define these two problems.

Problem A (`ExistOracle`)**.** Given a graph G with vertices s, t, u, v, check if there exists a s-t simple path that contains (u, v), and output such a path if it exists.

Problem B (2**-vertex connectivity).** Given a graph G with vertices w, z, check if there exist two vertex-disjoint paths connecting w and z, and output such two vertex-disjoint paths if they exist.

Now we present the proof of reduction from Problem A to Problem B.

Proof. The reduction starts from a given instance of Problem A. Given a graph G with vertices s, t, u, v, we divide edge (u, v) into two edges $(u, w), (w, v)$ with an added virtual vertex w. Similarly, we also divide edge (s, t) into two edges $(s, z), (z, t)$ with an added virtual vertex z. Now, we show that finding a s-t simple path that contains (u, v) is equivalent to finding two vertex-disjoint paths connecting w and z.

(i) If we have a s-t simple path p that contains (u, v), then p has two subpaths p_1, p_2 connecting s, w and w, t, respectively, where p_1, p_2 do not have overlapped vertices. We concatenate p_1 and (s, z), and concatenate p_2 and (t, z). Then, we obtain two vertex-disjoint paths connecting w and z.

(ii) If we have two vertex-disjoint paths connecting w and z, then using the facts that w is only connected to vertices u, v and z is only connected to vertices s, t, we can obtain two vertex-disjoint paths q_1, q_2 connecting s, u and t, v, respectively (or connecting s, v and t, u, respectively). We concatenate q_1, q_2 and (u, v). Then, we obtain a s-t simple path that contains (u, v).

Therefore, we showed that for any given instance of Problem A, we can transform it to an instance of Problem B (by the above construction), and then use an existing oracle of Problem B [20] to solve the given instance of Problem A. □

F Extension to General Reward Functions

F.1 Problem Setting

In this section, we study the extension of CPE-B to general reward functions (CPE-G) in the fixed-confidence setting. Let $f(M, \boldsymbol{w})$ denote the expected reward function of super arm M, which only depends on $\{w(e)\}_{e \in M}$. Different from previous CPE works [11, 10, 24] which either study the linear reward function or impose strong assumptions (continuous and separable [24]) on nonlinear reward functions, we only make the following two standard assumptions:

Assumption 1 (Monotonicity)**.** *For any $M \in \mathcal{M}$ and $\boldsymbol{v}, \boldsymbol{v}' \in \mathbb{R}^n$ such that $\forall e \in [n],\ v'(e) \leq v(e)$, we have $f(M, \boldsymbol{v}') \leq f(M, \boldsymbol{v})$.*

Assumption 2 (Lipschitz continuity with ∞-norm)**.** *For any $M \in \mathcal{M}$ and $\boldsymbol{v}, \boldsymbol{v}' \in \mathbb{R}^n$, there exists a universal constant $U > 0$ such that $|f(M, \boldsymbol{v}) - f(M, \boldsymbol{v}')| \leq U \max_{e \in M} |v(e) - v'(e)|$.*

A wide family of reward functions satisfy these two mild assumptions, with the linear reward function (CPE-L) [11, 10], bottleneck reward function (CPE-B) and continuous and separable reward functions (CPE-CS) [24] as its special cases. In addition, many other interesting problems, such as the quadratic network flow [37], quadratic network allocation [23] and the densest subgraph [21], are encompassed by CPE-G.

F.2 Algorithm for CPE-G

For CPE-G, we propose a novel algorithm `GenLUCB` as in Algorithm 8. We allow `GenLUCB` to access an efficient maximization oracle `MaxOracle` for reward function f to find an optimal super arm from the given decision class and weight vector. Formally, $\texttt{MaxOracle}(\mathcal{F}, \boldsymbol{v}) \in \operatorname{argmax}_{M \in \mathcal{F}} f(M, \boldsymbol{v})$. We describe the procedure of `GenLUCB` as follows: at each timestep, we compute the lower and upper confidence bounds of the base arm rewards, and use the maximization oracle `MaxOracle` to find the super arm M_t with the maximum pessimistic reward from $\mathcal{M}$ and super arm $\tilde{M}_t$ with the maximum optimistic reward from $\mathcal{M} \setminus \{M_t\}$. Then, we play the base arm p_t with the maximum confidence radius from $M_t \cup \tilde{M}_t$. When we see that the pessimistic reward of M_t is higher than the optimistic reward of $\tilde{M}_t$, which implies that M_t has a higher reward than any other super arm with high confidence, we stop the algorithm and return M_t.

