Fixed-Budget Real-Valued Combinatorial Pure Exploration of Multi-Armed Bandit

Shintaro Nakamura
University of Tokyo
RIKEN AIP

Masashi Sugiyama
RIKEN AIP
University of Tokyo

Abstract

We study the real-valued combinatorial pure exploration of the multi-armed bandit in the fixed-budget setting. We first introduce the Combinatorial Successive Assign (CSA) algorithm, which is the first algorithm that can identify the best action even when the size of the action class is exponentially large with respect to the number of arms. We show that the upper bound of the probability of error of the CSA algorithm matches a lower bound up to a logarithmic factor in the exponent. Then, we introduce another algorithm named the Minimax Combinatorial Successive Accepts and Rejects (Minimax-CombSAR) algorithm for the case where the size of the action class is polynomial, and show that it is optimal, which matches a lower bound. Finally, we experimentally compare the algorithms with previous methods and show that our algorithm performs better.

1 Introduction

The multi-armed bandit (MAB) model is an important framework in online learning since it is useful to investigate the trade-off between exploration and exploitation in decision-making problems (Auer et al., 2002; Audibert et al., 2009). Although investigating this trade-off is intrinsic in many applications, some application domains only focus on obtaining the optimal object, e.g., an arm or a set of arms, among a set of candidates, and do not care about the loss or rewards that occur during the exploration procedure. This learning problem called the pure exploration (PE) task has received much attention (Bubeck et al., 2009; Audibert et al., 2010).

One of the important sub-fields among PE of MAB is the *combinatorial pure exploration* of the MAB (CPE-MAB)(Chen et al., 2014; Gabillon et al., 2016; Chen et al., 2017). In the CPE-MAB, we have a set of d stochastic arms, where the reward of each arm $s \in \{1, \dots, d\}$ follows an unknown distribution with mean μ_s, and an *action class* $\mathcal{A}$, which is a collection of subsets of arms with certain combinatorial structures. Then, the goal is to identify the best action from the action class $\mathcal{A}$ by pulling a single arm each round. There are mainly two settings in the CPE-MAB. One is the *fixed confidence* setting, where the player tries to identify the optimal action with high probability with as few rounds as possible, and the other is the *fixed-budget* setting, where the player tries to identify the optimal action with a fixed number of rounds (Chen et al., 2014; Katz-Samuels et al., 2020; Wang and Zhu, 2022a; Gabillon et al., 2016). Abstractly, the goal is to identify $\boldsymbol{\pi}^*$, which is the optimal solution for the following constraint optimization problem:

$$\begin{array}{ll} \text{maximize}_{\boldsymbol{\pi}} & \boldsymbol{\mu}^\top \boldsymbol{\pi} \\ \text{subject to} & \boldsymbol{\pi} \in \mathcal{A}, \end{array} \tag{1}$$

where $\boldsymbol{\mu}$ is a vector whose s-th element is the mean reward of arm s and $\top$ denotes the transpose.

Although CPE-MAB can be applied to many models which can be formulated as (1), most of the existing works in CPE-MAB (Chen et al., 2014; Wang and Zhu, 2022b; Gabillon et al., 2016; Chen et al., 2017; Du et al., 2021; Chen et al., 2016) assume $\mathcal{A} \subseteq \{0,1\}^d$. This means that although we can apply the CPE-MAB framework to the shortest path problem (Sniedovich, 2006), top-K arms identification (Kalyanakrishnan and Stone, 2010), matching (Gibbons, 1985), and spanning trees (Pettie and Ramachandran, 2002), we cannot apply it to problems where $\mathcal{A} \subset \mathbb{R}^d$, such as the optimal transport problem (Villani, 2008), the knapsack problem (Dantzig and Mazur, 2007), and the production planning problem

Proceedings of the 27$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

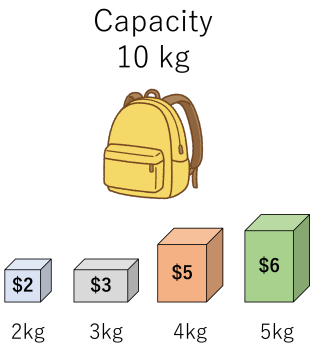

Figure 1: A simple sketch of the knapsack problem. We want to know how many of each item should be included in the bag to maximize the total value. Here, the total weight of every item cannot exceed 10kg, which is the capacity of the bag.

(Pochet and Wolsey, 2010). For instance, in the knapsack problem shown in Figure 1, actions are not binary vectors since, for each item, we can put more than one in the bag, e.g., one blue one and two orange ones.

To overcome this limitation, Nakamura and Sugiyama (2023) introduced the real-valued CPE-MAB (R-CPE-MAB), where the action class is a set of real vectors, i.e., $\mathcal{A} \subset \mathbb{R}^d$. Though they have investigated the R-CPE-MAB in the fixed confidence setting, there is still room for investigation in the fixed-budget setting. To the best of our knowledge, the only existing work that can be applied to the fixed-budget R-CPE-MAB is the Peace algorithm introduced by Katz-Samuels et al. (2020). However, the Peace algorithm required enumerating all of the feasible actions. This is nearly impossible since, in general, there is no efficient algorithm for enumerating all actions for a combinatorial optimization problem. Even if they could be enumerated, it would require an operation to sort an exponentially large number of actions by estimated reward, which is practically unrealistic in terms of computation time. Secondly, there is a hyper-parameter for it, and one needs to carefully choose them for the feasibility of the Peace algorithm (Yang and Tan, 2022). Finally, it needs an assumption that the rewards follow a normal distribution, which may not be satisfied in real-world applications.

In this work, we first introduce a parameter-free algorithm named the Combinatorial Successive Assign (CSA) algorithm, which is a generalized version of the Combinatorial Successive Accepts and Rejects (CSAR) algorithm proposed by Chen et al.(2014) for the ordinary CPE-MAB. Since the CSA algorithm does not require enumerating all the actions in $\mathcal{A}$, it can be applied to cases even when the size of the action class $\mathcal{A}$ is exponentially large in d. We show that the upper bound of the probability of error of the CSA algorithm matches a lower bound up to a logarithmic factor in the exponent.

Since the CSA algorithm does not match a lower bound, we introduce another algorithm named the Minimax Combinatorial Successive Accepts and Rejects (Minimax-CombSAR) algorithm inspired by Yang and Tan (2022) for the case where the size of the action class is polynomial with respect to the number of arms. In Section 4, we show that the Minimax-CombSAR algorithm is optimal, which means that the upper bound of the probability of error of the best action matches a lower bound. We also show that the Minimax-CombSAR algorithm has only one hyper-parameter, and is easily interpreted.

Finally, we report the results of numerical experiments. First, we show that the CSA algorithm can identify the best action in a knapsack problem, where the size of the action class can be exponentially large in d. Then, when the size of the action class is polynomial in d, we show that the Minimax-CombSAR algorithm performs better than the CSA algorithm and the Peace algorithm in identifying the best action in the knapsack and optimal transport problems.

2 Problem Formulation

In this section, we formally define our R-CPE-MAB model. Suppose we have d arms, numbered $1, \dots, d$. Assume that each arm $s \in [d]$ is associated with a reward distribution ϕ_s, where $[d] = \{1, \dots, d\}$. We assume all reward distributions have R-sub-Gaussian tails for some known constant $R > 0$. Formally, if X is a random variable drawn from ϕ_s for some $s \in [d]$, then, for all $\lambda \in \mathbb{R}$, we have $\mathbb{E}[\exp(\lambda X - \lambda \mathbb{E}[X])] \leq \exp(R^2\lambda^2/2)$. It is known that the family of R-sub-Gaussian tail distributions includes all distributions that are supported on $[a, b]$, where $(b-a)/2 = R$, and also many unbounded distributions such as Gaussian distributions with variance R^2 (Rivasplata, 2012; Rinaldo and Bong, 2018). Let $\boldsymbol{\mu} = (\mu_1, \dots, \mu_d)^\top$ denote the vector of expected rewards, where each element $\mu_s = \mathbb{E}_{X \sim \phi_s}[X]$ denotes the expected reward of arm s and $\top$ denotes the transpose. With a given $\boldsymbol{\nu}$, let us consider the following linear optimization problem:

$$\begin{array}{ll} \text{maximize}_{\boldsymbol{\pi}} & \boldsymbol{\nu}^\top \boldsymbol{\pi} \\ \text{subject to} & \boldsymbol{\pi} \in \mathcal{C} \subset \mathbb{R}^d_{\geq 0}. \end{array} \tag{2}$$

Here, $\mathcal{C}$ is a problem-dependent feasible region of $\boldsymbol{\pi}$, which satisfies some combinatorial structures. Then, for any $\boldsymbol{\nu} \in \mathbb{R}^d$, we denote by $\boldsymbol{\pi}^{\boldsymbol{\nu},\mathcal{C}}$ the solution of (2). We define the action class $\mathcal{A}$ as the set of vectors that

contains optimal solutions of (2) for any $\boldsymbol{\nu}$, i.e.,

$$\mathcal{A} = \left\{ \boldsymbol{\pi}^{\boldsymbol{\nu},\mathcal{C}} \in \mathbb{R}^d_{\geq 0} \mid \forall \boldsymbol{\nu} \in \mathbb{R}^d \right\}. \tag{3}$$

We assume that the size of $\mathcal{A}$ is finite and denote it by K. This assumption is relatively mild since, for instance, in linear programming, the optimal solution can always be found at one of the vertices of the feasible region (Ahmadi, 2016).

Also, let $\texttt{POSSIBLE-PI}(s) = \{x \in \mathbb{R} \mid \exists \boldsymbol{\pi} \in \mathcal{A}, \pi_s = x\}$, where π_s denotes the s-th element of $\boldsymbol{\pi}$. We can see $\texttt{POSSIBLE-PI}(s)$ as the set of all possible values that an action in the action set can take as the s-th element. Let us denote the size of $\texttt{POSSIBLE-PI}(s)$ by B_s. Note that K, the size of $\mathcal{A}$, can be exponentially large in d, i.e., $|\mathcal{A}| = \mathcal{O}(\prod_{s=1}^d B_s)$.

We assume we have *offline oracles* which efficiently solve the linear optimization problem (2) once ν is given. For instance, offline oracles can be algorithms which output $\boldsymbol{\pi}^*(\boldsymbol{\nu}) = \arg\max_{\boldsymbol{\pi} \in \mathcal{A}} \boldsymbol{\nu}^\top \boldsymbol{\pi}$ in polynomial or pseudo-polynomial [1] time in d.

The player's objective is to identify the best action $\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{\pi} \in \mathcal{A}} \boldsymbol{\mu}^\top \boldsymbol{\pi}$ by pulling a single arm each round. The player is given a *budget* T, and cannot pull arms more than T times. The player outputs an action $\boldsymbol{\pi}^{\text{out}}$ at the end, and she is evaluated by the *probability of error*, which is formally $\Pr[\boldsymbol{\pi}^{\text{out}} \neq \boldsymbol{\pi}^*]$.

3 Lower Bound of the Fixed-Budget R-CPE-MAB

In this section, we show a lower bound of the probability of error in R-CPE-MAB. As preliminaries, let us introduce some notions. First, we introduce $\boldsymbol{\pi}^{(s)}$ as follows (Nakamura and Sugiyama, 2023):

$$\boldsymbol{\pi}^{(s)} = \arg\min_{\boldsymbol{\pi} \in \mathcal{A} \setminus \{\boldsymbol{\pi}^*\}} \frac{\boldsymbol{\mu}^\top (\boldsymbol{\pi}^* - \boldsymbol{\pi})}{|\pi^*_s - \pi_s|}. \tag{4}$$

Intuitively, among the actions whose s-th element is different from $\boldsymbol{\pi}^*$, $\boldsymbol{\pi}^{(s)}$ is the action which is the most difficult to determine whether it is the best action or not. We also introduce the notion *G-gap* (Nakamura and Sugiyama, 2023) as follows:

$$\begin{aligned} \Delta_s &= \min_{\boldsymbol{\pi} \in \mathcal{A} \setminus \{\boldsymbol{\pi}^*\}} \frac{\boldsymbol{\mu}^\top (\boldsymbol{\pi}^* - \boldsymbol{\pi})}{|\pi^*_s - \pi_s|} \\ &= \frac{\boldsymbol{\mu}^\top \left(\boldsymbol{\pi}^* - \boldsymbol{\pi}^{(s)}\right)}{\left|\pi^*_s - \pi^{(s)}_s\right|}. \end{aligned} \tag{5}$$

[1] A pseudo-polynomial time algorithm is a numeric algorithm whose running time is polynomial in the numeric value of the input, but not necessarily in the length of the input (Garey and Johnson, 1990)

G-Gap was first introduced in Nakamura and Sugiyama (2023) as a natural generalization of the notion *gap* in the CPE-MAB literature (Chen et al., 2014, 2016, 2017). This was introduced as a key notion that characterizes the difficulty of the problem instance.

In Theorem 1, we show that the sum of the inverse of squared *G-Gaps*,

$$\mathbf{H} = \sum_{s=1}^d \left(\frac{1}{\Delta_s}\right)^2, \tag{6}$$

appears in the lower bound of the probability of error of R-CPE-MAB, which implies that it characterizes the difficulty of the problem instance.

Theorem 1. *For any action class and any algorithm that returns an action $\boldsymbol{\pi}^{\text{out}}$ after T times of arm pulls, the probability of error is at least*

$$\mathcal{O}\left(\exp\left(-\frac{T}{\mathbf{H}}\right)\right). \tag{7}$$

We show the proof in Appendix A. If $\mathcal{A}$ is a set of d dimensional standard basis, R-CPE-MAB becomes the standard best arm identification problem whose objective is to identify the best arm with the largest expected reward among d arms (Bubeck et al., 2009; Audibert et al., 2010; Carpentier and Locatelli, 2016). From Carpentier and Locatelli (2016), a lower bound is $\mathcal{O}\left(\exp\left(-\frac{T}{\log(d)\mathbf{H}}\right)\right)$ for the standard best arm identification problem. It is a future work that whether the lower bound is $\mathcal{O}\left(\exp\left(-\frac{T}{\log(d)\mathbf{H}}\right)\right)$ for general action classes.

4 The Combinatorial Successive Assign (CSA) Algorithm

In this section, we first introduce the CSA algorithm, which can be seen as a generalization of the CSAR algorithm (Chen et al., 2014). This algorithm can be applied to fixed-budget R-CPE-MAB even when the size of the action class $\mathcal{A}$ is exponentially large in d. Then, we show an upper bound of the probability of error of the best action of the CSA algorithm. We also discuss the number of times offline oracles have to be called.

4.1 CSA Algorithm

In this subsection, we introduce the CSA algorithm, a fully parameter-free algorithm for fixed-budget R-CPE-MAB that works even when the action set $\mathcal{A}$ can be exponentially large in d.

We first define the *constrained offline oracle* which is used in the CSA algorithm.

Definition 1 (Constrained offline oracle)**.** *Let* $\boldsymbol{S} = \{(e,x) | (e,x) \in \mathbb{Z} \times \mathbb{R}\}$ *be a set of tuples. A constrained offline oracle is denoted by* COracle*:* $\mathbb{R}^d \times \boldsymbol{S} \to \mathcal{A} \cup \perp$ *and satisfies*

$$\mathrm{COracle}(\boldsymbol{\mu}, \boldsymbol{S}) = \begin{cases} \arg\max_{\boldsymbol{\pi} \in \mathcal{A}_{\boldsymbol{S}}} \boldsymbol{\mu}^\top \boldsymbol{\pi} & (\text{if } \mathcal{A}_{\boldsymbol{S}} \neq \emptyset), \\ \perp & (\text{if } \mathcal{A}_{\boldsymbol{S}} = \emptyset), \end{cases}$$

where we define $\mathcal{A}_{\boldsymbol{S}} = \{\boldsymbol{\pi} \in \mathcal{A} \mid \forall (e,x) \in \boldsymbol{S}, \pi_e = x\}$ *as the collection of feasible actions and* $\perp$ *is a null symbol.*

Here, we can see that a COracle is a modification of an offline oracle specified by $\boldsymbol{S}$. In other words, for all (e,x) in $\boldsymbol{S}$, the COracle outputs an action whose e-th element is x; otherwise, it outputs the null symbol. In Appendix B, we discuss how to construct such COracles for some combinatorial problems, such as the optimal transport problem and the knapsack problem.

We introduce the CSA algorithm in Algorithm 1. The CSA algorithm divides the budget into d rounds. In each round, we pull each of the remaining arms the same number of times (line 5). At each round t, the CSA outputs the empirically best action $\hat{\boldsymbol{\pi}}(t)$ (line 6), chooses a single arm $p(t)$ (line 14), and assigns $\hat{\pi}_{p(t)}(t)$ for the e-th element of $\boldsymbol{\pi}^{\mathrm{out}}$. Indices that are assigned are maintained in $\boldsymbol{F}(t)$, and arms $s \in F(t)$ will no longer be pulled in the next rounds. The pair of the index and the assigned value for $\boldsymbol{\pi}^{\mathrm{out}}$, $\boldsymbol{S}(t)$, is updated at every round (line 16).

