A Fast Algorithm for PAC Combinatorial Pure Exploration

Noa Ben-David and Sivan Sabato

Department of Computer Science Ben-Gurion University of the Negev Beer Sheva Israel
bendanoa@post.bgu.ac.il, sabatos@cs.bgu.ac.il

Abstract

We consider the problem of Combinatorial Pure Exploration (CPE), which deals with finding a combinatorial set of arms with a high reward, when the rewards of individual arms are unknown in advance and must be estimated using arm pulls. Previous algorithms for this problem, while obtaining sample complexity reductions in many cases, are highly computationally intensive, thus making them impractical even for mildly large problems. In this work, we propose a new CPE algorithm in the PAC setting, which is computationally light weight, and so can easily be applied to problems with tens of thousands of arms. This is achieved since the proposed algorithm requires a very small number of combinatorial oracle calls. The algorithm is based on successive acceptance of arms, along with elimination which is based on the combinatorial structure of the problem. We provide sample complexity guarantees for our algorithm, and demonstrate in experiments its usefulness on large problems, whereas previous algorithms are impractical to run on problems of even a few dozen arms. The code is provided at https://github.com/noabdavid/csale. The full version of this paper is available at https://arxiv.org/abs/2112.04197.

1 Introduction

Combinatorial pure exploration (CPE) is an important statistical framework, used in diverse applications such as channel selection in a wireless network (Xue et al. 2018), job applicant screening (Schumann et al. 2019), and robot decision making (Marcotte et al. 2020). In the CPE framework, there are n arms that can be pulled, where each arm is associated with an unknown distribution of real-valued rewards. Whenever an arm is pulled, an instantaneous reward is independently drawn from that arm's reward distribution. An *arm set* is a set of arms, and its reward is the sum of rewards of the arms it contains. The goal is to find a valid arm set with a high reward. The set of valid arm sets is called the *decision class*, and it is typically the set of legal solutions of a combinatorial problem. The algorithm uses individual arm pulls (samples) to collect information on the rewards, and attempts to select from the decision class an arm set with a high expected reward, using a low sample complexity.

Many applications can be mapped to the CPE setting. As an example, consider the problem of finding the shortest routing path in a network (Talebi et al. 2017). The network is modeled as a directed graph, where each edge represents a link between two routing servers, and the cost of routing through the link is the (random) delay in the link. The task is to find the path from a source to a target with the smallest expected delay. In this problem, the CPE arms are mapped to the links (graph edges), and pulling an arm is done by sending a packet through the corresponding link and measuring its delay. The decision class is thus the set of possible paths between the source and the target. As another example, consider pairing online players for matches, based on their skill compatibility. This can be represented by a maximum weight matching problem on a full graph, where the reward measures the compatibility of the players, and it can be sampled by running a simulation of a match.

As evident in the examples above, in many natural cases the size of the decision class of a CPE problem is exponential in the number of arms. Algorithms for the general CPE problem with such decision classes thus usually assume access to an *oracle* which can efficiently solve the combinatorial optimization problem (see, e.g., Chen et al. 2014; Gabillon et al. 2016; Chen et al. 2017; Cao and Krishnamurthy 2019). Using such an oracle, these algorithms are efficient, in that their computational complexity is polynomial in the problem parameters. However, while theoretically efficient, running the oracle is usually highly demanding in terms of computation resources for large problems. In existing CPE algorithms, the number of oracle calls is either similar to the sample complexity or a high degree polynomial. As a result, they are computationally heavy, and impractical to run even on moderately large problems.

In this work, we propose a new CPE algorithm that is significantly less computationally demanding than previous algorithms, and so it is practical to run on large problems. At the same time, it has sample complexity guarantees that show it can be useful for many problems. The algorithm, called `CSALE` (Combinatorial Successive Acceptance with Light Elimination), uses a significantly smaller number of oracle calls than previous algorithms, and does not use any other computationally demanding operations. `CSALE` works in the (ϵ, δ)-PAC setting, in which the algorithm is provided with an error parameter ϵ and a confidence parameter δ, and finds, with a probability of $1 - \delta$, an arm set from the decision class with an ϵ-optimal reward. Compared with re-

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

quiring that the algorithm find an optimal solution, the PAC approach allows convergence even if the optimal solution is not unique, and is also more suitable for cases where the gap between the (unique) optimal solution and the second-best solution is very small.

The number of oracle calls used by CSALE is only $O(d\log(d))$, where d is the maximal cardinality of an arm set. This is achieved by avoiding the common approach of searching for arms to eliminate by estimating each arm's contribution to the solution value (its *gap*). CSALE successively *accepts* arms if they can be safely added to the output solution. It eliminates arms only due to the combinatorial structure of the problem. For instance, in the shortest path problem, arms that create a cycle with accepted arms can be safely eliminated. We provide sample complexity guarantees for CSALE, which show when it can provide useful sample complexity reductions.

We demonstrate the practical computational advantage of CSALE by running it on instances of the s-t shortest path problem and the maximum weight matching problem, which we generated based on real-world graph data sets. As a baseline, we compare to the CLUCB-PAC algorithm (Chen et al. 2014), which is the only existing PAC algorithm for the general CPE problem. We show that in small graph problems, with about a dozen nodes and up to a few dozen arms (edges), CLUCB-PAC usually performs somewhat better than CSALE in terms of number of samples, but CSALE runs 5 orders of magnitude faster. We note that existing best-arm-set CPE algorithms, even if they could be adapted to the PAC setting, are even more computationally demanding than CLUCB-PAC. Due to its computational requirements, we could not test CLUCB-PAC on graphs with more than a few dozen nodes and edges.