Algorithm 8 `GenLUCB`

```
1: Input: decision class $\mathcal{M}$, confidence $\delta \in (0,1)$, reward function $f$ and maximization oracle MaxOracle for $f$.
2: Initialization: play each base arm $e \in [n]$ once. Initialize empirical means $\hat{\boldsymbol{w}}_{n+1}$ and set $T_{n+1}(e) \leftarrow 1, \forall e \in [n]$.
3: for $t = n+1, n+2, \dots$ do
4:     $\mathrm{rad}_t(e) \leftarrow R\sqrt{2\ln(\frac{4nt^3}{\delta})/T_t(e)},\ \forall e \in [n]$
5:     $\underline{w}_t(e) \leftarrow \hat{w}_t(e) - \mathrm{rad}_t(e),\ \forall e \in [n]$
6:     $\bar{w}_t(e) \leftarrow \hat{w}_t(e) + \mathrm{rad}_t(e),\ \forall e \in [n]$
7:     $M_t \leftarrow \texttt{MaxOracle}(\mathcal{M}, \underline{\boldsymbol{w}}_t)$
8:     $\tilde{M}_t \leftarrow \texttt{MaxOracle}(\mathcal{M} \setminus \{M_t\}, \bar{\boldsymbol{w}}_t)$ or $\tilde{M}_t \leftarrow \texttt{MaxOracle}(\mathcal{M} \setminus \mathcal{S}(M_t), \bar{\boldsymbol{w}}_t)$
9:     if $f(M_t, \underline{\boldsymbol{w}}_t) \geq f(\tilde{M}_t, \bar{\boldsymbol{w}}_t)$ then
10:        return $M_t$
11:    end if
12:    $p_t \leftarrow \mathrm{argmax}_{M_t \cup \tilde{M}_t}\, \mathrm{rad}_t(e)$
13:    Play base arm $p_t$ and observe the reward
14:    Update empirical means $\hat{\boldsymbol{w}}_{t+1}$
15:    Update the number of pulls: $T_{t+1}(p_t) \leftarrow T_t(p_t) + 1$ and $T_{t+1}(e) \leftarrow T_t(e)$ for all $e \neq p_t$.
16: end for
```

Different from CPE-B or previous CPE works [11, 12] which only focus on the bottleneck base arms or those in symmetric difference, in CPE-G we select the base arm among the *entire* union set of two critical super arms, since for these two super arms, any base arm in their union can affect the reward difference and should be estimated.

F.3 Implementation of the Oracle in `GenLUCB`

Now we discuss the implementation of `MaxOracle` in `GenLUCB`. For $\mathcal{F} = \mathcal{M}$, we simply calculate an optimal super arm from $\mathcal{M}$ with respect to $\boldsymbol{v}$. Such a maximization oracle can be implemented efficiently for a rich class of decision classes and nonlinear reward functions, such as the densest subgraph [28], quadratic network flow problems [37] and quadratic network allocation problems [23]. For $\mathcal{F} = \mathcal{M} \setminus \{M_t\}$, it is more challenging to implement in polynomial time. We first discuss three common cases, where the step $\tilde{M}_t \leftarrow \texttt{MaxOracle}(\mathcal{M} \setminus \{M_t\}, \bar{\boldsymbol{w}}_t)$ (labeled as (a)) can be replaced with a more practical statement $\tilde{M}_t \leftarrow \texttt{MaxOracle}(\mathcal{M} \setminus \mathcal{S}(M_t), \bar{\boldsymbol{w}}_t)$ (labeled as (b)). Then, we can implement it as follows: repeatedly remove each base arm in M_t and compute the best super arm from the remaining decision class, and then return the one with the maximum reward.

Below we formally state the three cases:

Case (i). Any two super arms $M, M' \in \mathcal{M}$ satisfies $M \setminus M' \neq \varnothing$.