Algorithm 1 CSA: Combinatorial Successive Assign Algorithm

Input: Budget: $T \geq 0$; COracle: $\to \mathcal{A} \cup \{\perp\}$

1: Define $\tilde{\log}(n) = \sum_{i=1}^{d} \frac{1}{i}$
2: $\tilde{T}_0 \leftarrow 0$, $\boldsymbol{F}(t) \leftarrow \emptyset$, $\boldsymbol{S}(t) = \emptyset$
3: **for** $t = 1$ to d **do**
4: $\quad \tilde{T}(t) \leftarrow \lceil \frac{T-d}{\tilde{\log}(d)(d-t+1)} \rceil$
5: $\quad$ Pull each arm $e \in [d] \setminus F(t)$ for $\tilde{T}(t) - \tilde{T}_{t-1}$ times
6: $\quad \hat{\boldsymbol{\pi}}(t) \leftarrow \mathrm{COracle}(\hat{\boldsymbol{\mu}}(t), \boldsymbol{S}(t))$
7: $\quad$ **if** $\hat{\boldsymbol{\pi}}(t) = \perp$ **then**
8: $\quad\quad$ **Fail**: set $\boldsymbol{\pi}^{\mathrm{out}} \leftarrow \perp$ and return $\boldsymbol{\pi}^{\mathrm{out}}$
9: $\quad$ **end if**
10: $\quad$ **for all** e in $[d] \setminus \boldsymbol{F}(t)$ **do**
11: $\quad\quad$ `CHECK(e)` $\leftarrow$ `POSSIBLE-PI(e)` $\setminus \{\hat{\pi}_e(t)\}$
12: $\quad\quad \tilde{\boldsymbol{\pi}}^e(t) \leftarrow \arg\max_{x \in \texttt{CHECK(e)}} \mathrm{COracle}(\hat{\boldsymbol{\mu}}(t), \boldsymbol{S}(t) \cup (e,x))$
13: $\quad$ **end for**
14: $\quad p(t) \leftarrow \arg\max_{e \in [d] \setminus F(t)} \frac{\langle \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\pi}}(t) - \tilde{\boldsymbol{\pi}}^e(t) \rangle}{\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)}$
15: $\quad \boldsymbol{F}(t+1) \leftarrow \boldsymbol{F}(t) \cup \{p(t)\}$
16: $\quad \boldsymbol{S}(t+1) \leftarrow \boldsymbol{S}(t) \cup \{(p(t), \hat{\pi}_{p(t)}(t)\}$
17: **end for**
18: `// Convert` $\boldsymbol{S}(d+1)$ `to` $\boldsymbol{\pi}^{\mathrm{out}}$
19: **for** (e,x) in $\boldsymbol{S}(d+1)$ **do**
20: $\quad$ `// The` e`-th element of` $\boldsymbol{\pi}^{\mathrm{out}}$ `is` x
21: $\quad \pi^{\mathrm{out}}_e = x$
22: **end for**
23: **return** $\boldsymbol{\pi}^{\mathrm{out}}$

4.2 Theoretical Analysis of CSA Algorithm

Here, we first discuss an upper bound of the probability of error. Then, we discuss the number of times we call the offline oracle.

4.2.1 An Upper Bound of the Probability of Error

Here, we show an upper bound of the probability of error of the CSA algorithm. Let $\Delta_{(1)}, \ldots, \Delta_{(d)}$ be a permutation of $\Delta_1, \ldots, \Delta_d$ such that $\Delta_{(1)} \leq \cdots \leq \Delta_{(d)}$. Also, let us define

$$\mathbf{H}_2 = \max_{s \in [d]} \frac{s}{\Delta^2_{(s)}}.$$

One can verify that $\mathbf{H}_2$ is equivalent to $\mathbf{H}$ up to a logarithmic factor: $\mathbf{H}_2 \leq \mathbf{H} \leq \log(2d)\mathbf{H}_2$ (Audibert et al., 2010).

Theorem 2. *Given any* $T > d$*, action class* $\mathcal{A} \subset \mathbb{R}^d$*, and* $\boldsymbol{\mu} \in \mathbb{R}^d$*, the CSA algorithm uses at most* T *samples and outputs a solution* $\boldsymbol{\pi}^{\mathrm{out}} \in \mathcal{A} \cup \{\perp\}$ *such that*

$$\begin{aligned} & \Pr\left[\boldsymbol{\pi}^{\mathrm{out}} \neq \boldsymbol{\pi}^*\right] \\ \leq\ & d^2 \exp\left(-\frac{T-d}{2(2+L^2)^2 R^2 \tilde{\log}(d) U^2_{\mathcal{A}} \mathbf{H}_2}\right), \end{aligned} \quad (8)$$

where $L = \max_{e \in [d], \boldsymbol{\pi}^1, \boldsymbol{\pi}^2, \boldsymbol{\pi}^3 \in \mathcal{A}, \pi^1_e \neq \pi^3_e} \frac{|\pi^1_e - \pi^2_e|}{|\pi^1_e - \pi^3_e|}$, $\tilde{\log}(d) \triangleq \sum_{s=1}^{d} \frac{1}{s}$, *and* $U_{\mathcal{A}} = \max_{\boldsymbol{\pi}, \boldsymbol{\pi}' \in \mathcal{A}, e \in \{s \in [d] \mid \pi_s \neq \pi'_s\}} \frac{\sum_{s=1}^{d} |\pi_s - \pi'_s|}{|\pi_e - \pi'_e|}$.

We can see that the CSA algorithm is optimal up to a logarithmic factor in the exponent. Since $L = 1$ and $U_{\mathcal{A}} = \mathrm{width}(\mathcal{A})$ in the ordinary CPE-MAB, we can confirm that Theorem 2 can be seen as a natural generalization of Theorem 3 in Chen et al. (2014), which shows an upper bound of the probability of error of the CSAR algorithm.

4.2.2 The Oracle Complexity

Next, we discuss the *oracle complexity*, the number of times we call the offline oracle (Ito et al., 2019; Xu and Li, 2021). Note that, in the CSA algorithm, we

call the COracle $\mathcal{O}(d\sum_{s=1}^{d} B_s)$ times (line 12). Therefore, if each COracle call invokes the offline oracle N times, the *oracle complexity* is $\mathcal{O}\left(Nd\sum_{s=1}^{d} B_s\right)$. Finally, if the time complexity of each oracle call is Z, then the total time complexity of the CSA algorithm is $\mathcal{O}\left(ZNd\sum_{s=1}^{d} B_s\right)$.

Below, we discuss the oracle complexity of the CSA algorithm in some specific combinatorial problems such as the knapsack problem and the optimal transport problem. Note that the Peace algorithm (Katz-Samuels et al., 2020) has to enumerate all the actions in advance, where the number may be exponentially large in d. We show that the CSA algorithm mitigates the curse of dimensionality.

The Knapsack Problem (Dantzig and Mazur, 2007). In the knapsack problem, we have d items. Each item $s \in [d]$ has a weight w_s and value v_s. Also, there is a knapsack whose capacity is W in which we put items. Our goal is to maximize the total value of the knapsack, not letting the total weight of the items exceed the capacity of the knapsack. Formally, the optimization problem is given as follows:

$$\begin{aligned} &\text{maximize}_{\boldsymbol{\pi}\in\mathcal{A}} && \textstyle\sum_{s=1}^{d} v_s\pi_s \\ &\text{subject to} && \textstyle\sum_{s=1}^{d} \pi_s w_s \leq W, \end{aligned}$$

where $\pi_s \in \mathbb{Z}_{\geq 0}$ denotes the number of item s in the knapsack. Here, if we assume the weight of each item is known, but the value is unknown, we can apply the R-CPE-MAB framework to the knapsack problem, where we estimate the values of items. The knapsack problem is NP-complete (Garey and Johnson, 1979). Hence, it is unlikely that the knapsack problem can be solved in polynomial time. However, it is well known that the knapsack problem can be solved in pseudo-polynomial time $\mathcal{O}(dW)$ if we use dynamic programming (Kellerer et al., 2004; Fujimoto, 2016). It finds the optimal solution by constructing a table of size dW whose (s, w)-th element represents the maximum total value that can be achieved if the sum of the weights does not exceed w using up to the sth item. In some cases, it is sufficient to assume $\mathcal{O}(dW)$ time-complexity is enough, and therefore, we use this dynamic programming method as the offline oracle. We can construct the *COrcale* for the knapsack problem by calling this offline oracle once (see Appendix B.1 for details). Therefore, the CSA algorithm calls the offline oracle $\mathcal{O}(1 \times d\sum_{s=1}^{d} B_s) = \mathcal{O}(d\sum_{s=1}^{d} B_s)$ times, and the total time complexity of the CSA algorithm is $\mathcal{O}(dW \times d\sum_{s=1}^{d} B_s) = \mathcal{O}(d^2W\sum_{s=1}^{d} B_s)$. This is much more computationally friendly than the Peace algorithm with time complexity $\mathcal{O}\left(\prod_{s=1}^{d} B_s\right)$.

The Optimal Transport (OT) Problem (Peyré and Cuturi, 2019). OT can be regarded as the cheapest plan to deliver resources from m suppliers to n consumers, where each supplier i and consumer j have supply s_i and demand d_j, respectively. Let $\boldsymbol{\gamma} \in \mathbb{R}_{\geq 0}^{m\times n}$ be the cost matrix, where γ_{ij} denotes the cost between supplier i and demander j. Our objective is to find the optimal transportation matrix,

$$\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi}\in\mathcal{G}(\boldsymbol{s},\boldsymbol{d})}{\arg\min} \sum_{i,j} \pi_{ij}\gamma_{ij}, \tag{9}$$

where

$$\mathcal{G}(\boldsymbol{s},\boldsymbol{d}) \triangleq \left\{\boldsymbol{\Pi} \in \mathbb{R}_{\geq 0}^{m\times n} \,\middle|\, \boldsymbol{\Pi}\mathbf{1}_n = \boldsymbol{s}, \boldsymbol{\Pi}^\top \mathbf{1}_m = \boldsymbol{d}\right\}. \tag{10}$$

Here, $\boldsymbol{s} = (s_1, \ldots, s_m)$ and $\boldsymbol{d} = (d_1, \ldots, d_n)$. π_{ij} represents how much resources one sends from supplier i to demander j. If we assume that the cost is unknown and changes stochastically, e.g., due to some traffic congestions, we can apply the R-CPE-MAB framework to the optimal transport problem, where we estimate the cost of each edge (i, j) between supplier i and consumer j.

Once $\boldsymbol{\gamma}$ is given, we can compute $\boldsymbol{\pi}^*$ in $\mathcal{O}(l^3 \log l)$, where $l = \max(m, n)$, by using network simplex or interior point methods (Cuturi, 2013), and we can use them as the offline oracle. It is known that the solution of linear programming can always be found at one of the vertices of the feasible region (Ahmadi, 2016), and therefore the size of the action space $\mathcal{A} = \left\{\boldsymbol{\pi}^{\boldsymbol{\nu},\mathcal{G}(\boldsymbol{s},\boldsymbol{d})} \in \mathbb{R}_{\geq 0}^{m\times n} \mid \forall \boldsymbol{\nu} \in \mathbb{R}^{m\times n}\right\}$ is finite. However, it is difficult to construct $\{\texttt{POSSIBLE-PI}((i, j))\}_{i\in[m],j\in[n]}$ in general to run the CSA algorithm. On the other hand, if $\boldsymbol{s}$ and $\boldsymbol{d}$ are both integer vectors, we can construct $\{\texttt{POSSIBLE-PI}((i, j))\}_{i\in[m],j\in[n]}$ thanks to the fact that $\mathcal{A} = \left\{\boldsymbol{\pi}^{\boldsymbol{\nu},\mathcal{G}(\boldsymbol{s},\boldsymbol{d})} \in \mathbb{Z}_{\geq 0}^{m\times n} \mid \forall \boldsymbol{\nu} \in \mathbb{R}^{m\times n}\right\}$ is a set of non-negative integer matrices (Goodman and O'Rourke, 1997). In this case, all actions are restricted to non-negative integers, and $\texttt{POSSIBLE-PI}(i, j) = \{0, 1, \ldots, \min(s_i, d_j)\}$. In Appendix B.2, we show that the COracle can be constructed by calling the offline oracle once. Therefore, we call the offline oracle $\mathcal{O}(1 \times mn\sum_{s=1}^{mn} B_s) = \mathcal{O}(mn\sum_{s=1}^{mn} B_s)$ times, and the total time complexity is $\mathcal{O}\left(mn\sum_{s=1}^{mn} B_s \cdot l^3 \log l\right)$, where $l = \max(m, n)$. Again, this is much more computationally friendly than the Peace algorithm with time complexity $\mathcal{O}\left(\prod_{s=1}^{d} B_d\right)$.

A General Case When $K(= |\mathcal{A}|) = \text{poly}(d)$. In some cases, we can enumerate all the possible actions in $\mathcal{A}$. For instance, one may use some prior knowledge of each arm, which is sometimes obtainable in the real world, to narrow down the list of actions, and make the size of the action class $\mathcal{A}$ polynomial

in d. In Appendix B.3, we show that the time complexity of the COracle is $\mathcal{O}(dK + K\log K)$, and therefore the total time complexity of the CSA algorithm is $\mathcal{O}\left((dK + K\log K)\cdot d\sum_{s=1}^{d} B_s\right)$.

5 The Minimax-CombSAR Algorithm for R-CPE-MAB where $|\mathcal{A}| = \mathrm{poly}(d)$

In this section, we show an algorithm for fixed-budget R-CPE-MAB named the Minimax Combinatorial Successive Accept (Minimax-CombSAR) algorithm, for the case where we can assume that the size of $\mathcal{A}$ is polynomial in d. Let $\mathcal{A} = \{\boldsymbol{\pi}^1 = \boldsymbol{\pi}^*, \boldsymbol{\pi}^2, \ldots, \boldsymbol{\pi}^K\}$, where $\boldsymbol{\mu}^\top \boldsymbol{\pi}^i \geq \boldsymbol{\mu}^\top \boldsymbol{\pi}^{i+1}$, for all $i \in [K-1]$. The Minimax-CombSAR algorithm is inspired by the Optimal Design-based Linear Best Arm Identification (OD-LinBAI) algorithm Yang and Tan (2022), which eliminates actions in order from those considered suboptimal and finally outputs the remaining action as the optimal action. For the Minimax-CombSAR algorithm, we show an upper bound of the probability of error.

5.1 Minimax-CombSAR Algorithm

We show the Minimax-CombSAR in Algorithm 2. Here, we explain it at a higher level.

We have $\lceil \log d \rceil$ phases in the Minimax-CombSAR algorithm, and it maintains an *active* action set $\mathbb{A}(r)$ in each phase r. In each phase $r \in [\lceil \log d \rceil]$, it pulls arms $m(r) = \frac{T' - d\lceil \log_2 d \rceil}{B/2^{r-1}}$ times in total, where $B = 2^{\lceil \log_2 d \rceil} - 1$ and $\beta \in [0,1]$ is a hyperparameter, and $T' = T - \left\lfloor \frac{T}{d}\beta \right\rfloor \times d$. In each phase, we compute an *allocation vector* $\boldsymbol{p}(r) \in \left\{ \boldsymbol{v} \in \mathbb{R}^d \mid \sum_{s=1}^{d} v_s = 1 \right\} \triangleq \Pi_d$, and pull each arm s $\lceil p_s(r)\cdot m(r) \rceil$ times. Then, at the end of each phase r, it eliminates a subset of possibly suboptimal actions. Eventually, there is only one action $\boldsymbol{\pi}^{\mathrm{out}}$ in the active action set, which is the output of the algorithm.

The key to identifying the best action with high confidence is the choice of the allocation vector $\boldsymbol{p}(r)$, which determines how many times we pull each arm in phase r.