We run CSALE on problems as large as thousands of nodes and tens of thousands of edges (arms) and compare it to a naive baseline, which pulls each arm the same number of times. We show that CSALE provides sample complexity improvements compared to this baseline, while still being practical to run in terms of computational resources, even on these large problems.

2 Related Work

CPE was first introduced by Chen et al. (2014). They proposed CLUCB-PAC, a PAC algorithm which calls an optimization oracle twice for every arm pull. They also propose an algorithm in the fixed-budget setting (which does not give PAC guarantees), that relies on a *constrained oracle*: this is an oracle that outputs the optimal arm set subject to problem-specific constraints. They observe that such an oracle exists whenever an unconstrained oracle exists, since the former can be implemented by calling the latter with a simple transformation of the inputs. Other PAC algorithms have been proposed for special cases of CPE, such as top-k arm selection (Kalyanakrishnan et al. 2012; Zhou, Chen, and Li 2014; Chaudhuri and Kalyanakrishnan 2019), matroids (Chen, Gupta, and Li 2016), and dueling bandits (Chen et al. 2020).

CPE algorithms for the fixed-confidence setting were proposed in Chen et al. (2014); Gabillon et al. (2016); Chen et al. (2017); Cao and Krishnamurthy (2019). In these algorithms, the number of oracle calls is either similar to the sample complexity or is a high-degree polynomial. We note that none of the algorithms mentioned above have been empirically evaluated in the works above, which focus on theoretical guarantees. As far as we know, they also do not have available implementations.

The idea of successive acceptance with limited elimination was previously proposed in a different context, of active structure learning of Bayesian networks (Ben-David and Sabato 2021). That problem also involves sampling for the purpose of combinatorial selection, but it is not a CPE problem. Moreover, the algorithm proposed in that work is specifically tailored to the challenges of Bayesian network learning, and relies on computationally demanding operations. In contrast, the current work proposes a computationally light algorithm for the CPE setting.

3 Setting and Notation

For an integer i, denote $[i] := \{1, \ldots, i\}$. Let n be the number of arms, and denote the set of arms by $[n]$. Each arm is associated with a reward distribution. For simplicity, we assume that the support of all reward distributions is in $[0, 1]$. The results can easily be generalized to sub-Gaussian distributions. Denote the expected reward of an arm $a \in [n]$ by μ_a. Let $\boldsymbol{\mu}$ be the vector of expected rewards of the arms in $[n]$. The reward of an arm set $M \subseteq [n]$ is the sum of the expected rewards of its arms, denoted $\mu(M) := \sum_{a\in M} \mu_a$. Denote the decision class of potential arm sets by $\mathcal{M} \subseteq 2^{[n]}$. Denote the optimal score of an arm set in $\mathcal{M}$ by $\mu^* := \max_{M\in\mathcal{M}} \mu(M)$. Given $\epsilon > 0, \delta \in (0, 1)$, our goal is to find an arm set $\hat{M} \in \mathcal{M}$ such that with a probability of at least $1 - \delta$, $\mu(M) \geq \mu^* - \epsilon$. Denote the maximal cardinality among the arm sets in $\mathcal{M}$ by $d(\mathcal{M}) := \max_{M\in\mathcal{M}} |M|$. Whenever it is clear from context, we use $d := d(\mathcal{M})$.

We assume access to a constrained oracle for the given combinatorial problem. Given a decision class, a constrained oracle $\mathcal{M}$ accepts a reward vector $\boldsymbol{\mu}$, a set of required arms $A \subseteq [n]$, and a set of forbidden arms $B \subseteq [n]$, and returns an optimal solution in the decision class that satisfies these constraints. Formally, the constrained oracle is a function $\mathsf{COracle} : \mathbb{R}^n \times 2^{[n]} \times 2^{[n]} \to \mathcal{M}$ such that $\mathsf{COracle}(\boldsymbol{\mu}, A, B) \in \operatorname{argmax}\{\mu(M) \mid M \in \mathcal{M}, A \subseteq M, B \cap M = \emptyset\}$.

Denote the set of all optimal arm sets in $\mathcal{M}$ by $\mathcal{M}^* := \{M \in \mathcal{M} \mid \mu(M) = \mu^*\}$. The *gap* of an arm $a \in [n]$ is defined as:

$$\Delta(a) := \begin{cases} \mu^* - \max_{M:a\notin M} \mu(M), & a \in \bigcup_{M^*\in\mathcal{M}^*} M^* \\ \mu^* - \max_{M:a\in M} \mu(M), & a \notin \bigcup_{M^*\in\mathcal{M}^*} M^* \end{cases}.$$

Note that if a is included in some, but not all, of the optimal arm sets, then $\Delta(a) = 0$.

4 The CSALE Algorithm

In this section, we present CSALE, the proposed algorithm. The main idea of the algorithm is to perform successive acceptance, and to use the combinatorial structure of the decision class for elimination, thus considerably restricting the

number of required oracle calls. The algorithm starts with a large accuracy threshold, and iteratively attempts to accept arms while decreasing the threshold. Arms are accepted if they belong to the current empirically optimal solution (under the constraint that it must include all arms that have already been accepted) and have a large gap $\Delta(a)$. This is inspired by the acceptance criterion of graph sub-structures, proposed in (Ben-David and Sabato 2021) in the context of Bayesian structure learning.

The gap $\Delta(a)$ is estimated by finding an empirically optimal constrained solution that does not include a. If the gap is sufficiently large, the arm is accepted. In the final round, the accuracy parameter is set to guarantee the true desired accuracy ϵ, and the output arm-set is calculated, by estimating the rewards of the remaining candidate arms and maximizing the empirical score subject to all arms accepted so far.