In this case, $\mathcal{S}(M_t) = M_t$ and the statements (a),(b) are equivalent. Many decision classes such as top k, maximum cardinality matchings, spanning trees fall in this case.

Case (ii). f is set monotonically decreasing.

As CPE-B, $f(M_t, \boldsymbol{w}) \geq f(M', \boldsymbol{w})$ for all $M' \in \mathcal{S}(M_t)$, and we only need to compare M_t against super arms in $\mathcal{M} \setminus \mathcal{S}(M_t)$.

Case (iii). f is strictly set monotonically increasing.

According to Line 7 of Algorithm 8, we have that $\mathcal{S}(M_t) = M_t$ and the statements (a),(b) are equivalent. Linear (CPE-L), quadratic, and continuous and separable (CPE-CS) reward functions satisfy this property when the expected rewards of base arms are non-negative.

If neither of the above cases holds, algorithm `GenLUCB` executes $\tilde{M}_t \leftarrow \texttt{MaxOracle}(\mathcal{M} \setminus \{M_t\}, \bar{\boldsymbol{w}}_t)$ by disabling M_t in some way and finding the best super arm from the remaining decision class with the basic maximization oracle. For the densest subgraph problem, for example, we can construct $\texttt{MaxOracle}(\mathcal{M} \setminus \{M_t\}, \bar{\boldsymbol{w}}_t)$ efficiently by the following procedure. Given $M_t \subseteq E$, we consider the corresponding a set of vertices $S_t \subseteq V$. First, for each vertex $i \in S_t$, we remove $i \in S_t$ from V, and obtain the best solution S_i^* in the remaining graph by using any exact algorithms. Second, for each $j \notin S_t$, we force $\{j\} \cup S$ to be included, and obtain the best solution S_j^*. Then we output the best solution among them. Note that the second step can be efficiently done by an exact flow-based algorithm with a min-cut procedure [21].

F.4 Sample Complexity of GenLUCB

Now we show the sample complexity of GenLUCB for CPE-G. For any $e \notin M_*$, let $\Delta_e^{\mathrm{G}} = f(M_*, \boldsymbol{w}) - \max\limits_{M \in \mathcal{M}: e \in M} f(M, \boldsymbol{w})$, and for any $e \in M_*$, let $\Delta_e^{\mathrm{G}} = f(M_*, \boldsymbol{w}) - \max_{M \neq M_*} f(M, \boldsymbol{w}) = \Delta_{\min}$. We formally state the sample complexity of GenLUCB in Theorem 10.

Theorem 10. *With probability at least $1-\delta$, the* GenLUCB *algorithm for CPE-G will return the optimal super arm with sample complexity*

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

Compared to the uniform sampling algorithm (presented in Appendix G) which has the $O(\frac{R^2 U^2 n}{\Delta_{\min}^2} \ln(\frac{R^2 U^2 n}{\Delta_{\min}^2 \delta}))$ sample complexity, GenLUCB achieves a much tighter result owing to the adaptive sample strategy, which validates its effectiveness. Moreover, to our best knowledge, GenLUCB is the first algorithm with non-trivial sample complexity for CPE with general reward functions, which encompass a rich class of nonlinear combinatorial problems, such as the densest subgraph problem [21], quadratic network flow problem [37] and quadratic network allocation problem [23].

To prove the sample complexity of algorithm GenLUCB (Theorem 10), we first introduce the following Lemmas 18,19.

Lemma 18 (Correctness of GenLUCB)**.** *Assume that event ξ occurs. Then, if algorithm* GenLUCB *(Algorithm 8) terminates at round t, we have $M_t = M_*$.*

Proof. According to the stop condition (Line 9 of Algorithm 8), when algorithm GenLUCB (Algorithm 8) terminates at round t, we have that for any $M \neq M_t$,

$$f(M_t, \boldsymbol{w}) \geq f(M_t, \underline{\boldsymbol{w}}_t) \geq f(M, \bar{\boldsymbol{w}}_t) \geq f(M, \boldsymbol{w}).$$

Thus, we have $M_t = M_*$. □

Lemma 19. *Assume that event ξ occurs. For any $e \in [n]$, if $\mathrm{rad}_t(e) < \frac{\Delta_e^G}{4U}$, then, base arm e will not be pulled at round t, i.e., $p_t \neq e$.*

Proof. (i) Suppose that for some $e \notin M_*$, $\mathrm{rad}_t(e) < \frac{\Delta_e^G}{4U} = \frac{1}{4U}(f(M_*, \boldsymbol{w}) - \max_{M \in \mathcal{M}: e \in M} f(M, \boldsymbol{w}))$ and $p_t = e$. According to the selection strategy of p_t, we have that for any $e' \in M_t \cup \tilde{M}_t$, $\mathrm{rad}_t(e') \leq \mathrm{rad}_t(e) < \frac{\Delta_e^G}{4U}$.