Choice of $\boldsymbol{p}(r)$

We discuss how to choose an allocation vector that is beneficial to identify the best action. Let us denote the number of times arm s is pulled before phase r starts by $T_s(r)$. Also, we denote by $\hat{\mu}_s(r)$ the empirical mean of the reward from arm s before phase r starts. Then, at the end of phase r, from Hoeffding's inequality (Hoeffding, 1963), we have

$$\begin{aligned} &\Pr\left[\left|(\hat{\boldsymbol{\mu}}(r+1) - \boldsymbol{\mu})^\top (\boldsymbol{\pi}^a - \boldsymbol{\pi}^b)\right| \geq \epsilon\right] \\ \leq\ & \exp\left(-\frac{\epsilon^2}{\kappa^{a,b,\boldsymbol{p}}(r) R^2}\right) \end{aligned} \tag{11}$$

for any $\boldsymbol{\pi}^a, \boldsymbol{\pi}^b \in \mathcal{A}$, and $\epsilon \in \mathbb{R}$. Here, $\hat{\boldsymbol{\mu}}(r+1) = (\hat{\mu}_1(r+1), \ldots, \hat{\mu}_d(r+1))^\top$ and

$$\kappa^{a,b,\boldsymbol{p}}(r) = \sum_{s=1}^{d} \frac{\left(\pi_s^a - \pi_s^b\right)^2}{T_s(r) + \lceil p_s(r)\cdot m(r) \rceil}. \tag{12}$$

(11) shows that the empirical difference $\hat{\boldsymbol{\mu}}^\top(\boldsymbol{\pi}^a - \boldsymbol{\pi}^b)$ between $\boldsymbol{\pi}^a$ and $\boldsymbol{\pi}^b$ is closer to the true difference $\boldsymbol{\mu}^\top(\boldsymbol{\pi}^a - \boldsymbol{\pi}^b)$ with high probability if we make $\kappa^{a,b,\boldsymbol{p}}(r)$ small. In that case, we have a higher chance to distinguish whether $\boldsymbol{\pi}^a$ is better than $\boldsymbol{\pi}^b$ or not. However, when we have more than two actions in $\mathbb{A}(r)$,

$$\boldsymbol{p}^{a,b}(r) = \arg\min_{\boldsymbol{p} \in \Pi_d} \kappa^{a,b,\boldsymbol{p}}(r)$$

is not necessarily a good allocation vector for investigating the true difference between other pairs of actions in $\mathbb{A}(r)$. Therefore, we propose the following allocation vector as an alternative:

$$\boldsymbol{p}^{\min}(r) \triangleq \arg\min_{\boldsymbol{p} \in \Pi_d} \max_{\boldsymbol{\pi}^a, \boldsymbol{\pi}^b \in \mathcal{A}} \kappa^{a,b,\boldsymbol{p}}(r). \tag{13}$$

(13) takes a minimax approach, which computes an allocation vector that minimizes the maximum value

Algorithm 2 Minimax Combinatorial Successive Accept Algorithm

Input: Budget: $T \geq 0$, initialization parameter: β, action set: $\mathbb{A}(1) = \mathcal{A}$

// Initialization
for $s \in [d]$ **do**
  Pull arm s $\left\lfloor \frac{T}{d}\beta \right\rfloor$ times and update $\hat{\mu}_s(1)$
  $T_s(r) \leftarrow \left\lfloor \frac{T}{d}\beta \right\rfloor$
end for
$T' \leftarrow T - \left\lfloor \frac{T}{d}\beta \right\rfloor \times d$
for $r = 1$ to $\lceil \log_2 d \rceil$ **do**
  $m(r) = \frac{T' - d\lceil \log_2 d \rceil}{B/2^{r-1}}$
  Compute $\boldsymbol{p}(r)$ according to (13) or (15)
  for $s \in [d]$ **do**
    Pull arm s $\lceil p_s(r)\cdot m(r) \rceil$ times
    Update $\hat{\mu}_s(r+1)$ with the observed samples
  end for
  For each action $\boldsymbol{\pi} \in \mathbb{A}(r)$, estimate the expected reward: $\langle \hat{\boldsymbol{\mu}}(r+1), \boldsymbol{\pi} \rangle$
  Let $\mathbb{A}(r+1)$ be the set of $\left\lceil \frac{d}{2^r} \right\rceil$ actions in $\mathbb{A}(r)$ with the largest estimates of the expected rewards
end for

Output: The only action $\boldsymbol{\pi}^{\mathrm{out}}$ in $\mathbb{A}(\lceil \log_2 d \rceil + 1)$

of the right-hand side of (11) among all of the pairs of actions in $\mathbb{A}(r)$. Since (13) is a d-dimensional non-linear optimization problem, it becomes computationally costly as d grows. Thus, another possible choice of the allocation vector is as follows:

$$\boldsymbol{q}^{\min}(r) \triangleq \arg\min_{\boldsymbol{p}\in\Pi_d} \max_{\boldsymbol{\pi}^a,\boldsymbol{\pi}^b\in\mathcal{A}} \lambda^{a,b,\boldsymbol{p}}(r), \quad (14)$$

where $\lambda^{a,b,\boldsymbol{p}} = \sum_{s=1}^{d} \frac{\left(\pi_s^a-\pi_s^b\right)^2}{p_s(r)\cdot m(r)}$. For specific actions $\boldsymbol{\pi}^k, \boldsymbol{\pi}^l \in \mathcal{A}$, from the method of Lagrange multipliers (Hoffmann and Bradley, 2009), we have

$$\begin{aligned}\boldsymbol{q}^{k,l}(r) &\triangleq \arg\min_{\boldsymbol{p}\in\Pi_d} \lambda^{k,l,\boldsymbol{p}}(r) \\ &= \left(\frac{\left|\pi_1^k-\pi_1^l\right|}{\sum_{s=1}^d |\pi_s^k-\pi_s^l|}, \ldots, \frac{\left|\pi_d^k-\pi_d^l\right|}{\sum_{s=1}^d |\pi_s^k-\pi_s^l|}\right)^\top,\end{aligned}$$

and therefore, $\boldsymbol{q}^{\min}(r)$ in (14) can be written explicitly as follows:

$$\begin{aligned}\boldsymbol{q}^{\min}(r) &= \boldsymbol{q}^{i,j}(r) \\ &= \left(\frac{\left|\pi_1^i-\pi_1^j\right|}{\sum_{s=1}^d \left|\pi_s^i-\pi_s^j\right|}, \ldots, \frac{\left|\pi_d^i-\pi_d^j\right|}{\sum_{s=1}^d \left|\pi_s^i-\pi_s^j\right|}\right)^\top,\end{aligned} \quad (15)$$

where $\boldsymbol{\pi}^i, \boldsymbol{\pi}^j = \arg\max_{\boldsymbol{\pi}^k,\boldsymbol{\pi}^l\in\mathcal{A}} \lambda^{k,l,\boldsymbol{p}}(r)$. If we use (15) for the allocation vector instead of computing (13), we do not have to solve a d-dimensional non-linear optimization problem, and thus is computationally more friendly.

In some cases, actions in $\mathcal{A}$ can be sparse, and $\boldsymbol{p}(1)$ can also be sparse. If $\boldsymbol{p}(1)$ is sparse, we may only have a few samples for some arms, and therefore accidentally eliminate the best action in the first phase in line 15 of Algorithm 2. To cope with this problem, we have the *initialization* phase (lines 2–5) to pull each arm $\left\lfloor \frac{T}{d}\beta \right\rfloor$ times, where $\beta \in [0,1]$ is a hyperparameter. Intuitively, β represents how much of the total budget will be spent on the initialization phase. If β is too small, we may accidentally eliminate the best action in the early phase, and if it is too large, we may not have enough budget to distinguish between the best action and the next best action.

5.2 Theoretical Analysis of Minimax-CombSAR

Here, in Theorem 3, we show an upper bound of the probability of error of the Minimax-CombSAR algorithm.

Theorem 3. *For any problem instance in fixed-budget R-CPE-MAB, the Minimax-CombSAR algorithm outputs an action* $\boldsymbol{\pi}^{\text{out}}$ *satisfying*

$$\begin{aligned}&\Pr\left[\boldsymbol{\pi}^{\text{out}} \neq \boldsymbol{\pi}^*\right] \\ &\leq \left(\frac{4K}{d} + 3\log_2 d\right) \exp\left(-\frac{T' - \lceil \log_2 d \rceil}{R^2 V^2} \cdot \frac{1}{\mathbf{H}_2}\right),\end{aligned} \quad (16)$$

where $V = \max_{\boldsymbol{\pi}^i\in\mathcal{A}\setminus\{\boldsymbol{\pi}^*\}} \left(\sum_{s=1}^{d} \frac{\left|\pi_s^1-\pi_s^i\right|}{\left|\pi_{s(i)}^1-\pi_{s(i)}^i\right|}\right)^2$ *and* $s(i) = \arg\max_{s\in[d]} \left|\pi_s^1-\pi_s^i\right|$.

We can see that the Minimax-CombSAR algorithm is an optimal algorithm whose upper bound of the probability of error matches the lower bound shown in (7) since we have $\mathbf{H}_2 \leq \mathbf{H} \leq \log(2d)\mathbf{H}_2$.

5.3 Computational Complexity

We discuss the time complexity of the MinMax-CombSAR algorithm. We can compute the allocation vector $\boldsymbol{p}(r)$ with time complexity of $\mathcal{O}(1)$ if we use Equation (15) (line 9). The time complexity of the elimination procedure for $\mathcal{A}$ in lines 14 and 15 is of $\mathcal{O}(K\log K)$. In total, since we have $\lceil \log d \rceil$ phases, the time complexity is $\mathcal{O}\left(K \log K \lceil \log d \rceil\right)$.

5.4 Comparison with Katz-Samuels et al. (2020)

Here, we compare the Minimax-CombSAR algorithm with the Peace algorithm introduced in Katz-Samuels et al. (2020). The upper bound of the Peace algorithm can be written as follows:

$$\begin{aligned}&\Pr\left[\boldsymbol{\pi}^{\text{out}} \neq \boldsymbol{\pi}^*\right] \\ &\leq 2\lceil \log(d) \rceil \exp\left(-\frac{T}{c' \log(d)\left(\gamma_* + \rho_*\right)}\right),\end{aligned} \quad (17)$$

where

$$\gamma_* = \inf_{\boldsymbol{\lambda}\in\Pi_d} \mathbb{E}_{\boldsymbol{\eta}\sim\mathcal{N}(0,I)} \left[\sup_{\boldsymbol{\pi}\in\mathcal{A}\setminus\{\boldsymbol{\pi}^*\}} \frac{\left(\boldsymbol{\pi}^*-\boldsymbol{\pi}\right)^\top \boldsymbol{A}\left(\lambda\right)^{\frac{1}{2}} \boldsymbol{\eta}}{\boldsymbol{\mu}^\top\left(\boldsymbol{\pi}^*-\boldsymbol{\pi}\right)}\right],$$

$$\rho_* = \min_{\boldsymbol{\lambda}\in\Pi_d} \max_{\boldsymbol{\pi}\in\mathcal{A}\setminus\{\boldsymbol{\pi}^*\}} \frac{\sum_{s=1}^{d} \frac{|\pi_s^*-\pi_s|^2}{\lambda_s}}{\Delta^2_{\boldsymbol{\pi}^*,\boldsymbol{\pi}}},$$

and c' is a problem-dependent constant. Here, $\boldsymbol{A}\left(\boldsymbol{\lambda}\right)$ is a diagonal matrix whose (i,i) element is λ_i. In general, it is not clear whether the upper bound of the Minimax-CombSAR algorithm in (2) is tighter than that of the Peace algorithm shown in (17). In the experiment section, we compare the two algorithms numerically and show that our algorithm outperforms the Peace algorithm.

Table 1: The percentage of correctly identified optimal actions in the knapsack problem. The size of the action set is exponentially large in d. We conducted experiments for 100 times for each d.

d	10	20	30	40	50	60	70	80	90	100
Naive Baseline	93 %	82 %	63 %	65 %	49 %	47 %	33 %	30 %	24 %	15 %
CSA	98 %	84 %	77 %	70 %	55 %	49 %	51 %	50 %	35 %	32 %

Table 2: Comparison of the percentage of the best actions correctly identified by the CSA, Minimax-CombSAR, and Peace algorithms in the knapsack problem. We assume that $|\mathcal{A}| \leq 1000$ is guaranteed. The horizontal axis represents the number of items d. We ran the experiments a hundred times for each d.

d	10	15	20	25	30	35	40	45	50
Peace	97 %	86 %	88 %	78 %	74 %	71 %	58 %	58 %	55 %
CSA	98 %	96 %	93 %	87 %	83 %	81 %	80 %	73 %	73 %
MinMaxCombSAR($\beta = 0.2$)	98 %	93 %	95 %	89 %	93 %	87 %	83 %	85 %	83 %
MinMaxCombSAR($\beta = 0.2$)	99 %	94 %	95 %	92 %	93 %	85 %	86 %	79 %	83 %

6 Experiment

In this section, we numerically compare the CSA, Minimax-CombSAR, and Peace algorithms. First, we run an experiment where we assume $|\mathcal{A}|$ is exponentially large in d. We see the performance of the CSA algorithm with different budgets. Also, we compare the CSA algorithm with a naive baseline method. Next, we compare the CSA, Minimax-CombSAR, and Peace algorithms, where we can assume that $|\mathcal{A}|$ is polynomial in d.

6.1 When $|\mathcal{A}|$ is Exponentially Large in d

Here, the goal is to identify the best action for the knapsack problem. In the knapsack problem, we have d items. Each item $s \in [d]$ has a weight w_s and value v_s. Also, there is a knapsack whose capacity is W in which we put items. Our goal is to maximize the total value of the knapsack, not letting the total weight of the items exceed the capacity of the knapsack. Formally, the optimization problem is given as follows:

$$\begin{aligned} &\text{maximize}_{\boldsymbol{\pi} \in \mathcal{A}} && \textstyle\sum_{s=1}^{d} v_s \pi_s \\ &\text{subject to} && \textstyle\sum_{s=1}^{d} \pi_s w_s \leq W, \end{aligned}$$

where π_s denotes the number of item s in the knapsack. Here, the weight of each item is known, but the value is unknown, and therefore has to be estimated. In each time step, the player chooses an item s and gets an observation of value v_s, which can be regarded as a random variable from an unknown distribution with mean v_s.

Here, we assume $|\mathcal{A}|$ is exponentially large in d. Therefore, we can not use the Minimax-CombSAR and the Peace algorithm since they have to enumerate all the possible actions. For our experiment, we generated the weight of each item s, w_s, uniformly from $\{1, 2, \ldots, 200\}$. For each item s, we generate v_s as $v_s = w_s \times x$, where x is a uniform sample from $[1.0, 1.1]$. The capacity of the knapsack is $W = 200$. Each time we choose an item s, we observe a value $v_s + \epsilon$ where ϵ is a noise from $\mathcal{N}(0, 1)$. We set $R = 1$.

We compare the CSA algorithm to a naive baseline, which pulls each arm equally and returns the action with the largest reward using the sample mean. The budget T is set to 10000. The results are shown in Table 1. We can see that the CSA algorithm outperforms the baseline method in all d.

6.2 When $|\mathcal{A}|$ is Polynomial in d.

Here, we assume that we have prior knowledge of the rewards of arms, i.e., knowledge of the values of the items. We assume that we know each v_s is in $[w_s, 1.1 \times w_s]$, and use this prior knowledge to generate the action class $\mathcal{A}$ in the following procedure. We first generate a vector $\boldsymbol{v}'$ whose s-th element v'_s is uniformly sampled from $[w_s, 1.1 \times w_s]$, and then solve the knapsack problem with v'_s and adde the obtained solution $\boldsymbol{\pi}$ to $\mathcal{A}$. We repeate this 1000 times; therefore $|\mathcal{A}| \leq 1000$. In this experiment, the budget is $T = 20000$. For the Minimax-CombSAR algorithm, we set $\beta = 0.2, 0.4$. We run forty experiments for each $d \in \{10, 15, \ldots, 100\}$. We show the result in Table 2. We can see that the Minimax-CombSAR algorithm outperforms the other two algorithms for almost every d. Also, the CSA algorithm outperforms the Peace algorithm for almost every d.

Next, we compare the performance of each method by conducting an experiment on the best action identification task on the optimal transport problem. We use the notation introduced in Section 4.2.2. We set the supplier vector and the demand vector as

Table 3: Comparison of the percentage of the best actions correctly identified by the CSA, Minimax-CombSAR, and Peace algorithms in the optimal transport problem. We assume that $|\mathcal{A}| \leq 100$ is guaranteed. We show the accuracy of identifying the optimal action for $d = \{4, 9, 16, 25, 36, 49, 64, 81, 100\}$. For each d, we conducted the experiment for 100 times.

d	4	9	16	25	36	49	64	81	100
Peace	84 %	67 %	58 %	49 %	41 %	35 %	36 %	24 %	11 %
CSA	84 %	75 %	57 %	46 %	50 %	38 %	31 %	39 %	15 %
MinMax-CombSAR ($\beta = 0.2$)	82 %	83 %	73 %	67 %	59 %	54 %	49 %	48 %	29 %

$\boldsymbol{s} = (1, 2, \ldots, m)$ and $\boldsymbol{d} = (1, 2, \ldots, n)$, respectively. Also, we set $m = n$. We generate each element of the cost matrix $\boldsymbol{\gamma}$ from a uniform distribution $U(0, 1)$. Here, we assume that $\boldsymbol{\gamma}$ is unknown, and therefore has to be estimated. Note that the number of arms d is $mn = m^2$ in this setting. To generate the action set, we follow the following procedure. We first generate a random matrix $\boldsymbol{\gamma}'$, where each element is sampled from $U(0, 1)$. Then, we solve the optimal transport problem with the offline oracle and add the obtained solution $\boldsymbol{\pi}'$ to $\mathcal{A}$. We repeat this procedure for 100 times, and therefore $|\mathcal{A}| \leq 100$. We show the results in Table 3. We can see that the MinMax-CombSAR algorithm outperforms the other two methods.