To obtain a significant reduction in sample complexity compared to a naive uniform sampling baseline, it is helpful to eliminate arms from the list of candidates. However, as discussed above, directly estimating the gaps of arms to eliminate involves computationally demanding operations such as a large number of oracle calls. We avoid such direct estimation, and instead use the structure of the decision class itself. In many cases, this structure allows eliminating arms as a result of accepting some other arms. For instance, in the shortest path problem, edges (arms) that complete a cycle with accepted edges can be eliminated.

Formally, we define an *elimination function*, denoted $\mathrm{elim}_n : 2^{[n]} \to 2^{[n]}$, which gets as input the set of arms accepted so far, and outputs the set of arms that can be eliminated as candidates given the accepted arms. This function is implemented with respect to the specific decision class. Formally, we require

$$\mathrm{elim}_n(A) \subseteq [n] \setminus \bigcup_{M \in \mathcal{M}, A \subseteq M} M.$$

The function elim_n is used by CSALE to eliminate candidate arms after each acceptance. Note that elim_n is not required to always identify all possible eliminations, since this may be a computationally difficult task.

CSALE is listed in Alg. 1. It gets the confidence parameter δ and the required accuracy level ϵ. For $\tilde{\epsilon}, \tilde{\delta} \in (0,1)$, we denote by $N(\tilde{\epsilon}, \tilde{\delta}) := \lceil \log(2/\tilde{\delta})/(2\tilde{\epsilon}^2) \rceil$ the number of samples required by Hoeffding's inequality for estimating the expectation of random variable with support $[0,1]$, with a probability of at least $1-\tilde{\delta}$ and an error of no more than $\tilde{\epsilon}$. CSALE maintains a set of *active arms* $\mathcal{A}$. This is the set of arms that have not been accepted so far, but also have not been precluded from participating in the output solution. The set of accepted arms by iteration j is denoted Acc_j.

The algorithm works in rounds. At each round t, it pulls arms a sufficient number of times to obtain the required accuracy for that round, denoted ϵ_t. Denote by $\hat{\boldsymbol{\mu}}_t$ the vector of empirical estimates of expected arm rewards, based on the samples observed until round t. Calling COracle with $\hat{\boldsymbol{\mu}}_t$, CSALE calculates $\hat{M}_t$, the empirically optimal arm set at round t, out of the arm sets that are consistent with the arms that have been accepted so far. It then calculates the empirical gap of each active arm in $\hat{M}_t$, by calling COracle again, each time forbidding one of these arms.

In the next stage, every arm with an empirical gap above a calculated threshold θ_j is accepted. We show in the analysis that using this threshold, only arms in the optimal solution are accepted. Once an arm is accepted, it is removed from $\mathcal{A}$ along with all the arms that can be eliminated using elim_n. We note that when calling COracle, it is not necessary to explicitly ban the eliminated arms, since by definition, only arms that cannot share an arm-set with the accepted arms are eliminated. The rounds stop when the accuracy ϵ_t is sufficiently small. Note that the total number of rounds is at most $T - 1 = \lceil \log_2(d) \rceil$, where T is defined in line 1 in CSALE. After the rounds stop, if any arms are still active, they are pulled to obtain a final batch of samples. CSALE then returns an arm set that maximizes the empirical reward, subject to the constraints set by the arms accepted so far.

Number of oracle calls. It can be easily seen that except for oracle calls, CSALE does not involve any computationally demanding operations. Therefore, the number of oracle calls that it makes is a good proxy for its computational burden. In the worst case, CSALE calls the oracle once for every arm in $\hat{M}_t$, in every round $t \in [T-1]$. Therefore, the total number of oracle calls in CSALE is

Algorithm 1: CSALE: Combinatorial Successive Acceptance with Light Elimination

Input: $\delta \in (0,1)$; $\epsilon > 0$.
Output: An arm set $M \in \mathcal{M}$
1 **Initialize**: $\mathcal{A} \leftarrow [n]$, $\mathrm{Acc}_1 \leftarrow \emptyset$, $N_0 \leftarrow 0$, $t \leftarrow 1$, $j \leftarrow 1$, $\mathcal{M}_1 \leftarrow \mathcal{M}$, $\epsilon_1 \leftarrow \epsilon$, $\theta_1 \leftarrow d(\mathcal{M}_1) \cdot \epsilon_1$, $T \leftarrow \lceil \log_2(d) \rceil + 1$.
2 **while** $\epsilon_t > \epsilon/(d(\mathcal{M}_j) - |\mathrm{Acc}_j|)$ **do**
3 $N_t \leftarrow N(\epsilon_t/2, \delta/(T|\mathcal{A}|))$
4 Pull each arm in $\mathcal{A}$ for $N_t - N_{t-1}$ times
5 Update $\hat{\boldsymbol{\mu}}_t$ based on the results of the arm pulls
6 $\hat{M}_t \leftarrow \mathsf{COracle}(\hat{\boldsymbol{\mu}}_t, \mathrm{Acc}_j, \emptyset)$
7 **foreach** $a \in \hat{M}_t \cap \mathcal{A}$ **do**
8 $\widetilde{M}_t^a \leftarrow \mathsf{COracle}(\hat{\boldsymbol{\mu}}_t, \mathrm{Acc}_j, \{a\})$
9 $\hat{\Delta}(a) := \hat{\boldsymbol{\mu}}_t(\hat{M}_t) - \hat{\boldsymbol{\mu}}_t(\widetilde{M}_t^a)$
10 **while** $\exists a \in \hat{M}_t \cap \mathcal{A}$ *such that* $\hat{\Delta}(a) > \theta_j$ **do**
11 Let a be some arm that satisfies the condition
12 $\mathrm{Acc}_{j+1} \leftarrow \mathrm{Acc}_j \cup \{a\}$ # accept arm a
13 $\mathcal{A} \leftarrow \mathcal{A} \setminus \mathrm{elim}_n(\mathrm{Acc}_{j+1})$
14 $\mathcal{M}_{j+1} \leftarrow \{M \in \mathcal{M} \mid \mathrm{Acc}_{j+1} \subseteq M\}$
15 $\theta_{j+1} \leftarrow (d(\mathcal{M}_{j+1}) - |\mathrm{Acc}_{j+1}|) \cdot \epsilon_t$
16 $j \leftarrow j + 1$
17 **If** $\mathrm{Acc}_j \in \mathcal{M}_j$ **then return** $\hat{M} := \mathrm{Acc}_j$.
18 $t \leftarrow t + 1$
19 $\epsilon_t \leftarrow \epsilon_{t-1}/2; \theta_j \leftarrow \theta_j/2$.
20 $\epsilon_{\mathrm{last}} \leftarrow \epsilon/(d(\mathcal{M}_j) - |\mathrm{Acc}_j|)$
21 $N_T \leftarrow N(\epsilon_{\mathrm{last}}/2, \delta/(T|\mathcal{A}|))$
22 Pull arms in $\mathcal{A}$ for $N_T - N_{T-1}$ times; update $\hat{\boldsymbol{\mu}}_T$.
23 **Return** $\hat{M} \leftarrow \mathsf{COracle}(\hat{\boldsymbol{\mu}}_T, \mathrm{Acc}_j, \emptyset)$.