First, we can prove that $M_t \neq M_*$ and $\tilde{M}_t \neq M_*$. Otherwise, one of $M_t, \tilde{M}_t$ is M_* and the other is a sub-optimal super arm containing e, which is denoted by M'. Then,

$$\begin{aligned} f(M_*, \underline{\boldsymbol{w}}_t) - f(M', \bar{\boldsymbol{w}}_t) \geq & (f(M_*, \boldsymbol{w}) - 2U \max_{i \in M_*} \mathrm{rad}_i) - (f(M', \boldsymbol{w}) + 2U \max_{j \in M'} \mathrm{rad}_j) \\ > & \Delta_{M_*, M'}^G - \frac{\Delta_e^G}{2} - \frac{\Delta_e^G}{2} \\ = & 0, \end{aligned}$$

which gives a contradiction.

Then, if $e \in \tilde{M}_t$, we have

$$\begin{aligned} f(\tilde{M}_t, \bar{\boldsymbol{w}}_t) \leq & f(\tilde{M}_t, \boldsymbol{w}) + 2U \max_{i \in \tilde{M}_t} \mathrm{rad}_i \\ < & f(\tilde{M}_t, \boldsymbol{w}) + \frac{\Delta_e^G}{2} \end{aligned}$$

$$\begin{aligned} &< f(M_*, \boldsymbol{w}) \\ &\le f(M_*, \bar{\boldsymbol{w}}_t), \end{aligned}$$

which contradicts the definition of $\tilde{M}_t$.

If $e \in M_t$, we have

$$\begin{aligned} f(\tilde{M}_t, \bar{\boldsymbol{w}}_t) - f(\tilde{M}_t, \underline{\boldsymbol{w}}_t) \ge & f(M_*, \bar{\boldsymbol{w}}_t) - f(M_t, \underline{\boldsymbol{w}}_t) \\ \ge & f(M_*, \boldsymbol{w}) - f(M_t, \boldsymbol{w}) \\ = & \Delta^G_{M_*, M_t}. \end{aligned}$$

On the other hand, we have

$$\begin{aligned} f(\tilde{M}_t, \bar{\boldsymbol{w}}_t) - f(\tilde{M}_t, \underline{\boldsymbol{w}}_t) \le & (f(\tilde{M}_t, \hat{\boldsymbol{w}}) + U \max_{i \in \tilde{M}_t} \mathrm{rad}_i) - (f(\tilde{M}_t, \hat{\boldsymbol{w}}) - U \max_{i \in \tilde{M}_t} \mathrm{rad}_i) \\ = & 2U \max_{i \in \tilde{M}_t} \mathrm{rad}_i \\ < & \frac{\Delta^G_e}{2} \\ \le & \frac{\Delta^G_{M_*, M_t}}{2} \\ < & \Delta^G_{M_*, M_t}, \end{aligned}$$

which gives a contradiction.

(ii) Suppose that for some $e \in M_*$, $\mathrm{rad}_t(e) < \frac{\Delta^G_e}{4U} = \frac{\Delta_{\min}}{4U}$ and $p_t = e$. According to the selection strategy of p_t, we have that for any $e' \in M_t \cup \tilde{M}_t$, $\mathrm{rad}_t(e') \le \mathrm{rad}_t(e) < \frac{\Delta_{\min}}{4U}$.