7 Conclusion

In this paper, we studied the fixed-budget R-CPE-MAB. We first introduced the CSA algorithm, which is the first algorithm that can identify the best action even when the size of the action class is exponentially large with respect to the number of arms. However, it still has an extra logarithmic term in the exponent. Then, we proposed an optimal algorithm named the Minimax-CombSAR algorithm, which, although it is applicable only when the action class is polynomial, matches a lower bound. We showed that both of the algorithms outperform the existing methods.

Acknowledgement

We thank Dr. Kevin Jamieson for his very helpful advice and comments on existing studies. SN was supported by JST SPRING, Grant Number JPMJSP2108. MS was supported by the Institute for AI and Beyond, UTokyo.

References

Ahmadi, A. A. (2016). Lec11p1, ORF363/COS323.

Audibert, J.-Y., Bubeck, S., and Munos, R. (2010). Best arm identification in multi-armed bandits. In *The 23rd Conference on Learning Theory*, pages 41–53.

Audibert, J.-Y., Munos, R., and Szepesvári, C. (2009). Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theor. Comput. Sci.*, 410:1876–1902.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256.

Bubeck, S., Munos, R., and Stoltz, G. (2009). Pure exploration in multi-armed bandits problems. In *International Conference on Algorithmic Learning Theory*.

Carpentier, A. and Locatelli, A. (2016). Tight (lower) bounds for the fixed budget best arm identification bandit problem. In *Annual Conference Computational Learning Theory*.

Chen, L., Gupta, A., and Li, J. (2016). Pure exploration of multi-armed bandit under matroid constraints. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 647–669. JMLR.org.

Chen, L., Gupta, A., Li, J., Qiao, M., and Wang, R. (2017). Nearly optimal sampling algorithms for combinatorial pure exploration. In Kale, S. and Shamir, O., editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 482–534. PMLR.

Chen, S., Lin, T., King, I., Lyu, M. R., and Chen, W. (2014). Combinatorial pure exploration of multi-armed bandits. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, page 379–387, Cambridge, MA, USA. MIT Press.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Dantzig, T. and Mazur, J. (2007). *Number: The Language of Science*. A Plume book. Penguin Publishing Group.

Du, Y., Kuroki, Y., and Chen, W. (2021). Combinatorial pure exploration with bottleneck reward function. In *Advances in Neural Information Processing Systems*, volume 34, pages 23956–23967. Curran Associates, Inc.

Fujimoto, N. (2016). A pseudo-polynomial time algorithm for solving the knapsack problem in polynomial space. In Chan, T.-H. H., Li, M., and Wang, L., editors, *Combinatorial Optimization and Applications*, pages 624–638, Cham. Springer International Publishing.

Gabillon, V., Lazaric, A., Ghavamzadeh, M., Ortner, R., and Bartlett, P. (2016). Improved learning complexity in combinatorial pure exploration bandits. In Gretton, A. and Robert, C. C., editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 1004–1012, Cadiz, Spain. PMLR.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, first edition edition.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA.

Gibbons, A. (1985). *Algorithmic Graph Theory*. Cambridge University Press.

Goodman, J. E. and O'Rourke, J., editors (1997). *Handbook of Discrete and Computational Geometry*. CRC Press, Inc., USA.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30.

Hoffmann, L. and Bradley, G. (2009). *Calculus for Business, Economics, and the Social and Life Sciences, Brief*. McGraw-Hill Companies,Incorporated.

Ito, S., Hatano, D., Sumita, H., Takemura, K., Fukunaga, T., Kakimura, N., and Kawarabayashi, K.-I. (2019). Oracle-efficient algorithms for online linear optimization with bandit feedback. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Kalyanakrishnan, S. and Stone, P. (2010). Efficient selection of multiple bandit arms: Theory and practice. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 511–518, Madison, WI, USA. Omnipress.

Katz-Samuels, J., Jain, L., Karnin, Z., and Jamieson, K. (2020). An empirical process approach to the union bound: Practical algorithms for combinatorial and linear bandits. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer, Berlin, Germany.

Nakamura, S. and Sugiyama, M. (2023). Thompson sampling for real-valued combinatorial pure exploration of multi-armed bandit. In *The 38th Annual AAAI Conference on Artificial Intelligence*.

Pettie, S. and Ramachandran, V. (2002). An optimal minimum spanning tree algorithm. *J. ACM*, 49(1):16–34.

Peyré, G. and Cuturi, M. (2019). Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607.

Pochet, Y. and Wolsey, L. A. (2010). *Production Planning by Mixed Integer Programming*. Springer Publishing Company, Incorporated, 1st edition.

Rinaldo, A. and Bong, H. (2018). 36-710: Advanced statistical theory. lecture 3.

Rivasplata, O. (2012). Subgaussian random variables : An expository note.

Sniedovich, M. (2006). Dijkstra's algorithm revisited: the dynamic programming connexion. *Control and Cybernetics*, 35:599–620.

Villani, C. (2008). *Optimal transport – Old and new*, volume 338, pages xxii+973.

Wang, S. and Zhu, J. (2022a). Thompson sampling for (combinatorial) pure exploration. In *Proceedings of the 39 th International Conference on Machine Learning*, Baltimore, Maryland, USA.

Wang, S. and Zhu, J. (2022b). Thompson sampling for (combinatorial) pure exploration. In *Proceedings of the 39 th International Conference on Machine Learning*, Baltimore, Maryland, USA.

Xu, H. and Li, J. (2021). Simple combinatorial algorithms for combinatorial bandits: corruptions and approximations. In de Campos, C. P., Maathuis, M. H., and Quaeghebeur, E., editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI 2021, Virtual Event, 27-30 July 2021*, volume 161 of *Proceedings of Machine Learning Research*, pages 1444–1454. AUAI Press.

Yang, J. and Tan, V. (2022). Minimax optimal fixed-budget best arm identification in linear bandits. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

Checklist

The checklist follows the references. For each question, choose your answer from the three possible options: Yes, No, Not Applicable. You are encouraged to include a justification to your answer, either by referencing the appropriate section of your paper or providing a brief inline description (1-2 sentences). Please do not modify the questions. Note that the Checklist section does not count towards the page limit. Not including the checklist in the first submission won't result in desk rejection, although in such case we will ask you to upload it during the author response period and include it in camera ready (if accepted).

In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Not Applicable]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A Proof of Theorem 1

Here, we prove Theorem 1. For the reader's convenience, we restate the Theorem.

Theorem 1. *For any action class and any algorithm that returns an action $\boldsymbol{\pi}^{\mathrm{out}}$ after T times of arm pulls, the probability of error is at least*

$$\mathcal{O}\left(\exp\left(-\frac{T}{\mathbf{H}}\right)\right). \tag{7}$$

We follow a similar discussion to Carpentier and Locatelli (2016). This is a lower bound that will hold in the much easier problem where the learner knows that the bandit setting she is facing is one of only d given bandit settings. This lower bound ensures that even in this much simpler case, the learner will make a mistake.

Let $\mathcal{N}(\mu, 1)$ denote the Gaussian distribution of mean μ and unit variance. For any $k \in [d]$, we write $\nu_k := \mathcal{N}(\mu_k, 1)$. Also, for any $k \in [d]$, we define ν'_k as follows:

$$\nu'_k := \begin{cases} \mathcal{N}(\mu_k + 2\Delta_k, 1) & (\text{if } \pi^*_k < \pi^{(k)}_k) \\ \mathcal{N}(\mu_k - 2\Delta_k, 1) & (\text{if } \pi^*_k > \pi^{(k)}_k) \end{cases}.$$

Recall that $\Delta_k = \frac{\boldsymbol{\mu}^\top(\boldsymbol{\pi}^* - \boldsymbol{\pi}^{(k)})}{\left|\pi^*_k - \pi^{(k)}_k\right|}$ and $\boldsymbol{\pi}^{(k)} = \underset{\boldsymbol{\pi} \neq \boldsymbol{\pi}^*}{\arg\min} \frac{\boldsymbol{\mu}^\top(\boldsymbol{\pi}^* - \boldsymbol{\pi})}{\left|\pi^*_k - \pi_k\right|}$.

For any $s \in [d]$, we define the product distributions $\mathcal{G}^s$ as $\nu^s_1 \otimes \cdots \otimes \nu^s_d$, where for $1 \leq k \leq d$,

$$\nu^s_k := \nu_k \mathbf{1}\left[k \neq s\right] + \nu'_k \mathbf{1}\left[k = s\right]. \tag{18}$$

The bandit problem associated with distribution $\mathcal{G}^s$, and that we call "the bandit problem s" is such that, for any $1 \leq k \leq d$, arm k has distribution ν^s_k, i.e., all arms have distribution ν_k except for arm s that has distribution ν'_k. We denote by $\boldsymbol{\mu}^{(s)}$ the vector of expected rewards of bandit problem s, i.e., $\mu^{(s)}_k = \mu_k$ for $k \neq s$, $\mu^{(s)}_k = \mu_k + 2\Delta_k$ for $k = s$ and $\pi^*_k < \pi^{(k)}_k$, $\mu^{(s)}_k = \mu_k - 2\Delta_k$ for $k = s$ and $\pi^*_k > \pi^{(k)}_k$. For any $1 \leq s \leq d$, $\mathbb{P}_s := \mathbb{P}_{(\mathcal{G}^s)^{\otimes T}}$ for the probability distribution of the bandit problem s according to all the samples that a strategy could possibly collect up to horizon T, i.e., according to the samples $(X_{k,u})_{1 \leq k \leq d, 1 \leq u \leq T} \sim (\mathcal{G}^s)^{\otimes T}$. Also, we define $\mathbb{P}_0 := \mathbb{P}_{(\mathcal{G}^0)^{\otimes T}}$, where $\mathcal{G}^0 := \nu_1 \otimes \cdots \otimes \nu_d$.

In the bandit problem s, $\boldsymbol{\pi}^*$ is no longer the best action since the difference of the expected reward between $\boldsymbol{\pi}^*$ and $\boldsymbol{\pi}^{(s)}$ is

$$\begin{aligned} & \sum_{e \in [d] \setminus s} \mu_e(\pi^*_e - \pi^{(s)}_e) - 2\Delta_s \times \left|\pi^*_s - \pi^{(s)}_s\right| \\ \leq \; & -\boldsymbol{\mu}^\top\left(\boldsymbol{\pi}^* - \boldsymbol{\pi}^{(s)}\right) < 0. \end{aligned} \tag{19}$$

We denote by $\boldsymbol{\pi}^{*,(s)}$ the best action in the bandit problem s.

Now, we prove Theorem 1.

Step 1: Definition of KL-divergence

For two distributions ν and ν' defined on $\mathbb{R}$, and that are such that ν is absolutely continuous with respect to ν', we write

$$\mathrm{KL}(\nu, \nu') = \int_{\mathbb{R}} \log\left(\frac{d\nu(x)}{d\nu'(x)}\right) d\nu(x), \tag{20}$$

for the Kullback Leibler divergence between distribution ν and ν'. For any $k \in [d]$, let us write

$$\mathrm{KL}_k \triangleq \mathrm{KL}(\nu_k, \nu'_k) = \frac{1}{2}\left(\mu_a - \mu_b\right)^2. \tag{21}$$

for the Kullback-Leibler divergence between two Gaussian distributions $\mathcal{N}(\mu_a, 1)$ and $\mathcal{N}(\mu_b, 1)$.

Let $1 \leq t \leq T$. Also, let us fix $e \in [d]$, and think of bandit instance e. We define the quantity:

$$\begin{aligned}\widehat{\mathrm{KL}}_{s,t} &= \frac{1}{t}\sum_{u=1}^{t}\log\left(\frac{d\nu_s}{d\nu'_s}\left(X_{s,u}\right)\right) \\ &= \begin{cases}\frac{1}{t}\sum_{u=1}^{t} -\Delta_s\left(X_{s,u}-\mu_s-2\Delta_s\right) & \text{if} \pi_s^* > \pi_s^{(s)} \text{ and } e \neq s, \\ \frac{1}{t}\sum_{u=1}^{t} \Delta_s\left(X_{s,u}-\mu_s+2\Delta_s\right) & \text{if } \pi_s^* < \pi_s^{(s)} \text{and } e \neq s, \\ 0 & \text{if } e = s\end{cases} \end{aligned} \tag{22}$$

where $X_{s,u} \sim_{\text{i.i.d}} \nu_s^e$ for any $u \leq t$.

Next, let us define the following event:

$$\xi = \left\{\forall 1 \leq s \leq d, \forall 1 \leq t \leq T, \left|\widehat{\mathrm{KL}}_{s,t}\right| - \mathrm{KL}_s \leq \sqrt{2\Delta_s^2\frac{\log(12Td)}{t}}\right\}. \tag{23}$$

The following lemma shows a concentration bound for $\left|\widehat{\mathrm{KL}}_{s,t}\right|$.

Lemma 1. *It holds that*

$$\Pr_e\left[\xi\right] \geq \frac{5}{6} \tag{24}$$

Proof. When $e = s$, $\left|\widehat{\mathrm{KL}}_{s,t}\right| - \mathrm{KL}_s \leq \sqrt{2\Delta_s^2\frac{\log(12Td)}{t}}$ holds for any t. Therefore, we think of $e \neq s$. Since $X_{s,t}$ is a 1-sub-Gaussian random variable, we can see that $\widehat{\mathrm{KL}}_{s,t}$ is a $\frac{{\Delta_s}^2}{t}$-sub-Gaussian random variable from (22). We can apply the Hoeffding's inequality to this quantity, and we have that with probability larger than $1 - \frac{1}{6dT}$,

$$\left|\widehat{\mathrm{KL}}_{s,t}\right| - \mathrm{KL}_s \leq \sqrt{\frac{2\Delta_s^2\log(12Td)}{t}} \tag{25}$$

□

Step2: A change of measure

Let $\mathcal{ALG}$ denote the active strategy of the learner, that returns some action $\boldsymbol{\pi}_{\text{out}}$ after pulling arms T times in total. Let $(T_s)_{1\leq s\leq T}$ denote the number of samples collected by $\mathcal{ALG}$ on each arm. These quantities are stochastic but it holds that $\sum_{s=1}^{d} T_s = T$. For any $1 \leq s \leq d$, let us write

$$t_s = \mathbb{E}_0[T_s]. \tag{26}$$

It holds also that $\sum_{s=1}^{d} t_s = T$.

We recall the change of measure identity (Audibert et al., 2010), which states that for any measurable event $\mathcal{E}$ and for any $1 \leq s \leq d$,

$$\Pr_s\left[\mathcal{E}\right] = \mathbb{E}_0\left[\mathbf{1}\left[\mathcal{E}\right]\exp\left(-T_s\widehat{\mathrm{KL}}_{s,T_s}\right)\right], \tag{27}$$

where $\mathbf{1}$ denotes the indicator function.

Then, we consider the following event:

$$\mathcal{E}_s = \left\{\boldsymbol{\pi}_{\text{out}} = \boldsymbol{\pi}^*\right\} \cap \left\{\xi\right\} \cap \left\{T_s \leq 6t_s\right\}, \tag{28}$$

i.e., the event where the algorithm outputs action $\boldsymbol{\pi}^*$ at the end, where ξ holds, and where the number of times arm s is pulled is smaller than $6t_s$. From (28), we have

$$\begin{aligned}\Pr_s[\mathcal{E}_s] &= \mathbb{E}_0\left[\mathbf{1}\left[\mathcal{E}_s\right]\exp\left(-T_s\widehat{\mathrm{KL}}_{s,T_s}\right)\right] &(29)\\ &\geq \mathbb{E}_0\left[\mathbf{1}\left[\mathcal{E}_s\right]\exp\left(-T_s\mathrm{KL}_s - \sqrt{2T_s\Delta_s^2\log(12Td)}\right)\right] &(30)\\ &\geq \mathbb{E}_0\left[\mathbf{1}\left[\mathcal{E}_s\right]\exp\left(-6t_s\mathrm{KL}_s - \sqrt{2T\Delta_s^2\log(12Td)}\right)\right] &(31)\\ &\geq \exp\left(-6t_s\mathrm{KL}_s - \sqrt{2T\Delta_s^2\log(12Td)}\right)\Pr_0\left[\mathcal{E}_s\right], &(32)\end{aligned}$$

since on $\mathcal{E}_s$, we have that ξ holds and that $T_s \leq 6t_s$, and $\mathbb{E}_0\left[\widehat{\mathrm{KL}}_{s,t}\right] = \mathrm{KL}_s$ for any $t \leq T$.

Step 3: Lower bound on $\Pr_0[\mathcal{E}_1]$ for any reasonable algorithm
Assume that the algorithm $\mathcal{ALG}$ is a δ-correct algorithm. Then, we have $\Pr_0[\boldsymbol{\pi}_{\mathrm{out}} \neq \boldsymbol{\pi}^*] \leq \delta$. From the Markov's inequality, we have

$$\Pr_0[T_s \geq 6t_s] \leq \frac{\mathbb{E}_0 T_s}{6t_s} = \frac{1}{6}, \tag{33}$$

since $\mathbb{E}_0[T_s] = t_s$ for algorithm $\mathcal{ALG}$.