$O(dT) = O(d\log(d))$. This is contrasted with previous algorithms, in which the number of oracle calls is either similar to the sample complexity, or is a high degree polynomial. For instance, in CLUCB-PAC, the number of oracle calls is of order $\Omega(d^2n/\epsilon^2)$ in the worst case. In the next section, we provide the analysis of CSALE.

5 Analysis

In this section, we provide correctness and sample complexity analysis for CSALE. First, we define a uniform convergence event which will be used in the analysis. Denote by J the total number of acceptance iterations (see line 10 in Alg. 1) during the entire run of the algorithm. For any $j \in [J]$, let $t(j)$ be the round in which iteration j occurred. Let θ_j be as defined in line 15, and $\theta_J := (d(\mathcal{M}_J) - |\mathrm{Acc}_J|)\cdot\epsilon_{\mathrm{last}}$. Let η be the event such that $\forall j \in [J], \forall M \in \mathcal{M}_j$,

$$|\hat{\mu}_{t(j)}(M \setminus \mathrm{Acc}_j) - \mu(M \setminus \mathrm{Acc}_j)| \le \theta_j/2.$$

By a standard argument, η holds with a probability at least $1-\delta$.

To show that CSALE is indeed a PAC algorithm, we prove the following theorem. The proof is provided in the full version of this paper.

Theorem 5.1. *With probability at least $1-\delta$, the arm set $\hat{M}$ returned by CSALE satisfies $\mu(\hat{M}) \ge \mu^* - \epsilon$.*

We now study the sample complexity of CSALE. CSALE can reduce the number of samples compared to a naive uniform sampling solution if it accepts arms early, before the last sampling batch. An even larger reduction can be obtained if the elimination function elim_n for the given decision class leads to many arm eliminations. For $k \in \mathbb{N}$ such that $k \le n$, define the *elimination measure* of elim_n for a set of size k to be $Q(n,k) := k + \min_{A\subseteq[n]:|A|=k} |\mathrm{elim}_n(A)|$. Note that in many natural problems and elimination functions, $Q(n,k)$ grows with n. For instance, for maximum weight matching in a graph $G=(V,E)$, a natural elimination function would be $\mathrm{elim}_n(E') := \bigcup_{(u,v)\in E'}\{e \in E \mid v \text{ or } u \text{ are nodes of } e\}$. Recall that in this problem, $n = |E|$. Thus, if G is a full graph, we get $Q(n,k) = \Theta(k\sqrt{n})$.

Denote the set of optimal arm sets in $\mathcal{M}_j$ by $\mathcal{M}_j^* := \mathcal{M}^* \cap \mathcal{M}_j$, and the set of arms that are shared by all optimal solutions by $A^* := \bigcap_{M\in\mathcal{M}^*} M$. As shown in Lemma **??** in Appendix **??** (see the full version of this paper), only arms in A^* may be accepted early. For any $\beta > 0$ and $\square \in \{>,\ge,<,\le\}$ let $A^*_{\square\beta} = \{a \in A^* \mid \Delta(a)\square\beta\}$.

Theorem 5.2. *Let $Q(n,\cdot)$ be the elimination measure of elim_n. For $t \in [T-1]$, let $\epsilon_t = 2\epsilon/2^t$ be the accuracy level in round t of CSALE, and define $k_t := |A^*_{\ge 2d\epsilon_t} \setminus A^*_{\ge 2d\epsilon_{t-1}}|$, where $\epsilon_0 := \infty$. Define $\bar{k}_t := \sum_{i\in[t]} k_t$. Further define $\Delta_t := \max_{j\in A^*_{<2(d-\bar{k}_{t-1})\epsilon_{t-1}}} \Delta(j)$. The maximal number of samples required by CSALE is*

$$\widetilde{O}\Big(\sum_{t=1}^{T-1} \frac{(d-\bar{k}_{t-1})^2 Q(n,k_t)}{\Delta_t^2} + \frac{(d-\bar{k}_{T-1})^2(n - Q(n,\bar{k}_{T-1}))}{\epsilon^2}\Big), \tag{1}$$

where $\widetilde{O}$ suppresses logarithmic factors. In addition,

1. *For every $t \in [T-1]$, if $Q(n,k_t) > 0$, then $\Delta_t > 2\epsilon$;*
2. *$\bar{k}_{T-1} = |A^*_{\ge 4\epsilon}|$.*

Before proving the theorem, we discuss the meaning of these sample complexity guarantees. First, compare these guarantees to a naive uniform sampling algorithm, which simply pulls each arm $N(\epsilon/d, \delta/n)$ times and selects the empirically-best arm set. Such an algorithm requires $\tilde{\Theta}(d^2 n\log(1/\delta)/\epsilon^2)$ samples, similarly to the worst case of CSALE. This is a baseline requirement which ensures that using CSALE over naive sampling is never harmful. Like all CPE algorithms, the improvement of CSALE over the naive baseline is instance dependent.