First, we can prove that $M_t \neq M_*$ and $\tilde{M}_t \neq M_*$. Otherwise, one of $M_t, \tilde{M}_t$ is M_* and the other is a sub-optimal super arm, which is denoted by M'. Then,

$$\begin{aligned} f(M_*, \underline{\boldsymbol{w}}_t) - f(M', \bar{\boldsymbol{w}}_t) \ge & (f(M_*, \boldsymbol{w}) - 2U \max_{i \in M_*} \mathrm{rad}_i) - (f(M', \boldsymbol{w}) + 2U \max_{j \in M'} \mathrm{rad}_j) \\ > & \Delta^G_{M_*, M'} - \frac{\Delta_{\min}}{2} - \frac{\Delta_{\min}}{2} \\ = & 0, \end{aligned}$$

which gives a contradiction.

Thus, both M_t and $\tilde{M}_t$ are sub-optimal super arms.

However, on the other hand, we have

$$\begin{aligned} f(\tilde{M}_t, \bar{\boldsymbol{w}}_t) \le & f(\tilde{M}_t, \boldsymbol{w}) + 2U \max_{i \in \tilde{M}_t} \mathrm{rad}_i \\ < & f(\tilde{M}_t, \boldsymbol{w}) + \frac{\Delta_{\min}}{2} \\ < & f(M_*, \boldsymbol{w}) \\ \le & f(M_*, \bar{\boldsymbol{w}}_t), \end{aligned}$$

which contradicts the definition of $\tilde{M}_t$. □

Now, we prove Theorem 10.

Proof. For any $e \in [n]$, let $T(e)$ denote the number of samples for base arm e, and t_e denote the last timestep at which e is pulled. Then, we have $T_{t_e} = T(e) - 1$. Let T denote the total number of samples. According to Lemma 19, we have

$$R \sqrt{\frac{2 \ln(\frac{4 n t_e^3}{\delta})}{T(e) - 1}} \ge \frac{\Delta^G_e}{4U}$$

Algorithm 9 `UniformFC`

```
Input: decision class $\mathcal{M}$, confidence $\delta \in (0,1)$, reward function $f$ and maximization oracle MaxOracle for $f$.
for $t = 1, 2, \dots$ do
    For each base arm $e \in [n]$, pull $e$ once and then update its empirical mean $\hat{w}_t(e)$ and number of samples $T_t(e)$
    $\mathrm{rad}_t \leftarrow R\sqrt{2\ln(\frac{4nt^3}{\delta})/t}$
    $\underline{w}_t(e) \leftarrow \hat{w}_t(e) - \mathrm{rad}_t,\ \forall e \in [n]$
    $\bar{w}_t(e) \leftarrow \hat{w}_t(e) + \mathrm{rad}_t,\ \forall e \in [n]$
    $M_t \leftarrow$ MaxOracle$(\mathcal{M}, \underline{\boldsymbol{w}}_t)$
    $\tilde{M}_t \leftarrow$ MaxOracle$(\mathcal{M} \setminus \mathcal{S}(M_t), \bar{\boldsymbol{w}}_t)$
    if $f(M_t, \underline{\boldsymbol{w}}_t) \geq f(\tilde{M}_t, \bar{\boldsymbol{w}}_t)$ then
        return $M_t$
    end if
end for
```

Thus, we obtain

$$T(e) \leq \frac{32R^2U^2}{(\Delta_e^G)^2} \ln\left(\frac{4nt_e^3}{\delta}\right) + 1 \leq \frac{32R^2U^2}{(\Delta_e^G)^2} \ln\left(\frac{4nT^3}{\delta}\right) + 1$$

Summing over $e \in [n]$, we have

$$T \leq \sum_{e\in[n]} \frac{32R^2U^2}{(\Delta_e^G)^2} \ln\left(\frac{4nT^3}{\delta}\right) + n \leq \sum_{e\in[n]} \frac{96R^2U^2}{(\Delta_e^G)^2} \ln\left(\frac{2nT}{\delta}\right) + n,$$

where $\sum_{e\in[n]} \frac{R^2U^2}{(\Delta_e^G)^2} \geq n$. Then, applying Lemma 20, we have

$$\begin{aligned}
T \leq & \sum_{e\in[n]} \frac{576R^2U^2}{(\Delta_e^G)^2} \ln\left(\frac{2n^2}{\delta} \sum_{e\in[n]} \frac{96R^2U^2}{(\Delta_e^G)^2}\right) + n \\
=& O\left(\sum_{e\in[n]} \frac{R^2U^2}{(\Delta_e^G)^2} \ln\left(\sum_{e\in[n]} \frac{R^2U^2n^2}{(\Delta_e^G)^2\delta}\right) + n\right) \\
=& O\left(\sum_{e\in[n]} \frac{R^2U^2}{(\Delta_e^G)^2} \ln\left(\sum_{e\in[n]} \frac{R^2U^2n}{(\Delta_e^G)^2\delta}\right)\right).
\end{aligned}$$

□

G Uniform Sampling Algorithms

In this section, we present the uniform sampling algorithms for the fixed-confidence and fixed-budget CPE problems.