Combining Lemma 1, (33), and the fact that $\mathcal{ALG}$ is a δ-correct algorithm, by the union bound, we have

$$\Pr_0[\mathcal{E}_s] \geq 1 - (1/6 + \delta + 1/6) = \frac{2}{3} - \delta. \tag{34}$$

This fact combined with (32), (21), and the fact that $\Pr_s\left[\boldsymbol{\pi}_{\mathrm{out}} \neq \boldsymbol{\pi}^{*,(s)}\right] \geq \Pr_s[\mathcal{E}_s]$, we have

$$\begin{aligned}\Pr_s\left[\boldsymbol{\pi}_{\mathrm{out}} \neq \boldsymbol{\pi}^{*,(s)}\right] &\geq \left(\frac{2}{3} - \delta\right)\exp\left(-6t_s\mathrm{KL}_s - \sqrt{2T\Delta_s^2\log(12Td)}\right)\\ &= \left(\frac{2}{3} - \delta\right)\exp\left(-12t_s(\Delta_s)^2 - \sqrt{2T\Delta_s^2\log(12Td)}\right)\end{aligned}$$

since $\mathrm{KL}_s = 2\Delta_s^2$.

Step 4: Conclusion. Since $\mathbf{H} = \sum_{s=1}^{d}\left(\frac{1}{\Delta_s}\right)^2$ and $\sum_{s=1}^{d} t_s = T$, there exists $e \in [d]$ such that

$$t_e \leq \frac{T}{\mathbf{H}\Delta_e^2}, \tag{35}$$

as the contradiction yields an immediate contradiction. For this e, it holds by (35) that

$$\Pr_e\left[\boldsymbol{\pi}_{\mathrm{out}} \neq \boldsymbol{\pi}^{*,(e)}\right] \geq \left(\frac{2}{3} - \delta\right)\exp\left(-12\frac{T}{\mathbf{H}} - 2\sqrt{T\Delta_e^2\log(6Td)}\right). \tag{36}$$

This concludes the proof.

B How to construct COracles

Here, we show how to construct COracles once $\boldsymbol{\mu}$ and $\boldsymbol{S}(t)$ are given for some specific combinatorial problems.

B.1 COracle for the Knapsack Problem

Here, we show that we can construct the *COracle* for the knapsack problem by calling the *offline oracle* once. Let $\boldsymbol{S}^1(t)$ be the set that collects only the first component of each of all elements in $\boldsymbol{S}(t)$. Also, let $W_{\mathrm{done}} = \sum_{(e,x)\in\boldsymbol{S}(t)} x w_e$. If $W - W_{\mathrm{done}} < 0$, the COracle outputs $\perp$. Otherwise, we call the *offline oracle* and solve the following optimization problem:

$$\begin{aligned}&\text{maximize}_{\boldsymbol{\xi}} && \sum_{s\in[d]\setminus\boldsymbol{S}^1(t)} \mu_s\xi_s\\ &\text{subject to} && \sum_{s\in\boldsymbol{S}^1(t)} \xi_s w_s \leq W - W_{\mathrm{done}}.\end{aligned}$$

Then, we return $\mathrm{COracle}(\boldsymbol{\mu}, \boldsymbol{S}(t)) = \boldsymbol{S}(t) \cup \boldsymbol{\xi}$.

B.2 COracle for the Optimal Transport Problem

Here, in Algorithm 3, we show the *COracle* for the optimal transport problem, which calls the *offline oracle* once. First, let us define $\boldsymbol{S}^1(t)$ as the set that collects only the first element of each of all elements in $\boldsymbol{S}(t)$. Then, let us define $\boldsymbol{\pi}^{\mathrm{CO}}$ as the final output of the COracle. Also, let us define $\boldsymbol{s}'$ and $\boldsymbol{d}'$ as follows:

$$\forall i \in [m], j \in [n], x \in \texttt{POSSIBLE-PI}((i,j)),\ s'_i = \begin{cases} x & (\text{if } ((i,j),x) \in \boldsymbol{S}(t), \\ 0 & (\text{otherwise}), \end{cases}$$

and

$$\forall i \in [m], j \in [n], x \in \texttt{POSSIBLE-PI}((i,j)),\ d'_j = \begin{cases} x & (\text{if } ((i,j),x) \in \boldsymbol{S}(t), \\ 0 & (\text{otherwise}). \end{cases}$$

Also, let us define $\boldsymbol{\pi}' \in \mathbb{R}^{m \times n}$ whose (i,j)-th element is defined as follows:

$$\forall i \in [m], j \in [n], x \in \texttt{POSSIBLE-PI}((i,j)),\ \boldsymbol{\pi}'_{ij} = \begin{cases} x & (\text{if } ((i,j),x) \in \boldsymbol{S}(t), \\ 0 & (\text{otherwise}). \end{cases}$$

Intuitively, for any $((i,j),x) \in \boldsymbol{S}(t)$, we have to send resources from supplier i to demander j for a value of $s'_i (= d'_j)$. This means that we have resources of $s_i - s'_i$ left to send from supplier i, and $d_j - d'_j$ left to send to demander j. Since we can not send a negative value of resources, $s_i - s'_i \geq 0$ and $d_j - d'_j \geq 0$ have to be satisfied.

From the above discussion, we can formally write $\boldsymbol{\pi}^{\mathrm{CO}}$ as follows:

$$\boldsymbol{\pi}^{\mathrm{CO}} = \begin{cases} \sum_{i,j} \pi'_{ij}\gamma_{ij} + \underset{\boldsymbol{\pi}'' \in \mathcal{G}(\boldsymbol{s}-\boldsymbol{s}', \boldsymbol{d}-\boldsymbol{d}')}{\arg\min} \sum_{i,j} \pi''_{ij}\gamma'_{ij} & (\text{ if there is no negative value in neither } \boldsymbol{s}-\boldsymbol{s}' \text{ nor } \boldsymbol{d}-\boldsymbol{d}'), \\ \bot & (\text{otherwise}). \end{cases}$$

Here, $\boldsymbol{\gamma}'$ is defined as

$$\forall i \in [m], j \in [n], x \in \texttt{POSSIBLE-PI}((i,j)),\ \gamma'_{ij} = \begin{cases} -\infty & (\text{if } ((i,j),x) \in \boldsymbol{S}(t), \\ \gamma_{ij} & (\text{otherwise}), \end{cases}$$

to not let $\boldsymbol{\pi}^{\mathrm{CO}}$ send materials more than the amount determined by $\boldsymbol{S}(t)$.

Algorithm 3 COracle for the Optimal Transport Problem

Input: Cost matrix: $\boldsymbol{\gamma}$, supply vector $\boldsymbol{s} \in \mathbb{R}^m$, demand vector $\boldsymbol{d} \in \mathbb{R}^n$, $\boldsymbol{S}(t)$

$\boldsymbol{\pi}' \leftarrow \boldsymbol{0}^{m \times n}$
for $((i,j),x)$ in $\boldsymbol{S}(t)$ **do**
  $\pi'_{ij} \leftarrow x$
  $s_i \leftarrow s_i - x$
  $d_j \leftarrow d_j - x$
  $\gamma_{ij} \leftarrow -\infty$
  if $s_i < 0$ or $d_j < 0$ **then**
    Return $\bot$
  end if
end for
Compute $\boldsymbol{\pi}'' = \underset{\boldsymbol{\pi} \in \mathcal{G}(\boldsymbol{s}, \boldsymbol{d})}{\arg\min} \sum_{i,j} \pi_{ij}\gamma_{ij}$ using the offline oracle
Return $\boldsymbol{\pi}^{\mathrm{CO}} = \boldsymbol{\pi}' + \boldsymbol{\pi}''$

B.3 COracle for a General Case When $K(= |\mathcal{A}|) = \mathrm{poly}(d)$

Here, we show the time complexity of the COracle when the size of the action class is polynomial in d. If $\mathcal{A}_S$ is an empty set, the COracle returns $\bot$. Otherwise, we return $\boldsymbol{\pi}^{\mathrm{return}} = \underset{\boldsymbol{\pi} \in \mathcal{A}_S}{\arg\max}\, \boldsymbol{\mu}^\top \boldsymbol{\pi}$. The time complexity to construct $\mathcal{A}_S$ is $\mathcal{O}(dK)$. This is because, for each action $\boldsymbol{\pi} \in \mathcal{A}$, we check every dimension to see if $\boldsymbol{\pi} \in \mathcal{A}_S$. Then, the time complexity of finding $\boldsymbol{\pi}^{\mathrm{return}}$ from $\mathcal{A}_S$ is $\mathcal{O}(K \log K)$. Therefore the time complexity of the COracle is $\mathcal{O}\left(dK + K \log K\right)$.

C Proof of Theorem 2

Here, we prove Theorem 2. We first introduce some notions that are useful to prove it. Then, we show some preparatory lemmas that are needed to prove Theorem 2.

C.1 Preparatories

Let us introduce some notions that are useful to prove the theorem.

C.1.1 Arm-Value Pair

We define an *arm-value pair set* $\boldsymbol{M}(\boldsymbol{\pi}) \subset [d] \times \mathbb{R}$ of $\boldsymbol{\pi}$ as follows:

$$\boldsymbol{M}(\boldsymbol{\pi}) = \{(e, \pi_e) \mid \forall e \in [d]\}.$$

Also, we define the *arm-value pair family* $\mathcal{M}$ as follows:

$$\mathcal{M} = \{\boldsymbol{M}(\boldsymbol{\pi}) \mid \forall \boldsymbol{\pi} \in \mathcal{A}\}.$$

For any arm-value pair set $\boldsymbol{M} \in \mathcal{M}$, we denote by M_e the second component of the ordered pair whose first component is e, i.e., $M_e = x$ for any $(e, x) \in \boldsymbol{M}$. Also, we call M_e the e-th element of $\boldsymbol{M}$.

For any two different arm-value pair set $\boldsymbol{M}, \boldsymbol{M}' \in \mathcal{M}$, we define an operator $\boxminus$ such that the e-th element of $\boldsymbol{M}' \boxminus \boldsymbol{M} \subset [d] \times \mathbb{R}$, $(\boldsymbol{M}' \boxminus \boldsymbol{M})_e$, is defined as follows:

$$(\boldsymbol{M}' \boxminus \boldsymbol{M})_e = \begin{cases} M'_e - M_e & (\text{if } M'_e > M_e) \\ 0 & (\text{if } M'_e \leq M_e) \end{cases}.$$

C.1.2 Exchange Class

We define an *exchange set* b as an ordered pair of sets $b = (\boldsymbol{b}^+, \boldsymbol{b}^-)$, where $\boldsymbol{b}^+, \boldsymbol{b}^- \subset [d] \times \mathbb{R} \setminus \{0\}$. We say $\boldsymbol{b}^+$ (or $\boldsymbol{b}^-$) has as *e-changer* if it has an element whose first component is e. For any $\boldsymbol{b}^+$ (or $\boldsymbol{b}^-$), we denote by b^+_e the second component of the ordered pair whose first component is e, i.e., $b^+_e = x$ for any $(e, x) \in \boldsymbol{b}^+$. Also, for any $b = (\boldsymbol{b}^+, \boldsymbol{b}^-)$, we do not let both $\boldsymbol{b}^+$ and $\boldsymbol{b}^-$ have e-changers.

Next, for any arm-value pair set $\boldsymbol{M} \in \mathcal{M}$, exchange set $b = (\boldsymbol{b}^+, \boldsymbol{b}^-)$, and $e \in [d]$, we define operator $\oplus$ such that the e-the element of $\boldsymbol{M} \oplus b \subset [d] \times \mathbb{R}$, $(\boldsymbol{M} \oplus b)_e$, is defined as follows:

$$(\boldsymbol{M} \oplus b)_e = \begin{cases} M_e + b^+_e & (\text{if } \boldsymbol{b}^+ \text{ has an } e\text{−changer}) \\ M_e - b^-_e & (\text{if } \boldsymbol{b}^- \text{ has an } e\text{−changer}) \\ M_e & (\text{if neither } \boldsymbol{b}_+ \text{ nor } \boldsymbol{b}^- \text{ has an } e\text{−changer}) \end{cases}. \tag{37}$$

Similarly, for any arm-value pair set $\boldsymbol{M} \in \mathcal{M}$, exchange set $b = (\boldsymbol{b}^+, \boldsymbol{b}^-)$, and $e \in [d]$, we define operator $\ominus$ such that the e-the element of $\boldsymbol{M} \ominus b \subset [d] \times \mathbb{R}$, $(\boldsymbol{M} \ominus b)_e$, is defined as follows:

$$(\boldsymbol{M} \ominus b)_e = \begin{cases} M_e - b^+_e & (\text{if } \boldsymbol{b}^+ \text{ has an } e\text{−changer}) \\ M_e + b^-_e & (\text{if } \boldsymbol{b}^- \text{ has an } e\text{−changer}) \\ M_e & (\text{if neither } \boldsymbol{b}^+ \text{ nor } \boldsymbol{b}^- \text{ has an } e\text{−changer}) \end{cases}. \tag{38}$$

We call a collection of exchange sets $\mathcal{B}$ an *exchange class* for $\mathcal{M}$ if $\mathcal{B}$ satisfies the following property. For any $\boldsymbol{M}, \boldsymbol{M}' \in \mathcal{M}$ and $e \in [d]$, where $(\boldsymbol{M} \boxminus \boldsymbol{M}')_e > 0$, there exists an exchange set $b = (\boldsymbol{b}^+, \boldsymbol{b}^-) \in \mathcal{B}$ that satisfies the following five constraints: **(a)** $b^-_e = (\boldsymbol{M} \boxminus \boldsymbol{M}')_e$ **(b)** $\boldsymbol{b}^+ \subseteq \boldsymbol{M}' \boxminus \boldsymbol{M}$, **(c)** $\boldsymbol{b}^- \subseteq \boldsymbol{M} \boxminus \boldsymbol{M}'$, **(d)**$(\boldsymbol{M} \oplus b) \in \mathcal{M}$ and **(e)** $(\boldsymbol{M}' \ominus b) \in \mathcal{M}$.

For any $\boldsymbol{a} \subset [d] \times \mathbb{R}$, let a_e denote the second component of the ordered pair whose first component is e, i.e., $a_e = x$ for any $(e, x) \in \boldsymbol{a}$. Then, let $\boldsymbol{\chi}(\boldsymbol{a}) \in \mathbb{R}^d$ denote the vector defined as follows:

$$\chi_e(\boldsymbol{a}) = \begin{cases} a_e & (\text{if } \boldsymbol{a} \text{ has an element whose first component is } e\ ) \\ 0 & (\text{otherwise}) \end{cases}.$$

Also, for any exchange set b, we define $\boldsymbol{\chi}(b) = \boldsymbol{\chi}(\boldsymbol{b}^+) - \boldsymbol{\chi}(\boldsymbol{b}^-)$.

C.2 Preparatory Lemmas

Here, let us introduce some lemmas that are useful to prove the theorem. Below, we define $\boldsymbol{M}^* = \boldsymbol{M}(\boldsymbol{\pi}^*)$.

Lemma 2 (Interpolation Lemma)**.** *Let $\mathcal{B}$ be an exchange class of $\mathcal{M}$, and $\boldsymbol{M}, \boldsymbol{M}'$ be two different members of $\mathcal{M}$. Then, for any $e \in \{s \in [d] \mid M_s \neq M'_s\}$, there exists an exchange set $b = (\boldsymbol{b}^+, \boldsymbol{b}^-) \in \mathcal{B}$, which satisfies the following five constraints: **(a)** $b_e^- = (\boldsymbol{M} \boxminus \boldsymbol{M}')_e$ or $b_e^+ = (\boldsymbol{M}' \boxminus \boldsymbol{M})_e$, **(b)** $\boldsymbol{b}^- \subseteq (\boldsymbol{M} \boxminus \boldsymbol{M}')$, **(c)** $\boldsymbol{b}^+ \subseteq \boldsymbol{M}' \boxminus \boldsymbol{M}$, **(d)** $(\boldsymbol{M} \oplus b) \in \mathcal{M}$ and **(e)**$(\boldsymbol{M}' \ominus b) \in \mathcal{M}$. Moreover, if $\boldsymbol{M}' = \boldsymbol{M}^*$, then we have*

$$\frac{\langle \boldsymbol{\mu}, \boldsymbol{\chi}(b) \rangle}{\chi_e(b)} \geq \Delta_e. \tag{39}$$

Proof. We decompose our proof into two cases.

Case (1): $(\boldsymbol{M} \boxminus \boldsymbol{M}')_e > 0$

By the definition of the exchange class, we know that there exists $b = (\boldsymbol{b}^+, \boldsymbol{b}^-) \in \mathcal{B}$ which satisfies that there is an $e \in [d]$ where **(a)** $b_e^- = (\boldsymbol{M} \boxminus \boldsymbol{M}')_e$, **(b)** $\boldsymbol{b}^- \subseteq (\boldsymbol{M} \boxminus \boldsymbol{M}')$, **(c)** $\boldsymbol{b}^+ \subseteq \boldsymbol{M}' \boxminus \boldsymbol{M}$, **(d)** $(\boldsymbol{M} \oplus b) \in \mathcal{M}$ and **(e)**$(\boldsymbol{M}' \ominus b) \in \mathcal{M}$. Therefore the five constraints are satisfied.