We now compare the guarantees of CSALE to those of CLUCB-PAC (Chen et al. 2014), which is the only previous CPE algorithm for the PAC setting. The guarantees of the two algorithms are not directly comparable, since each algorithm has a different type of instance dependence. We now give an example of a simple case in which CSALE performs significantly better than CLUCB-PAC.

Example 5.3. Consider n arms, where $n > 10$ is an odd number, and let $s = (n-1)/2$. Let $\gamma > 0$. For every $a \in [s-1]$ we set $\mu_a = s\gamma$. We also set $\mu_s = \gamma$. All the other arms get a reward of zero. The decision class consists of three arm sets, each including exactly s arms: $\{1,\dots,s\}$, $\{1,\dots,s-1,s+1\}$, and $\{s+2,\dots,n\}$. Their respective rewards are $(s^2-s+1)\gamma$, $(s^2-s)\gamma$, and 0. Thus, the reward difference between the optimal solution and the second-best solution is γ. Set $\epsilon := \gamma$, so that the PAC algorithm is required to find the optimal solution.

The worst case sample complexity of CLUCB-PAC is $\widetilde{O}\left(\sum_{a\in A}\min\{\frac{\mathsf{width}^2(\mathcal{M})}{\Delta(a)^2}, \frac{d^2}{\epsilon^2}\}\right)$, where $\mathsf{width}(\mathcal{M})$ is a combinatorial property of the decision class. In our example, we have $d = s = \Theta(n)$ and $\mathsf{width}(\mathcal{M}) = 2s = \Theta(n)$. In addition, for all $a \in [n]\setminus\{s,s+1\}$, we have $\Delta(a) = (s^2-s+1)\gamma = \Theta(n^2\gamma)$, while $\Delta(s) = \Delta(s+1) = \gamma$. The sample complexity guarantee of CLUCB-PAC is thus $O(n^2/\gamma^2)$.

We now bound the sample complexity of CSALE, assuming that elim_n is maximal. We have $A^* = [s]$. By Theorem 5.2, claim 2, $\bar{k}_{T-1} = |A^*_{\ge 4\gamma}| = s-1$. In addition, since all arms in $A^*_{\ge 4\gamma}$ have the same gap, and $\bar{k}_{T-1} = \sum_{t\in[T-1]} k_t$ by definition, then there exists some $t_0 \in [T-1]$ for which $k_{t_0} = \bar{k}_{T-1} = s-1$, while $k_i = 0$ for $i \ne t_0$. By the definition of Q, we have that $Q(n,k_{t_0}) = Q(n,s-1) > 0$, and $Q(n,k_i) = 0$ for $i \ne t_0$. In addition, $\bar{k}_{t_0-1} = 0$. By Theorem 5.2, claim 1, we have that $\Delta_{t_0} > 2\gamma$. Since Δ_{t_0} is a gap of an arm in A^*, we have $\Delta_{t_0} = \Theta(n^2\gamma)$. From the definition of the elimination function, we have $Q(n,s-1) = s-1+s = n-2$. Substituting these quantities in Eq. (1), we get that the sample complexity of CSALE for this problem is $\widetilde{O}(1/\gamma^2)$, while the sample complexity of CLUCB-PAC is a factor of n^2 greater in this example. □

We note that the CPE algorithms of Chen et al. (2017); Cao and Krishnamurthy (2019) obtain a smaller sample

complexity than CLUCB-PAC, by applying a different sampling and convergence analysis technique. Their sample complexity is comparable with that of CSALE in the example above, but it can also be smaller than that of CSALE. However, these algorithms do not support the PAC setting, and their sampling technique is highly computationally demanding. An interesting question for future work is whether similar sample complexity improvements can be obtained within our framework, while keeping the algorithm computationally light.

We now turn to the proof of Theorem 5.2. First, we prove two useful lemmas. The first lemma gives a sufficient condition for an arm to be accepted by CSALE at a given iteration.

Lemma 5.4. *Assume that the event η defined above holds for a run of CSALE. Let j be an iteration in the run in some round t and let θ_j be as defined in Alg. 1. Let $\mathcal{A}_j$ be the value of the set of active arms $\mathcal{A}$ at the start of iteration j. If there exists some arm in $a \in A^* \cap \mathcal{A}_j$ such that $\Delta(a) > 2\theta_j$, then some arm is accepted by CSALE at iteration j in this round.*

Proof. Under the assumption of the lemma, $2\theta_j < \Delta(a)$. To prove the lemma, we show that $a \in \hat{M}_t$ and that $\hat{\Delta}(a) > \theta_j$. This implies that the condition in line 10 of CSALE holds at iteration j, and so some arm must be accepted in this iteration.