Algorithm 9 illustrates the uniform sampling algorithm `UniformFC` for the fixed-confidence CPE problem. Below we state the sample complexity of algorithm `UniformFC`.

Theorem 11. *With probability at least $1 - \delta$, the `UniformFC` algorithm (Algorithm 9) will return the optimal super arm with sample complexity*

$$O\left(\frac{R^2U^2n}{\Delta_{\min}^2} \ln\left(\frac{R^2U^2n}{\Delta_{\min}^2\delta}\right)\right).$$

Proof. Let $\Delta_{\min} = \min_{e\in[n]} \Delta_e^G = f(M_*, \boldsymbol{w}) - f(M_{\text{second}}, \boldsymbol{w})$. Assume that event ξ occurs. Then, if $\mathrm{rad}_t < \frac{\Delta_{\min}}{4U}$, algorithm `UniformFC` will stop. Otherwise,

$$f(M_t, \underline{\boldsymbol{w}}_t) - f(\tilde{M}_t, \bar{\boldsymbol{w}}_t) \geq f(M_t, \boldsymbol{w}) - 2U \max_{i\in M_t} \mathrm{rad}_t - (f(\tilde{M}_t, \boldsymbol{w}) + 2U \max_{i\in\tilde{M}_t} \mathrm{rad}_t)$$

Algorithm 10 UniformFB

```
Input: $\mathcal{M}$, budget $T$, reward function $f$ and maximization oracle MaxOracle for $f$.
Pull each base arm $e \in [n]$ for $\lfloor T/n \rfloor$ times
Update the empirical means $\hat{\boldsymbol{w}}_t$
$M_{\text{out}} \leftarrow$ MaxOracle($\mathcal{M}, \hat{\boldsymbol{w}}_t$)
return $M_{\text{out}}$
```

$$
\begin{aligned}
&= f(M_t, \boldsymbol{w}) - f(\tilde{M}_t, \boldsymbol{w}) - 4U\text{rad}_t \\
&> \Delta_{\text{min}} - \Delta_{\text{min}} \\
&= 0,
\end{aligned}
$$

which contradicts the stop condition.

Let T_n denote the number of rounds and T denote the total number of samples. Then, we have

$$
R\sqrt{\frac{2\ln(\frac{4nT_n^3}{\delta})}{T_n - 1}} \geq \frac{\Delta_{\text{min}}}{4U}
$$

Thus, we obtain

$$
T_n \leq \frac{32R^2U^2}{\Delta_{\text{min}}^2} \ln\left(\frac{4nT_n^3}{\delta}\right) + 1 \leq \frac{96R^2U^2}{\Delta_{\text{min}}^2} \ln\left(\frac{2nT_n}{\delta}\right) + 1.
$$

Then, applying Lemma 20, we have

$$
\begin{aligned}
T_n \leq & \frac{576R^2U^2}{\Delta_{\text{min}}^2} \ln\left(\frac{2n}{\delta} \frac{96R^2U^2}{\Delta_{\text{min}}^2}\right) + 1 \\
= & O\left(\frac{R^2U^2}{\Delta_{\text{min}}^2} \ln\left(\frac{R^2U^2n}{\Delta_{\text{min}}^2\delta}\right)\right).
\end{aligned}
$$

Summing over the number of samples for all the base arms, we obtain

$$
T = O\left(\frac{R^2U^2n}{\Delta_{\text{min}}^2} \ln\left(\frac{R^2U^2n}{\Delta_{\text{min}}^2\delta}\right)\right).
$$

□

Algorithm 10 illustrates the uniform sampling algorithm UniformFB for the fixed-budget CPE problem. Below we state the error probability of algorithm UniformFB.