Case (2): $(\boldsymbol{M}' \boxminus \boldsymbol{M})_e > 0$

Using the definition of the exchange class, we see that there exists $b = (\boldsymbol{c}^+, \boldsymbol{c}^-) \in \mathcal{B}$ such that **(a)** $c_e^- = (\boldsymbol{M}' \boxminus \boldsymbol{M})_e$, **(b)** $\boldsymbol{c}^- \subseteq (\boldsymbol{M}' \boxminus \boldsymbol{M})$, **(c)** $\boldsymbol{c}^+ \subseteq (\boldsymbol{M} \boxminus \boldsymbol{M}')$, **(d)** $(\boldsymbol{M} \oplus b) \in \mathcal{M}$, and **(e)** $(\boldsymbol{M}' \ominus b) \in \mathcal{M}$. We construct $b = (\boldsymbol{b}^+, \boldsymbol{b}^-)$ by setting $\boldsymbol{b}^+ = \boldsymbol{c}^-$ and $\boldsymbol{b}^- = \boldsymbol{c}^+$. Notice that, by the construction of b, we have $\boldsymbol{M} \oplus b = \boldsymbol{M} \ominus c$ and $\boldsymbol{M}' \ominus b = \boldsymbol{M}' \oplus c$. Therefore, it is clear that b satisfies the five constraints of the lemma.

Next, let us think when $\boldsymbol{M}' = \boldsymbol{M}^*$ for both cases. Let us consider

$$\boldsymbol{\pi}^1 = \mathop{\arg\min}_{\boldsymbol{\pi} \in \mathcal{M} \setminus \boldsymbol{\pi}^*} \frac{\langle \boldsymbol{\mu}, \boldsymbol{\pi}^* - \boldsymbol{\pi} \rangle}{|\pi_e^* - \pi_e|}. \tag{40}$$

Note that, by the definition of the *G-Gap*, we have $\frac{\langle \boldsymbol{\mu}, \boldsymbol{\pi}^* - \boldsymbol{\pi}^1 \rangle}{|\pi_e^* - \pi_e^1|} = \Delta_e$. We define $\boldsymbol{\pi}^0$ such that $\boldsymbol{M}(\boldsymbol{\pi}^0) = \boldsymbol{M}(\boldsymbol{\pi}^*) \ominus b$. Note that we already have $\boldsymbol{M}(\boldsymbol{\pi}^0) \in \mathcal{M}$. We can see that

$$\frac{\langle \boldsymbol{\mu}, \boldsymbol{\chi}(b) \rangle}{|\chi_e(b)|} = \frac{\langle \boldsymbol{\mu}, \boldsymbol{\pi}^* - \boldsymbol{\pi}^0 \rangle}{|\pi_e^* - \pi_e^0|} \geq \Delta_e, \tag{41}$$

where the inequality follows from the definition of *G-Gap*. □

Next, we establish the confidence bounds used for the analysis of the CSA algorithm.

Lemma 3. *Given a phase $t \in [d]$, we define random events τ_t as follows:*

$$\tau(t) = \left\{ \forall s \in [d] \setminus F(t), |\hat{\mu}_s(t) - \mu_s| < \frac{\Delta_{(d-t+1)}}{(2 + L^2) U_{\mathcal{A}}} \right\}, \tag{42}$$

where $L = \max_{e \in [d], \boldsymbol{\pi}^1, \boldsymbol{\pi}^2, \boldsymbol{\pi}^3 \in \mathcal{A}} \frac{|\pi_e^1 - \pi_e^2|}{|\pi_e^1 - \pi_e^3|}$. Then, we have

$$\Pr\left[\bigcap_{t=1}^{d} \tau(t) \right] \geq 1 - d^2 \exp\left(- \frac{T - d}{2 \left(2 + L^2\right)^2 R^2 \tilde{\log}(d) U_{\mathcal{M}}^2 \mathbf{H}_2} \right) \tag{43}$$

Proof. Fix some $t \in [d]$ and active arm $s \in [d] \setminus F(t)$ of phase t. Note that arm s has been pulled for $\tilde{T}(t)$ times during the first t phases. Therefore, by Hoeffding's inequality, we have

$$\Pr\left[|\hat{\mu}_s(t) - \mu_s| \geq \frac{\Delta_{(d-t+1)}}{(2 + L^2) U_{\mathcal{A}}} \right] \leq 2 \exp\left(- \frac{\tilde{T}(t) \Delta_{(d-t+1)}^2}{2 \left(2 + L^2\right)^2 R^2 U_{\mathcal{A}}^2} \right). \tag{44}$$

By plugging the definition of $\tilde{T}(t)$, the quantity $\tilde{T}(t)\Delta^2_{(d-t+1)}$ on the right hand side of (44) can be further bounded by

$$\begin{aligned}\tilde{T}(t)\Delta^2_{(d-t+1)} &\geq \frac{T-d}{\tilde{\log}(d)(d-t+1)}\Delta^2_{(d-t+1)} \\ &\geq \frac{T-d}{\tilde{\log}(d)\mathbf{H}_2},\end{aligned}$$

where the last inequality follows from the definition of $\mathbf{H}_2 = \max_{s\in[d]} \frac{s}{\Delta^2_{(s)}}$. By plugging the last equality into (44), we have

$$\Pr\left[|\hat{\mu}_s(t) - \mu_s| \geq \frac{\Delta_{(d-t+1)}}{(2+L^2)U_{\mathcal{A}}}\right] \leq 2\exp\left(-\frac{T-d}{2\left(2+L^2\right)^2 R^2\tilde{\log}(d)U^2_{\mathcal{A}}\mathbf{H}_2}\right). \tag{45}$$

Using (45) and a union bound for all $t \in [d]$ and all $s \in [d] \setminus F(t)$, we have

$$\begin{aligned}\Pr\left[\bigcap_{t=1}^{d}\tau(t)\right] &\geq 1 - 2\sum_{t=1}^{d}(d-t+1)\exp\left(-\frac{T-d}{2\left(2+L^2\right)^2 R^2\tilde{\log}(d)U^2_{\mathcal{A}}\mathbf{H}_2}\right) \\ &\geq 1 - d^2\exp\left(-\frac{T-d}{2\left(2+L^2\right)^2 R^2\tilde{\log}(d)U^2_{\mathcal{M}}\mathbf{H}_2}\right)\end{aligned}$$

□

The following lemma builds the confidence bound of inner products.

Lemma 4. *Fix a phase $t \in [d]$. Suppose that random event $\tau(t)$ occurs. For any vector $\boldsymbol{a} \in \mathbb{R}^d$, we have*

$$|\langle\hat{\boldsymbol{\mu}}(t), \boldsymbol{a}\rangle - \langle\boldsymbol{\mu}, \boldsymbol{a}\rangle| < \frac{\Delta_{(d-t+1)}}{(2+L^2)U_{\mathcal{A}}}\|\boldsymbol{a}\|_1. \tag{46}$$

Proof. Suppose that τ_t occurs. We have

$$\begin{aligned}|\langle\hat{\boldsymbol{\mu}}(t), \boldsymbol{a}\rangle - \langle\boldsymbol{\mu}, \boldsymbol{a}\rangle| &= |\langle\hat{\boldsymbol{\mu}}(t) - \boldsymbol{\mu}, \boldsymbol{a}\rangle| \\ &= \left|\sum_{s=1}^{d}(\hat{\mu}_s(t) - \mu_s)a_s\right| \\ &\leq \sum_{s=1}^{d}|\hat{\mu}_s(t) - \mu_s||a_s| \\ &< \frac{\Delta_{(d-t+1)}}{(2+L^2)U_{\mathcal{A}}}\sum_{i=1}^{d}|a_s| \qquad (47) \\ &= \frac{\Delta_{(d-t+1)}}{(2+L^2)U_{\mathcal{A}}}\|\boldsymbol{a}\|_1,\end{aligned}$$

where (47) follows from the definition of τ_t in (42). □

C.3 Main Lemmas

Let $\boldsymbol{R}(t) = \{(e,x) \mid \forall e \in [d], \forall x \in \texttt{POSSIBLE-PI(e)}, (e,x) \notin \boldsymbol{S}(t)\}$. We begin with a technical lemma that characterizes several useful lemma properties of $\boldsymbol{S}(t)$ and $\boldsymbol{R}(t)$.

Lemma 5. *Fix a phase $t \in [d]$. Suppose that $\boldsymbol{S}(t) \subseteq \boldsymbol{M}^*$ and $\boldsymbol{R}(t) \cap \boldsymbol{M}^* = \emptyset$. Let $\boldsymbol{M}$ be a set such that $\boldsymbol{S}(t) \subseteq \boldsymbol{M}$ and $\boldsymbol{R}(t) \cap \boldsymbol{M} = \emptyset$. Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be two sets satisfying that $\boldsymbol{a} \subseteq \boldsymbol{M} \setminus \boldsymbol{M}^*$, $\boldsymbol{b} \subseteq \boldsymbol{M}^* \setminus \boldsymbol{M}$ and $\boldsymbol{a} \cap \boldsymbol{b} = \emptyset$. Then, we have*

$$\boldsymbol{S}(t) \subseteq (\boldsymbol{M} \setminus \boldsymbol{a} \cup \boldsymbol{b}) \quad \textit{and} \quad \boldsymbol{R}(t) \cap (\boldsymbol{M} \setminus \boldsymbol{a} \cup \boldsymbol{b}) = \emptyset$$

Proof. We first prove the first part as follows:

$$\begin{aligned}
\boldsymbol{S}(t) \cap (\boldsymbol{M} \setminus \boldsymbol{a} \cup \boldsymbol{b}) &= (\boldsymbol{S}(t) \cap (\boldsymbol{M} \setminus \boldsymbol{a})) \cup (\boldsymbol{S}(t) \cap \boldsymbol{b}) \\
&= \boldsymbol{S}(t) \cap (\boldsymbol{M} \setminus \boldsymbol{a}) \qquad (48) \\
&= (\boldsymbol{S}(t) \cap \boldsymbol{M}) \setminus \boldsymbol{a} \\
&= \boldsymbol{S}(t) \setminus \boldsymbol{a} \qquad (49) \\
&= \boldsymbol{S}(t), \qquad (50)
\end{aligned}$$

where (48) holds since we have $\boldsymbol{S}(t) \cap \boldsymbol{b} \subseteq \boldsymbol{S}(t) \cap (\boldsymbol{M}^* \setminus \boldsymbol{M}) \subseteq \boldsymbol{M} \cap (\boldsymbol{M}^* \setminus \boldsymbol{M}) = \emptyset$; (49) follows from $\boldsymbol{S}(t) \subseteq \boldsymbol{M}$; and (50) follows from $\boldsymbol{a} \subseteq \boldsymbol{M} \setminus \boldsymbol{M}^*$ and $\boldsymbol{S}(t) \subseteq \boldsymbol{M}^*$ which implies that $\boldsymbol{a} \cap \boldsymbol{S}(t) = \emptyset$. Notice that $\boldsymbol{S}(t) \subseteq (\boldsymbol{M} \setminus \boldsymbol{a} \cup \boldsymbol{b})$.

Then, we proceed to prove the second part in the following

$$\begin{aligned}
\boldsymbol{R}(t) \cap (\boldsymbol{M} \setminus \boldsymbol{a} \cup \boldsymbol{b}) &= (\boldsymbol{R}(t) \cap (\boldsymbol{M} \setminus \boldsymbol{a})) \cup (\boldsymbol{R}(t) \cap \boldsymbol{b}) \\
&= \boldsymbol{R}(t) \cap (\boldsymbol{M} \setminus \boldsymbol{a}) \qquad (51) \\
&= (\boldsymbol{R}(t) \cap \boldsymbol{M}) \setminus \boldsymbol{a} \\
&= \emptyset \setminus \boldsymbol{a} = \emptyset, \qquad (52)
\end{aligned}$$

where (51) follows from the fact that $\boldsymbol{R}(t) \cap \boldsymbol{b} \subseteq \boldsymbol{R}(t) \cap (\boldsymbol{M}^* \setminus M) \subseteq R(t) \cap \boldsymbol{M}^* = \emptyset$; and (52) follows from the fact that $\boldsymbol{R}(t) \cap \boldsymbol{M} = \emptyset$.

□

Let $\hat{\boldsymbol{M}}(t) = \boldsymbol{M}(\hat{\boldsymbol{\pi}}(t))$. The next lemma provides an important insight into the correctness of the CSA algorithm. Informally speaking, suppose that the algorithm does not make an error before phase t. Then, we show that, if arm e has a gap Δ_e larger than the "reference gap" $\Delta_{(d-t+1)}$ of phase t, then arm e must be correctly clarified by $\hat{\boldsymbol{M}}(t)$, i.e., $M_e(t) = M_e^*$.

Lemma 6. *Fix any phase $t > 0$. Suppose that event τ_t occurs. Also, assume that $\boldsymbol{S}(t) \subseteq \boldsymbol{M}^*$ and $\boldsymbol{R}(t) \cap \boldsymbol{S}(t) = \emptyset$. Let $e \in [d] \setminus \boldsymbol{F}(t)$ be an active arm. Suppose that $\Delta_{(d-t+1)} \leq \Delta_e$. Then, we have $(e, \pi_e^*) \in \boldsymbol{M}^* \cap \boldsymbol{S}(t)$.*

Proof. Suppose that $(e, \pi_e^*) \notin (\boldsymbol{M}^* \cap \hat{\boldsymbol{M}}(t))$. This is equivalent to the following

$$(e, \pi_e^*) \in (\boldsymbol{M}^* \cap \neg \hat{\boldsymbol{M}}(t)) \cup (\neg \boldsymbol{M}^* \cap \hat{\boldsymbol{M}}(t)) \qquad (53)$$

(53) can be further rewritten as

$$(e, \pi_e^*) \in (\boldsymbol{M}^* \setminus \hat{\boldsymbol{M}}(t)) \cup (\hat{\boldsymbol{M}}(t) \setminus \boldsymbol{M}^*). \qquad (54)$$

From this assumption, it is easy to see that $\hat{\boldsymbol{M}}(t) \neq \boldsymbol{M}^*$. Therefore, we can apply Lemma 2. We know that there exists $b = (\boldsymbol{b}^+, \boldsymbol{b}^-) \in \mathcal{B}$ such that **(a)** $b_e^- = (\hat{\boldsymbol{M}}(t) \boxminus \boldsymbol{M}^*)_e$ or $b_e^+ = (\boldsymbol{M}^* \boxminus \hat{\boldsymbol{M}}(t))_e$, **(b)** $\boldsymbol{b}^- \subseteq (\hat{\boldsymbol{M}}(t) \boxminus \boldsymbol{M}^*)$, **(c)** $\boldsymbol{b}^+ \subseteq \boldsymbol{M}^* \boxminus \hat{\boldsymbol{M}}(t)$, **(d)** $(\hat{\boldsymbol{M}}(t) \oplus b) \in \mathcal{M}$, **(e)** $(\boldsymbol{M}^* \ominus b) \in \mathcal{M}$, and $\frac{\langle \boldsymbol{\mu}, \boldsymbol{\chi}(b) \rangle}{\chi_e(b)} \geq \Delta_e > 0$.

Using Lemma 5, we see that $(\hat{\boldsymbol{M}}(t) \oplus b) \cap \boldsymbol{R}(t) = \emptyset$, $\boldsymbol{S}(t) \subseteq (\hat{\boldsymbol{M}}(t) \oplus b)$, and $(\boldsymbol{b}^+ \cup \boldsymbol{b}^-) \cap (\boldsymbol{S}(t) \cup \boldsymbol{R}(t)) = \emptyset$.