Let $M^* \in \mathcal{M}^*$ be an optimal arm set. By definition, $A^* \subseteq M^*$ and so $a \in M^*$. By Lemma **??**, $\mathrm{Acc}_j \subseteq A^*$. Thus, $\mathcal{M}^*_j = \mathcal{M}^*$ and $M^* \in \mathcal{M}_j$. Therefore, $\hat{\mu}_t(\hat{M}_t) \geq \hat{\mu}_t(M^*)$. For every $M \in \mathcal{M}_j$ we have that $\mu(M) = \mu(M \setminus \mathrm{Acc}_j) + \mu(\mathrm{Acc}_j)$, and the same holds for $\hat{\mu}_{t(j)}$. Therefore, by η, $\forall M, M' \in \mathcal{M}_j$, we have that

$$\hat{\mu}_{t(j)}(M) - \hat{\mu}_{t(j)}(M') \geq \mu(M) - \mu(M') - \theta_j. \quad (2)$$

In particular, this holds for $\hat{M}_t$ and M^*. Since we have $\hat{\mu}_t(\hat{M}_t) - \hat{\mu}_t(M^*) \geq 0$, it follows that $\mu^* - \mu(\hat{M}_t) \leq \theta_j$. Since $\theta_j < \Delta(a) = \mu^* - \max_{M: a \notin M} \mu(M)$, it follows that $\mu(\hat{M}_t) > \max_{M: a \notin M} \mu(M)$. Hence, $a \in \hat{M}_t$. To prove the second part, observe that

$$\begin{aligned}\hat{\Delta}(a) &= \hat{\mu}_t(\hat{M}_t) - \hat{\mu}_t(\widetilde{M}^a_t) \geq \hat{\mu}_t(M^*) - \hat{\mu}_t(\widetilde{M}^a_t) \\ &\geq \mu^* - \mu(\widetilde{M}^a_t) - \theta_j \geq \Delta(a) - \theta_j > \theta_j.\end{aligned}$$

In the second line, we used Eq. (2). This completes the proof. □

The second lemma gives a condition for early acceptance of arms in CSALE. Without loss of generality, suppose that $A^* = [v]$ for some $v \leq n$, and that the arms are ordered so that $\Delta(1) \geq \Delta(2) \geq \cdots \geq \Delta(v)$. Note that by the definition of A^*, $\Delta(v) > 0$.

Lemma 5.5. *For $i \in [v]$, let $A^*_i = \{a \in A^* \mid \Delta(a) \geq \Delta(i)\}$. Consider a run of CSALE in which η holds. Then CSALE accepts at least $|A^*_i| \geq i$ arms until the end of the first round t which satisfies $\epsilon_t \leq \Delta(i)/(2d)$.*

Proof. Suppose that CSALE has accepted fewer than $|A^*_i|$ arms until some iteration j. Then $A^*_i \setminus \mathrm{Acc}_j \neq \emptyset$. By Lemma **??**, $\mathrm{Acc}_j \subseteq A^*$. Therefore, $\mathcal{M}^* \subseteq \mathcal{M}_j$, which means that all arms in A^* that were not accepted are still active. Hence, $A^*_i \cap \mathcal{A}_j \neq \emptyset$, where $\mathcal{A}_j$ is as defined in Lemma 5.4. Let $a \in A^*_i \cap \mathcal{A}$. By the definition of A^*_i, $\Delta(a) \geq \Delta(i)$. The conditions of Lemma 5.4 thus hold if $2\theta_j < \Delta(i)$. This condition holds throughout the first round t that satisfies $2d\epsilon_t \leq \Delta(i)$. In this case, some arm will be accepted at iteration j. Round t only ends when no additional arms are accepted. Therefore, at least $|A^*_i|$ arms will be accepted until the end of this round. □

Using the lemmas above, we now prove Theorem 5.2.

Proof of Theorem 5.2. Let B_t be the number of arms that were accepted during round t. Consider the arms whose last round of being pulled by CSALE is round t, and note that their number is at least $Q(n, B_t)$, the number of arms that are accepted or eliminated in round t. Therefore, the sample complexity of CSALE is upper bounded by the expression $\sum_{t\in[T-1]} Q(n, B_t)N_t + (n - Q(n, B_T))N_T$, where $B_T = \sum_{t\in[T-1]} B_t$, A_t is the size of the active set $\mathcal{A}$ at the beginning of round t, and

$$N_t \equiv N(\epsilon_t/2, \delta/(T \cdot A_t)) = \widetilde{O}\big(\frac{1}{\epsilon_t^2}(\log(n/\delta) + \log\log(d)\big).$$

To bound this expression, observe that for every $t \in [T-1]$, letting $\alpha_t := 2(d - \bar{k}_{t-1})\epsilon_{t-1}$,

$$\Delta_t := \max_{j \in A^*_{<\alpha_t}} \Delta(j) < \alpha_t = 4(d - \bar{k}_{t-1})\epsilon_t.$$

Therefore, $\epsilon_t > \Delta_t/(4(d - \bar{k}_{t-1}))$. Hence, for $t \in [T-1]$, $N_t = \tilde{O}\big(\frac{(d-\bar{k}_{t-1})^2}{\Delta_t^2} \cdot (\log(n/\delta) + \log\log(d)\big)$.