Let $H^U = n(\Delta_{\text{min}})^{-2}$, where $\Delta_{\text{min}} = f(M_*, \boldsymbol{w}) - f(M_{\text{second}}, \boldsymbol{w})$.

Theorem 12. *For any $T > n$, algorithm* UniformFB *uses at most T samples and returns the optimal super arm with the error probability bounded by*

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

Proof. Since algorithm UniformFB allocates $\lfloor T/n \rfloor$ samples to each base arm $e \in [n]$, the total number of samples is at most T.

Now we prove the error probablity. Define event

$$
\mathcal{G} = \left\{\forall i \in [n],\ |\hat{w}_t(i) - w(i)| < \frac{\Delta_{\text{min}}}{4U}\right\}.
$$

According to the Hoeffding's inequality,

$$
\left\{|\hat{w}_t(i) - w(i)| \geq \frac{\Delta_{\text{min}}}{4U}\right\} \leq 2\exp\left(-\frac{T\Delta_{\text{min}}^2}{32R^2U^2n}\right).
$$

By a union bound over $e \in [n]$, we have

$$\begin{aligned}\Pr[\mathcal{G}] \geq & 1 - 2n \exp\left(-\frac{T\Delta_{\min}^2}{32R^2U^2n}\right)\\ = & 1 - 2n \exp\left(-\frac{T}{32R^2U^2H^U}\right)\end{aligned}$$

Below we prove that conditioning on event $\mathcal{G}$, $M_{\text{out}} = M_*$. Suppose that $M_{\text{out}} \neq M_*$,

$$\begin{aligned}f(M_{\text{out}}, \hat{\boldsymbol{w}}_t) - f(M_*, \hat{\boldsymbol{w}}_t) \leq & f(M_{\text{out}}, \boldsymbol{w}) + \frac{\Delta_{\min}}{4} - (f(M_*, \boldsymbol{w}) - \frac{\Delta_{\min}}{4})\\ \leq & -\Delta_{\min} + \frac{\Delta_{\min}}{2}\\ = & -\frac{\Delta_{\min}}{2}\\ < & 0,\end{aligned}$$

which contradicts the selection strategy of M_{out}. Thus, conditioning on event $\mathcal{G}$, algorithm `UniformFB` returns M_*. Then, we obtain Theorem 12. □

H Technical Tool

In this section, we present a technical tool used in the proofs of our results.

Lemma 20. *If $T \leq c_1 \ln(c_2 T) + c_3$ holds for some constants $c_1, c_2, c_3 \geq 1$ such that $\ln(c_1c_2c_3) \geq 1$, we have $T \leq 6c_1 \ln(c_1c_2c_3) + c_3$.*

Proof. In the inequality $T \leq c_1 \ln(c_2T) + c_3$, the LHS is linear with respect to T and the RHS is logarithmic with respect to T. Thus, we have $T > c_1 \ln(c_2T) + c_3$ for a big enough T. Then, to prove $T \leq T_0 \triangleq 6c_1 \ln(c_1c_2c_3) + c_3$, it suffices prove that $T_0 > c_1 \ln(c_2T_0) + c_3$. Since

$$\begin{aligned}c_1 \ln(c_2T_0) + c_3 = & c_1 \ln(c_2(6c_1 \ln(c_1c_2c_3) + c_3)) + c_3\\ = & c_1 \ln(6c_1c_2 \ln(c_1c_2c_3) + c_2c_3) + c_3\\ \leq & c_1 \ln(6c_1^2c_2^2c_3 + c_2c_3) + c_3\\ \leq & c_1 \ln(7c_1^2c_2^2c_3) + c_3\\ \leq & 2c_1 \ln(7c_1c_2c_3) + c_3\\ = & 2c_1 \ln(c_1c_2c_3) + 2c_1 \ln(7) + c_3\\ \leq & 2c_1 \ln(c_1c_2c_3) + 2\ln(7)c_1 \ln(c_1c_2c_3) + c_3\\ = & (2 + 2\ln(7))c_1 \ln(c_1c_2c_3) + c_3\\ \leq & 6c_1 \ln(c_1c_2c_3) + c_3\\ = & T_0,\end{aligned}$$

we obtain the lemma. □