Recall the definition $\hat{\boldsymbol{M}}(t) \in \underset{\boldsymbol{M} \in \mathcal{M}, \boldsymbol{S}(t) \subseteq \boldsymbol{M}, \boldsymbol{R}(t) \cap \boldsymbol{M} = \emptyset}{\arg\max} \langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\boldsymbol{M}) \rangle$ and also recall that $\hat{\boldsymbol{M}}(t) \oplus b \in \mathcal{M}$. Therefore, we obtain that

$$\frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\hat{\boldsymbol{M}}(t)) \rangle}{\chi_e(b)} \geq \frac{\langle \hat{\boldsymbol{\mu}}, \boldsymbol{\pi}(\hat{\boldsymbol{M}}(t) \oplus b) \rangle}{\chi_e(b)}. \qquad (55)$$

On the other hand, we have

$$\begin{aligned}
\frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\hat{\boldsymbol{M}}(t) \oplus b)\rangle}{\chi_e(b)} &= \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\hat{\boldsymbol{M}}(t)) + \boldsymbol{\chi}(b)\rangle}{\chi_e(b)} && (56)\\
&= \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\hat{\boldsymbol{M}}(t))\rangle}{\chi_e(b)} + \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\chi}(b)\rangle}{\chi_e(b)} \\
&> \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\hat{\boldsymbol{M}}(t))\rangle}{\chi_e(b)} + \frac{\langle \boldsymbol{\mu}, \boldsymbol{\chi}(b)\rangle}{\chi_e(b)} - \frac{\Delta_{(d-t+1)}}{(2+L^2)U_{\mathcal{A}}}\frac{\|\boldsymbol{\chi}(b)\|}{\chi_e(b)} && (57)\\
&\geq \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\hat{\boldsymbol{M}}(t))\rangle}{\chi_e(b)} + \frac{\langle \boldsymbol{\mu}, \boldsymbol{\chi}(b)\rangle}{\chi_e(b)} - \frac{\Delta_e}{(2+L^2)U_{\mathcal{A}}}\frac{\|\boldsymbol{\chi}(b)\|}{\chi_e(b)} \\
&\geq \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\hat{\boldsymbol{M}}(t))\rangle}{\chi_e(b)} + \frac{\langle \boldsymbol{\mu}, \boldsymbol{\chi}(b)\rangle}{\chi_e(b)} - \frac{\Delta_e}{(2+L^2)} && (58)\\
&\geq \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\hat{\boldsymbol{M}}(t))\rangle}{\chi_e(b)} + \frac{1+L^2}{2+L^2}\Delta_e && (59)\\
&\geq \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\hat{\boldsymbol{M}}(t))\rangle}{\chi_e(b)}. && (60)
\end{aligned}$$

This means that $\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\hat{\boldsymbol{M}}(t) \oplus b)\rangle > \langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\hat{\boldsymbol{M}}(t))\rangle$. This contradicts the definition of $\hat{\boldsymbol{\pi}}(t)$, and therefore, we have $(e, \pi_e^*) \in (\boldsymbol{M}^* \cap \hat{\boldsymbol{M}}(t))$.

□

The next lemma takes a step further. Hereinafter, we denote $\tilde{\boldsymbol{M}}^e(t)$ as $\boldsymbol{M}(\tilde{\boldsymbol{\pi}}^e(t))$.

Lemma 7. *Fix any phase $t > 0$. Suppose that event τ_t occurs. Also, assume that $\boldsymbol{S}(t) \subseteq M^*$ and $\boldsymbol{R}(t) \cap \boldsymbol{M}^* = \emptyset$. Let $e \in [d] \setminus \boldsymbol{F}(t)$ satisfy $\Delta_{(d-t+1)} \leq \Delta_e$. Then, we have*

$$\frac{\langle \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\pi}}(t) - \tilde{\boldsymbol{\pi}}^e(t)\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}_e^e(t)|} > \frac{L + 1/L}{2+L^2}\Delta_{(d-t+1)}. \tag{61}$$

Proof. By Lemma 6, we see that

$$(e, \pi_e^*) \in (\boldsymbol{M}^* \cap \boldsymbol{S}(t)). \tag{62}$$

From the definition of $\tilde{\boldsymbol{M}}^e(t)$, which ensures that $\tilde{M}_e^e(t) \neq M_e^*$, we have $(e, \pi_e^*) \in (\boldsymbol{M}^* \setminus \tilde{\boldsymbol{M}}^e(t))$.

Hence, we apply Lemma 2. There exists $b = (\boldsymbol{b}^+, \boldsymbol{b}^-) \in \mathcal{B}$ such that **(a)** $b_e^- = \left(\tilde{\boldsymbol{M}}^e(t) \boxminus \boldsymbol{M}^*\right)_e$ or $b_e^+ = \left(\boldsymbol{M}^* \boxminus \tilde{\boldsymbol{M}}^e(t)\right)_e$, **(b)** $\boldsymbol{b}^- \subseteq (\tilde{\boldsymbol{M}}^e(t) \boxminus \boldsymbol{M}^*)$, **(c)** $\boldsymbol{b}^+ \subseteq \boldsymbol{M}^* \boxminus \tilde{\boldsymbol{M}}^e(t)$, **(d)** $(\tilde{\boldsymbol{M}}^e(t) \oplus b) \in \mathcal{M}$, **(e)** $\boldsymbol{M}^* \ominus b \in \mathcal{M}$, and $\frac{\langle \boldsymbol{\mu}, \boldsymbol{\chi}(b)\rangle}{\chi_e(b)} \geq \Delta_e$.

Define $\overline{\boldsymbol{M}}^e(t) \triangleq \tilde{\boldsymbol{M}}^e(t) \oplus b$. Using Lemma 5, we have $\boldsymbol{S}(t) \subseteq \overline{\boldsymbol{M}}^e(t)$ and $\boldsymbol{R}(t) \cap \overline{\boldsymbol{M}}^e(t) = \emptyset$. Since $\overline{\boldsymbol{M}}^e(t) \in \mathcal{M}$ and by definition $\hat{\boldsymbol{M}}(t) = \underset{\boldsymbol{M} \in \mathcal{M}, \boldsymbol{S}_t \subseteq M, \boldsymbol{R}_t \cap \boldsymbol{M} = \emptyset}{\arg\max} \langle \hat{\boldsymbol{\mu}}, \boldsymbol{\pi}(\boldsymbol{M})\rangle$, we have

$$\frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\hat{\boldsymbol{M}}(t))\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}_e^e(t)|} \geq \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\overline{\boldsymbol{M}}^e(t))\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}_e^e(t)|}. \tag{63}$$

Hence, we have

$$\begin{aligned}
\frac{\langle \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\pi}}(t) - \tilde{\boldsymbol{\pi}}^e(t)\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} &\geq \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\overline{\boldsymbol{M}}^e(t)) - \tilde{\boldsymbol{\pi}}^e(t)\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} \\
&= \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\tilde{\boldsymbol{M}}^e(t) \oplus b)\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} - \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\tilde{\boldsymbol{M}}^e(t))\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} \\
&= \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\tilde{\boldsymbol{M}}^e(t)) + \boldsymbol{\chi}(b)\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} - \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(\tilde{\boldsymbol{M}}^e(t))\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} \qquad (64) \\
&= \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\chi}(b)\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} \\
&> \frac{\chi_e(b)}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|}\left(\frac{\langle \boldsymbol{\mu}, \boldsymbol{\chi}(b)\rangle}{\chi_e(b)} - \frac{\Delta_{(d-t+1)}}{(2+L^2)U_{\mathcal{A}}}\frac{\|\boldsymbol{\chi}(b)\|_1}{\chi_e(b)}\right) \qquad (65) \\
&\geq \frac{\chi_e(b)}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|}\left(\frac{\langle \boldsymbol{\mu}, \boldsymbol{\chi}(b)\rangle}{\chi_e(b)} - \frac{\Delta_e}{(2+L^2)U_{\mathcal{A}}}\frac{\|\boldsymbol{\chi}(b)\|_1}{\chi_e(b)}\right) \qquad (66) \\
&\geq \frac{\chi_e(b)}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} \cdot \frac{1+L^2}{2+L^2}\Delta_e \qquad (67) \\
&\geq \frac{L+1/L}{2+L^2}\Delta_e \\
&\geq \frac{L+1/L}{2+L^2}\Delta_{(d-t+1)}, \qquad (68)
\end{aligned}$$

where (65) follows from Lemma 4, the assumption on event τ_t; (66) follows from the assumption that $\Delta_e \geq \Delta_{(d-t+1)}$; (67) holds since $b \in \mathcal{B}$ and therefore $\frac{\|\boldsymbol{\chi}(b)\|_1}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} \leq U_{\mathcal{A}}$; (68) follows from the fact that $\frac{\langle \boldsymbol{\mu}, \boldsymbol{\chi}(b)\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} \geq \Delta_e$. □

The next lemma shows that, during phase t, if $\Delta_e \leq \Delta_{(d-t+1)}$ for some e, then the empirical gap between $\boldsymbol{\pi}(t)$ and $\tilde{\boldsymbol{\pi}}^e(t)$ is smaller than $\frac{1}{3}\Delta_{(d-t+1)}$.

Lemma 8. *Fix any phase $t > 0$. Suppose that event τ_t occurs. Also, assume that $\boldsymbol{S}(t) \subseteq \boldsymbol{M}^*$ and $\boldsymbol{R}(t) \subseteq \boldsymbol{M}^* = \emptyset$. Suppose an active arm $e \in [d] \setminus \boldsymbol{F}(t)$ satisfies that $M^*_e \neq M_e(t)$. Then, we have*

$$\frac{\langle \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\pi}}(t) - \tilde{\boldsymbol{\pi}}^e(t)\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} \leq \frac{L}{2+L^2}\Delta_{(d-t+1)}. \qquad (69)$$

Proof. Fix any exchange class $\mathcal{B} = \underset{\mathcal{B}' \in \mathrm{Exchange}(\mathcal{M})}{\arg\min} \mathrm{width}(\mathcal{B}')$.

From the assumption that $M^*_e \neq \hat{M}_e(t)$, we can apply Lemma 2, and have **(a)**$b^-_e = (\hat{\boldsymbol{M}}(t) \boxminus \boldsymbol{M}^*)_e$ or $b^+_e = (\boldsymbol{M}^* \boxminus \hat{\boldsymbol{M}}(t))_e$, **(b)** $\boldsymbol{b}^- \subseteq \hat{\boldsymbol{M}}(t) \boxminus \boldsymbol{M}^*$, **(c)** $\boldsymbol{b}^+ \subset \boldsymbol{M}^* \boxminus \hat{\boldsymbol{M}}(t)$, **(d)** $\hat{\boldsymbol{M}}(t) \oplus b \in \mathcal{M}$ **(e)** $\boldsymbol{M}^* \ominus b \in \mathcal{M}$, and $\frac{\langle \boldsymbol{\mu}, \boldsymbol{\chi}(b)\rangle}{\chi_e(b)} \geq \Delta_e > 0$.

Define $\overline{\boldsymbol{M}}^e(t) \triangleq \hat{\boldsymbol{M}}(t) \oplus b$, and let $\overline{\boldsymbol{\pi}}^e(t) = \boldsymbol{\pi}(\overline{\boldsymbol{M}}^e(t))$. We claim that

$$\langle \hat{\boldsymbol{\mu}}(t), \tilde{\boldsymbol{\pi}}^e(t)\rangle \geq \langle \hat{\boldsymbol{\mu}}, \overline{\boldsymbol{\pi}}^e(t)\rangle. \qquad (70)$$

From the definition of $\tilde{\boldsymbol{M}}^e(t)$ in Algorithm 1, we only need to show that (**a**): $\hat{\pi}_e(t) \neq \overline{\pi}_e(t)$ and (**b**): $\boldsymbol{S}(t) \subseteq \overline{\boldsymbol{M}}^e(t)$ and $\boldsymbol{R}(t) \cap \overline{\boldsymbol{M}}^e(t) = \emptyset$. Since, either $\boldsymbol{b}^+$ or $\boldsymbol{b}^-$ has an e-changer, the e-th element of $\overline{\boldsymbol{\pi}}(t)$ is different from that of $\hat{\boldsymbol{\pi}}(t)$. Next, we notice that this follows directly from Lemma 5 by setting $M = \hat{\boldsymbol{M}}(t)$. Hence, we have shown that (70) holds.

Therefore, we have

$$\begin{aligned}
\frac{\langle \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\pi}}(t) - \tilde{\boldsymbol{\pi}}^e(t)\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} &\leq \frac{\langle \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\pi}}(t) - \overline{\boldsymbol{\pi}}^e(t)\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} \\
&\leq \frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(t) - (\boldsymbol{\pi}(t) + \boldsymbol{\chi}(b))\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} \\
&= -\frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\chi}(b)\rangle}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} \\
&\leq \frac{\chi_e(b)}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} \cdot \left(-\frac{\langle \boldsymbol{\mu}, \boldsymbol{\chi}(b)\rangle}{\chi_e(b)} + \frac{\Delta_{(d-t+1)}}{(2+L^2)\mathrm{U}_{\mathcal{M}}} \frac{\|\boldsymbol{\chi}(b)\|_1}{\chi_e(b)} \right) \qquad (71) \\
&\leq \frac{\chi_e(b)}{|\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)|} \cdot \frac{\Delta_{(d-t+1)}}{(2+L^2)} \qquad (72) \\
&\leq \frac{L}{2+L^2} \Delta_{(d-t+1)}
\end{aligned}$$

□

C.4 Proof of Theorem 2

For the reader's convenience, we first restate Theorem 2 as follows.

Theorem 2. *Given any $T > d$, action class $\mathcal{A} \subset \mathbb{R}^d$, and $\boldsymbol{\mu} \in \mathbb{R}^d$, the CSA algorithm uses at most T samples and outputs a solution $\boldsymbol{\pi}^{\mathrm{out}} \in \mathcal{A} \cup \{\perp\}$ such that*

$$\begin{aligned}
&\Pr\left[\boldsymbol{\pi}^{\mathrm{out}} \neq \boldsymbol{\pi}^*\right] \\
\leq\ & d^2 \exp\left(-\frac{T-d}{2(2+L^2)^2 R^2 \tilde{\log}(d) U_{\mathcal{A}}^2 \mathbf{H}_2} \right), \qquad (8)
\end{aligned}$$

where $L = \max_{e\in[d],\boldsymbol{\pi}^1,\boldsymbol{\pi}^2,\boldsymbol{\pi}^3\in\mathcal{A},\pi^1_e\neq\pi^3_e} \frac{|\pi^1_e - \pi^2_e|}{|\pi^1_e - \pi^3_e|}$, $\tilde{\log}(d) \triangleq \sum_{s=1}^d \frac{1}{s}$, *and* $U_{\mathcal{A}} = \max_{\boldsymbol{\pi},\boldsymbol{\pi}'\in\mathcal{A},e\in\{s\in[d]\ |\ \pi_s\neq\pi'_s\}} \frac{\sum_{s=1}^d |\pi_s - \pi'_s|}{|\pi_e - \pi'_e|}$.

Proof. First, we show that the algorithm takes at most T samples. Note that exactly one arm is pulled for $\tilde{T}_1$ times, one arm is pulled $\tilde{T}_2$ times, ..., and one arm is pulled $\tilde{T}_d$ times. Therefore, the total number of samples used by the algorithm is bounded by

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

By Lemma 3, we know that the event $\tau = \bigcap_{t=1}^T \tau_t$ occurs with probability at least $1 - d^2 \exp\left(\frac{T-d}{R^2 \tilde{\log}(d) U_{\mathcal{A}}^2 \mathbf{H}_2}\right)$. Therefore, we only need to prove that, under event τ the algorithm outputs $\boldsymbol{M}^*$. We will assume that event τ occurs in the rest of the proof.

We prove the theorem by induction. Fix a phase $t \in [d]$. Suppose that the algorithm does not make any error before phase t, i.e., $\boldsymbol{S}(t) \subseteq \boldsymbol{M}^*$ and $\boldsymbol{R}(t) \cap \boldsymbol{M}^* \neq \emptyset$. We show that the algorithm does not err at phase t.

At the beginning of phase t, there are exactly $t-1$ inactive arms, i.e., $|F(t)| = t-1$. Therefore, there must exist an active arm $e(t) \in [d] \setminus \boldsymbol{F}(t)$, such that $\Delta_{e(t)} = \Delta_{(d-t+1)}$. Hence, by Lemma 7, we have

$$\frac{\langle \hat{\boldsymbol{\mu}}(t), \boldsymbol{\pi}(t) - \tilde{\boldsymbol{\pi}}^{e(t)}(t)\rangle}{\left|\hat{\pi}_{e(t)}(t) - \tilde{\pi}^{e(t)}_{e(t)}(t)\right|} \geq \frac{L + 1/L}{2+L^2} \Delta_{(d-t+1)}, \qquad (73)$$

where b is an exchange set of $\hat{\boldsymbol{M}}(t)$ and $\tilde{\boldsymbol{M}}^e(t)$.

Notice that the algorithm makes an error in phase t if and only if $(p(t), \hat{\pi}_{p(t)}(t)) \in (\boldsymbol{M}^* \cap \neg\hat{\boldsymbol{M}}(t)) \cup (\neg\boldsymbol{M}^* \cap \hat{\boldsymbol{M}}(t))$.