Next, we bound B_t. By the definition of k_t in the theorem statement, there are k_t arms in A^* with a gap between $2d\epsilon_t$ and $2d\epsilon_{t-1}$. By Lemma 5.5, at least k_t arms must be accepted by the end of round t. Therefore, since N_t is monotonically increasing with t, the sample complexity can be upper bounded by $\sum_{t\in[T-1]} Q(n, k_t)N_t + (n - Q(n, \bar{k}_{T-1}))N_T$. Moreover, for the last round T, we have $N_T = \tilde{O}\big(\frac{(d-|\mathrm{Acc}_J|)^2}{\epsilon^2}(\log(n/\delta) + \log\log(d))\big)$, and $|\mathrm{Acc}_J| \geq \bar{k}_{T-1}$. Combining the above and suppressing logarithmic factors, we get the sample complexity upper bound:

$$\widetilde{O}\Big(\big(\sum_{t=1}^{T-1} \frac{(d - \bar{k}_{t-1})^2 Q(n, k_t)}{\Delta_t^2} + \frac{(d - \bar{k}_{T-1})^2(n - Q(n, \bar{k}_{T-1}))}{\epsilon^2}\Big).$$

To complete the proof, we prove claims 1 and 2, which are listed in the theorem statement. To prove claim 1, let $t \in [T-1]$ be some round such that $Q(n, k_t) > 0$. By the definition of Q, it follows that $k_t > 0$. This means that there exists some arm $l \in A^*_{\geq 2d\epsilon_t} \setminus A^*_{\geq 2d\epsilon_{t-1}}$. Therefore, $\epsilon_t \leq \Delta(l)/(2d) < \epsilon_{t-1}$. Therefore, by the definition of Δ_t, $\Delta(l) \leq \Delta_t$. By the condition of the main loop in CSALE, we have that $\epsilon_t > \epsilon/d$. Combining the last two inequalities, we get that $\Delta_t > 2\epsilon$. To prove claim 2, note that $\bar{k}_{T-1} = \sum_{t\in[T-1]} k_t = |\bigcup_{t\in[T-1]} A^*_{\geq 2d\epsilon_t}| = |A^*_{\geq 2d\epsilon_{T-1}}|$, and $2d\epsilon_{T-1} = 2d(2\epsilon/2^{T-1})$. Substituting $T - 1 = \log_2(d)$, we get that $2d\epsilon_{T-1} = 4\epsilon$, which completes the proof. □

6 Experiments

In this section, we report experiments comparing CSALE to CLUCB−PAC and to the naive baseline algorithm described in Section 5. As noted in Section 1 and Section 2, there are no other known PAC-CPE algorithms, and algorithms for other settings are also highly computationally demanding. All the experiments were run on a standard PC. The code for the algorithms and for the experiments below is provided at https://github.com/noabdavid/csale.

We ran the algorithms on the two types of graph problems mentioned in Section 1, s-t shortest path and maximum weight matching in general graphs. All three algorithms require an optimization oracle. We used Dijkstra's algorithm as implemented in the `Dijkstar` package[1] for the s-t shortest path problem, and the algorithm of Galil (1986) as implemented in the `Networkx` package (Hagberg, Swart, and Schult 2008) for maximum weight matching in general graphs. The matching oracle is computationally heavier, thus we tested the matching problem on smaller graphs than those used for the shortest path problem. The properties and sources of all the data sets that we used are provided in the full version of this paper. We used graphs with up to thousands of nodes and tens of thousands of edges for the shortest path experiments, and graphs with up to hundreds of nodes and thousands of edges for the matching experiments. $\mathrm{elim}_n(E')$ for the shortest path problem was implemented to return all the edges that have the same start node or end node as an edge in E'. For the matching problem, it returned all the edges that share some node with some edge in E'.

In all of the experiments, we set $\delta = 0.05$. Each experiment was repeated 10 or 100 times, depending on its computational requirements, as detailed in each result table below. In all experiments, the weight of the edge was used as the parameter of a Bernoulli distribution describing its instantaneous rewards. In synthetic or unweighted graphs, the edge weights were sampled uniformly at random out of $\{0.1, 0.5, 0.9\}$ in each run. We tested a large range of ϵ values in each experiment. In all the reported experiments, all algorithms indeed found an ϵ-optimal solution.

In the first set of experiments, we compared CLUCB−PAC and CSALE on small graphs, in terms of both sample size and computation requirements. The graph sizes were kept small due to the high computational requirements of CLUCB−PAC. We tested the following graphs:

- Shortest path, synthetic: A synthetic graph with 14 nodes and 16 edges, with a topology of 4 disjoint paths of length 4 between the source and the target nodes (see illustration in the full version of this paper).
- Matching, synthetic: A full graph with 6 nodes.
- Shortest path, real: A source and a target were randomly drawn from the full graph of the USAir97 data set, and the smallest sub-tree that includes both nodes was extracted. If the number of edges between the source and the target was at least 4 and the number of nodes was at most 10, then the resulting graph was used for one of the repetitions of the experiment.
- Matching, real: 6 nodes were randomly drawn from the full graph of the USAir97 data set, and their sub-graph was extracted. If it included more than four edges, it was used for one of the repetitions of the experiment.

Table 1 reports the results of the comparison with CLUCB−PAC for the two smallest values of ϵ that were tested in these experiments. The results for all values of ϵ are provided in the full version of this paper. It can be seen that CLUCB−PAC usually has the best sample complexity, although in most cases CSALE also obtains a significant improvement over the Naive baseline. The important advantage of CSALE can be seen when comparing the number of oracle calls and the total run time of the algorithm. These are usually 5 orders of magnitude larger for CLUCB−PAC compared to CSALE. Moreover, for CLUCB−PAC, this number grows linearly with the number of samples. This is contrasted with CSALE, in which the computational burden remains similar regardless of the sample size. For this reason, running CLUCB−PAC on large graphs is impractical, while CSALE can be easily used.

Next, we tested CSALE on larger graphs, and compared it to the Naive baseline. As mentioned above, it was impractical to run CLUCB−PAC on these larger problems. Table 2 reports the results for $\epsilon = 0.001$ (the smallest value of ϵ that we tested) for all the graphs that were tested for each problem. The results for all values of ϵ are provided in the full paper. It can be seen in Table 2 that the sample size required by CSALE is significantly smaller than that of the naive baseline. The run time of CSALE is small even on large networks with a very large numbers of samples.