Suppose that $(p(t), \hat{\pi}_{p(t)}(t)) \in (\boldsymbol{M}^* \cap \neg\hat{\boldsymbol{M}}(t)) \cup (\neg\boldsymbol{M}^* \cap \hat{\boldsymbol{M}}(t))$. From Lemma 8, we have

$$\frac{\langle \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\pi}}(t) - \tilde{\boldsymbol{\pi}}^{p(t)}(t) \rangle}{\left| \hat{\pi}_{p(t)} - \tilde{\pi}^{p(t)}_{p(t)} \right|} \leq \frac{L}{2 + L^2} \Delta_{(d-t+1)}. \tag{74}$$

By combining (73) and (74), we see that

$$\frac{\langle \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\pi}}(t) - \tilde{\boldsymbol{\pi}}^{p(t)}(t) \rangle}{\left| \hat{\pi}_{p(t)}(t) - \tilde{\pi}^{p(t)}_{p(t)}(t) \right|} \leq \frac{L}{2 + L^2} \Delta_{(d-t+1)} < \frac{L + 1/L}{2 + L^2} \Delta_{(d-t+1)} \leq \frac{\langle \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\pi}}(t) - \tilde{\boldsymbol{\pi}}^{e(t)}(t) \rangle}{\hat{\pi}_{e(t)}(t) - \tilde{\pi}^{e(t)}_{e(t)}(t)}. \tag{75}$$

However, (75) is contradictory to the definition of

$$p(t) = \operatorname*{arg\,max}_{e \in [d] \setminus F(t)} \frac{\langle \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\pi}}(t) - \tilde{\boldsymbol{\pi}}(t) \rangle}{\hat{\pi}_e(t) - \tilde{\pi}^e_e(t)}. \tag{76}$$

Therefore, we have proven that $(p(t), \hat{\pi}_{p(t)}(t)) \notin (\boldsymbol{M}^* \cap \neg\hat{\boldsymbol{M}}(t)) \cup (\neg\boldsymbol{M}^* \cap \hat{\boldsymbol{M}}(t))$. This means that the algorithm does not err at phase t, or equivalently $\boldsymbol{S}(t+1) \subseteq \boldsymbol{M}^*$ and $\boldsymbol{R}(t+1) \cap \boldsymbol{M}^* = \emptyset$. By induction, we have proven that the algorithm does not err at any phase $t \in [d]$.

Hence, we have $\boldsymbol{S}(d+1) \subseteq \boldsymbol{M}^*$ and $\boldsymbol{R}(d+1) \subseteq \neg\boldsymbol{M}^*$ in the final phase. This means that $\boldsymbol{S}(d+1) = \boldsymbol{M}^*$, and therefore, $\boldsymbol{\pi}^{\text{out}} = \boldsymbol{\pi}^*$ after phase d. □

D Proof of Theorem 3

We first introduce some useful lemmas to prove Theorem 3. Lemma 9 shows that Algorithm 2 pulls arms no more than T times. Recall that $B = 2^{\lceil \log_2 d \rceil} - 1$ and $T' = T - \left\lfloor \frac{T}{d}\beta \right\rfloor \times d$.

Lemma 9. *Algorithm 2 terminates in phase $\lceil \log_2 d \rceil$ with no more than a total of T arm pulls.*

Proof. The total number of arm pulls T_{total} is bounded as follows.

$$\begin{aligned}
T_{\text{total}} &= \left\lfloor \frac{T}{d}\beta \right\rfloor \times d + \sum_{r=1}^{\lceil \log_2 d \rceil} \sum_{s \in [d]} \lceil p_s(r) \cdot m(r) \rceil \\
&\leq \left\lfloor \frac{T}{d}\beta \right\rfloor \times d + \sum_{r=1}^{\lceil \log_2 d \rceil} \left(d + \frac{T' - d\lceil \log_2 d \rceil}{B/2^{r-1}} \right) \\
&\leq \left\lfloor \frac{T}{d}\beta \right\rfloor \times d + d\lceil \log_2 d \rceil + \frac{2^{\lceil \log_2 d \rceil} - 1}{B} (T' - d\lceil \log_2 d \rceil) \\
&= T.
\end{aligned}$$

□

Let us write $\Delta'_{(i)} = \langle \boldsymbol{\mu}, \boldsymbol{\pi}^1 - \boldsymbol{\pi}^i \rangle$. Note that $\boldsymbol{\mu}^\top \boldsymbol{\pi}^i \geq \boldsymbol{\mu}^\top \boldsymbol{\pi}^{i+1}$ for all $i \in [K-1]$. Lemma 10 bounds the probability that a certain action has its estimate of the expected reward larger than that of the best action at the end of phase r.

Lemma 10. *For a fixed realization of $\mathbb{A}(r)$ satisfying $\boldsymbol{\pi}^* \in \mathbb{A}(r)$, for any action $\boldsymbol{\pi}^i \in \mathbb{A}(r)$,*

$$\Pr\left[\hat{\boldsymbol{\mu}}^\top (r+1) \boldsymbol{\pi}^1 < \hat{\boldsymbol{\mu}}^\top (r+1) \boldsymbol{\pi}^i \right] \leq \exp\left(- \frac{2 {\Delta'_{(i)}}^2}{\sum_{s=1}^{d} \frac{(\pi^1_s - \pi^i_s)^2}{T_s(r+1)} R^2} \right). \tag{77}$$

Proof. For any $i \in \{2, \dots, K\}$, we have

$$\begin{aligned}\Pr\left[\hat{\boldsymbol{\mu}}^\top(r+1)\boldsymbol{\pi}^1 < \hat{\boldsymbol{\mu}}^\top(r+1)\boldsymbol{\pi}^i\right] &\leq \Pr\left[\left\langle\hat{\boldsymbol{\mu}}(r+1) - \boldsymbol{\mu}, \boldsymbol{\pi}^1 - \boldsymbol{\pi}^i\right\rangle < -\Delta'_{(i)}\right] \\ &\leq \exp\left(-\frac{2{\Delta'_{(i)}}^2}{\sum_{s=1}^{d}\frac{(\pi^1_s-\pi^i_s)^2}{T_s(r+1)}R^2}\right), \end{aligned} \tag{78}$$

where the last inequality follows from Hoeffding's inequality (Hoeffding, 1963).

If we use allocation vector (15), (78) can be upper bounded by

$$\begin{aligned}(78) &\leq \exp\left(-\frac{2{\Delta'_{(i)}}^2}{\sum_{e=1}^{d}\frac{(\pi^1_e-\pi^i_e)^2}{\lceil p_s(r)m(r)\rceil}R^2}\right) \\ &\leq \exp\left(-\frac{2{\Delta'_{(i)}}^2}{R^2V'^2}\cdot\frac{T'-\lceil\log_2 d\rceil}{B/2^{r-1}}\right) \\ &\leq \exp\left(-\frac{{\Delta'_{(i)}}^2}{R^2V'^2}\cdot\frac{T'-\lceil\log_2 d\rceil}{d/2^{r-1}}\right)\end{aligned} \tag{79}$$

where $V' = \sum_{e=1}^{d}\left|\pi^1_e - \pi^i_e\right|$. □

Next, we bound the error probability of a single phase r in Lemma 11.

Lemma 11. *Assume that the best action is not eliminated prior to phase r, i.e., $\boldsymbol{\pi}^1 \notin \mathbb{A}(r)$. Then, the probability that the best action is eliminated in phase $r+1$ is bounded as*

$$\Pr\left[\boldsymbol{\pi}^1 \notin \mathcal{A}_r | \boldsymbol{\pi}^1 \in \mathcal{A}_{r-1}\right] \leq \begin{cases} \frac{4K}{d}\exp\left(\frac{T'-\lceil\log_2 d\rceil}{R^2V^2}\cdot\frac{{\Delta_{(i_r)}}^2}{i_r}\right) & (\textit{when } r=1) \\ 3\exp\left(\frac{T'-\lceil\log_2 d\rceil}{R^2V^2}\cdot\frac{{\Delta_{(i_r)}}^2}{i_r}\right) & (\textit{when } r>1), \end{cases} \tag{80}$$

where $i_r = \left\lceil\frac{d}{2^{r+1}}\right\rceil + 1$ and

$$V = \max_{\boldsymbol{\pi}\in\mathcal{A}\setminus\{\boldsymbol{\pi}^*\}, s\in\{e\in[d] \mid \pi^*_e\neq\pi_e\}}\frac{\sum_{u=1}^{d}|\pi^*_u - \pi_u|}{|\pi^*_s - \pi_s|}. \tag{81}$$

Proof. Define $\mathbb{B}(r+1)$ as the set of actions in $\mathbb{A}(r)$ excluding the best action and $\lceil\frac{d}{2^{r+1}}\rceil - 1$ suboptimal actions with the largest expected rewards. We have $|\mathbb{B}(r+1)| = |\mathbb{A}(r)| - \lceil\frac{d}{2^{r+1}}\rceil$.

Let us think of bijective function $p : [d] \to [d]$, which satisfies the following:

$$\forall s \in [d],\ \Delta_{(p(s))} = \Delta_s. \tag{82}$$

Also, we denote the inverse mapping of f by f^{-1}. Next, for any $i \in [d]$, we define $s(i)$ as follows:

$$s(i) = \arg\max_{s\in[d]}|\pi^1_s - \pi^i_s|.$$

Then, from the definition of $\{\Delta_s\}_{s=1,\dots,d}$, we have

$$\frac{\Delta'_{(i)}}{|\pi^1_{s(i)} - \pi^i_{s(i)}|} \geq \Delta_{s(i)} = \Delta_{(p(s(i)))}. \tag{83}$$

Also, we have

$$\min_{i\in\mathbb{B}(r+1)}\Delta_{(p(s(i)))} \geq \Delta_{(\lceil\frac{d}{2^{r+1}}\rceil+1)}. \tag{84}$$

If the best action is eliminated in phase r, then at least $\lceil \frac{d}{2^r} \rceil - \lceil \frac{d}{2^{r+1}} \rceil + 1$ actions of $\mathbb{B}(r+1)$ have their estimates of the expected rewards larger than that of the best action.

Let $N(r)$ denote the number of actions in $\mathbb{B}(r+1)$ whose estimates of the expected rewards are larger than that of the best action. By Lemma 10, we have

$$\begin{aligned}
\mathbb{E}\left[N_r\right] &= \sum_{i \in \mathcal{B}_r} \Pr\left[\hat{\boldsymbol{\mu}}^\top (r+1)\boldsymbol{\pi}^1 < \hat{\boldsymbol{\mu}}^\top (r+1)\boldsymbol{\pi}^i\right] \\
&\leq \sum_{i \in \mathcal{B}_r} \exp\left(-\frac{{\Delta'_{(i)}}^2}{R^2 {V'}^2} \cdot \frac{T' - \lceil \log_2 d \rceil}{d/2^{r-1}}\right) \\
&\leq |\mathcal{B}_r| \max_{i \in \mathbb{B}(r+1)} \exp\left(-\frac{{\Delta'_{(i)}}^2}{R^2 {V'}^2} \cdot \frac{T' - \lceil \log_2 d \rceil}{d/2^{r-1}}\right) \\
&= |\mathcal{B}_r| \max_{i \in \mathbb{B}(r+1)} \exp\left(-\frac{|\pi^1_{s(i)} - \pi^i_{s(i)}|^2 \Delta^2_{(p(s(i)))}}{R^2 {V'}^2} \cdot \frac{T' - \lceil \log_2 d \rceil}{d/2^{r-1}}\right) \qquad (85) \\
&\leq \left(|\mathcal{A}_{r-1}| - \left\lceil \frac{d}{2^{r+1}} \right\rceil\right) \exp\left(\frac{T' - \lceil \log_2 d \rceil}{R^2 V^2} \cdot \frac{{\Delta_{(\lceil \frac{d}{2^{r+1}} \rceil + 1)}}^2}{\left\lceil \frac{d}{2^{r+1)}} \right\rceil + 1}\right) \qquad (86)
\end{aligned}$$

where (85) follows from (83), (86) follows from (84), and

$$V = \max_{\boldsymbol{\pi} \in \mathcal{A} \setminus \{\boldsymbol{\pi}^*\}, s \in \{e \in [d] \mid \pi^*_e \neq \pi_e\}} \frac{\sum_{u=1}^d |\pi^*_u - \pi_u|}{|\pi^*_s - \pi_s|}. \qquad (87)$$

Then, together with Markov's inequality, we obtain

$$\begin{aligned}
\Pr\left[\boldsymbol{\pi}^1 \notin \mathcal{A}_r\right] &\leq \Pr\left[N_r \geq \left\lceil \frac{d}{2^r} \right\rceil - \left\lceil \frac{d}{2^{r+1}} \right\rceil + 1\right] \\
&\leq \frac{\mathbb{E}\left[N_r\right]}{\lceil \frac{d}{2^r} \rceil - \lceil \frac{d}{2^{r+1}} \rceil + 1} \\
&\leq \frac{|\mathcal{A}_{r-1}| - \lceil \frac{d}{2^{r+1}} \rceil}{\lceil \frac{d}{2^r} \rceil - \lceil \frac{d}{2^{r+1}} \rceil + 1} \exp\left(\frac{T' - \lceil \log_2 d \rceil}{R^2 V^2} \cdot \frac{\Delta_{(\lceil \frac{d}{2^{r+1}} \rceil + 1)}}{\left\lceil \frac{d}{2^{r+1)}} \right\rceil + 1}\right) \qquad (88)
\end{aligned}$$

When $r = 1$, we have $|\mathcal{A}_{r-1}| = K$. Thus,

$$\begin{aligned}
\frac{|\mathcal{A}_{r-1}| - \lceil \frac{d}{2^{r+1}} \rceil}{\lceil \frac{d}{2^r} \rceil - \lceil \frac{d}{2^{r+1}} \rceil + 1} &= \frac{K - \lceil \frac{d}{2^{r+1}} \rceil}{\lceil \frac{d}{2^r} \rceil - \lceil \frac{d}{2^{r+1}} \rceil + 1} \\
&\leq \frac{K}{\frac{d}{2} - \frac{d}{2^2}} \\
&\leq \frac{4K}{d}. \qquad (89)
\end{aligned}$$

When $r > 1$, we have $|\mathcal{A}_{r-1}| = \lceil \frac{d}{2^{r-1}} \rceil$. Thus,

$$\begin{aligned}
\frac{|\mathcal{A}_{r-1}| - \lceil \frac{d}{2^{r+1}} \rceil}{\lceil \frac{d}{2^r} \rceil - \lceil \frac{d}{2^{r+1}} \rceil + 1} &= \frac{\lceil \frac{d}{2^{r-1}} \rceil - \lceil \frac{d}{2^{r+1}} \rceil}{\lceil \frac{d}{2^r} \rceil - \lceil \frac{d}{2^{r+1}} \rceil + 1} \\
&\leq \frac{\frac{d}{2^{r-1}} + 1 - \lceil \frac{d}{2^{r+1}} \rceil}{\frac{d}{2^r} - \lceil \frac{d}{2^{r+1}} \rceil + 1} \\
&\leq \frac{3 \cdot \frac{d}{2^{r+1}} + \frac{d}{2^{r+1}} - \lceil \frac{d}{2^{r+1}} \rceil + 1}{\frac{d}{2^{r+1}} + \frac{d}{2^{r+1}} - \lceil \frac{d}{2^{r+1}} \rceil + 1} \\
&\leq 3, \qquad (90)
\end{aligned}$$

where the last inequality results from the fact that for any $x, y > 0$, $\frac{3x+y}{x+y} \leq 3$.
Therefore, for this specific realization of $\mathcal{A}_{r-1}$ satisfying $1 \in \mathcal{A}_{r-1}$,

$$\Pr\left[\boldsymbol{\pi}^1 \notin \mathcal{A}_r | \boldsymbol{\pi}^1 \in \mathcal{A}_{r-1}\right] \leq \begin{cases} \frac{4K}{d} \exp\left(\frac{T' - \lceil \log_2 d \rceil}{R^2 V^2} \cdot \frac{{\Delta_{(i_r)}}^2}{i_r}\right) & (\text{when } r = 1) \\ 3 \exp\left(\frac{T' - \lceil \log_2 d \rceil}{R^2 V^2} \cdot \frac{{\Delta_{(i_r)}}^2}{i_r}\right) & (\text{when } r > 1), \end{cases} \tag{91}$$

where $i_r = \lceil \frac{d}{2^{r+1}} + 1 \rceil$. □

Finally, we prove Theorem 3.

Proof of Theorem 3. We have

$$\begin{aligned} \Pr\left[\boldsymbol{\pi}^{\text{out}} \neq \boldsymbol{\pi}^1\right] &= \Pr\left[\boldsymbol{\pi}^1 \notin \mathcal{A}_{\lceil \log_2 d \rceil}\right] \\ &\leq \sum_{r=1}^{\lceil \log_2 d \rceil} \Pr\left[\boldsymbol{\pi}^1 \notin \mathcal{A}_r | \boldsymbol{\pi}^1 \in \mathcal{A}_{r-1}\right] \\ &\leq \frac{4K}{d} \exp\left(\frac{T' - \lceil \log_2 d \rceil}{R^2 V^2} \cdot \frac{\Delta_{(i_1)}^2}{i_1}\right) + \sum_{s=1}^{d} 3 \exp\left(\frac{T' - \lceil \log_2 d \rceil}{R^2 V^2} \cdot \frac{\Delta_{(i_r)}^2}{i_r}\right) \\ &\leq \left(\frac{4K}{d} + 3\left(\lceil \log_2 d \rceil - 1\right)\right) \exp\left(\frac{T' - \lceil \log_2 d \rceil}{R^2 V^2} \cdot \frac{1}{\max_{1 \leq s \leq d} \frac{s}{{\Delta_{(s)}}^2}}\right) \\ &< \left(\frac{4K}{d} + 3 \log_2 d\right) \exp\left(\frac{T' - \lceil \log_2 d \rceil}{R^2 V^2} \cdot \frac{1}{\mathbf{H}_2}\right), \end{aligned} \tag{92}$$

where

$$\mathbf{H}_2 = \max_{1 \leq s \leq d} \frac{s}{\Delta_{(s)}^2}.$$

□