We demonstrate the dependence on ϵ in Table 3, which lists the results for all values of ϵ for the p2p-Gnutella08 network. As ϵ becomes smaller, the sample size saving by CSALE (see the "sample size ratio" column) becomes more significant, up to a problem-dependent saturation point. The "accepted early" column indicates the fraction of the arms in the solution that were accepted early. When this ratio can no longer increase due to the problem structure, additional improvements are not possible.

Overall, our results show that unlike previous CPE algorithms, CSALE is practical to run on large problems, due to its light computational requirements and the independence of the number of oracle calls from the number of arm pulls. When ϵ is sufficiently small, it provides a significant reduction in the number of samples over the baseline.

7 Discussion

We propose a new algorithm for PAC-CPE. This algorithm has the advantage that it is computationally light, thus allowing it to run on large problems, while reducing the sample complexity compared to the baseline.

[1] https://github.com/wylee/Dijkstar

Experiment	Sample size $\times 10^4$ Naive	CLUCB-PAC	CSALE	Oracle calls CLUCB-PAC	CSALE	Time (millisec) CLUCB-PAC	CSALE
$\epsilon = 0.0625$							
path, synthetic	339 ± 0	$\mathbf{14} \pm 9$	25 ± 0	$29 \pm 17 \times 10^4$	$\mathbf{2} \pm 1$	$12 \pm 7 \times 10^3$	$\mathbf{0.5} \pm 0$
matching, synthetic	177 ± 0	$\mathbf{16} \pm 8$	23 ± 0	$32 \pm 15 \times 10^4$	$\mathbf{4} \pm 0$	$12 \pm 9 \times 10^4$	$\mathbf{2} \pm 1$
path, USAir97	490 ± 350	$\mathbf{190} \pm 160$	530 ± 470	$37 \pm 32 \times 10^5$	$\mathbf{7} \pm 2$	$13 \pm 12 \times 10^4$	$\mathbf{1} \pm 0$
matching, USAir97	$\mathbf{22} \pm 0$	30 ± 2	$\mathbf{22} \pm 0$	$60 \pm 3 \times 10^4$	$\mathbf{1} \pm 0$	$96 \pm 5 \times 10^3$	$\mathbf{0.3} \pm 0$
$\epsilon = 0.03125$							
path, synthetic	1355 ± 0	$\mathbf{16} \pm 10$	99 ± 0	$33 \pm 21 \times 10^4$	$\mathbf{2} \pm 1$	$13 \pm 9 \times 10^3$	$\mathbf{0.4} \pm 0$
matching, synthetic	707 ± 0	$\mathbf{19} \pm 9$	92 ± 0	$37 \pm 18 \times 10^4$	$\mathbf{4} \pm 0$	$15 \pm 11 \times 10^4$	$\mathbf{2} \pm 1$
path, USAir97	1960 ± 1410	$\mathbf{900} \pm 750$	2120 ± 1880	$18 \pm 15 \times 10^6$	$\mathbf{7} \pm 2$	$72 \pm 63 \times 10^4$	$\mathbf{1} \pm 1$
matching, USAir97	87 ± 0	$\mathbf{52} \pm 2$	87 ± 0	$10 \pm 0 \times 10^5$	$\mathbf{1} \pm 0$	$17 \pm 1 \times 10^4$	$\mathbf{0.3} \pm 0$

Table 1: A comparison of all three algorithms on small graphs. The synthetic and real experiments were repeated 100 and 10 times, respectively.

Type	Graph ID (see Table **??**)	Sample size Naive	CSALE	Sample size ratio (CSALE/ Naive)	CSALE Oracle calls	CSALE Time (millisec)
Shortest Path	1	33×10^{10}	$\mathbf{25} \pm 16 \times 10^{10}$	76%	5 ± 2	4 ± 2
	2	86×10^{10}	$\mathbf{22} \pm 28 \times 10^{10}$	26%	5 ± 3	5 ± 4
	3	117×10^{10}	$\mathbf{63} \pm 60 \times 10^{10}$	54%	5 ± 3	10 ± 5
	4	114×10^{10}	$\mathbf{49} \pm 56 \times 10^{10}$	43%	4 ± 2	9 ± 4
	5	33×10^{11}	$\mathbf{22} \pm 16 \times 10^{11}$	67%	6 ± 3	29 ± 12
	6	34×10^{11}	$\mathbf{12} \pm 15 \times 10^{11}$	35%	4 ± 2	26 ± 12
	7	215×10^{11}	$\mathbf{88} \pm 98 \times 10^{11}$	41%	12 ± 8	48 ± 30
	8	$13 \pm 16 \times 10^{13}$	$\mathbf{10} \pm 17 \times 10^{13}$	77%	13 ± 7	79 ± 47
	9	116×10^{13}	$\mathbf{21} \pm 38 \times 10^{13}$	18%	17 ± 14	830 ± 626
	10	142×10^{13}	$\mathbf{26} \pm 48 \times 10^{13}$	18%	17 ± 15	852 ± 587
	11	73×10^{13}	$\mathbf{12} \pm 23 \times 10^{13}$	16%	16 ± 12	518 ± 333
	12	1153×10^{12}	$\mathbf{90} \pm 278 \times 10^{12}$	8%	13 ± 11	561 ± 373
Matching	1	84×10^{10}	$\mathbf{18} \times 10^{10}$	21%	22 ± 0	102 ± 7
	2	468×10^{10}	$\mathbf{15} \times 10^{10}$	3%	18 ± 0	204 ± 3
	3	244×10^{11}	$\mathbf{66} \times 10^{11}$	27%	34 ± 2	622 ± 43
	4	22×10^{12}	$\mathbf{17} \times 10^{12}$	77%	66 ± 2	756 ± 60
	5	177×10^{12}	$\mathbf{57} \times 10^{12}$	32%	69 ± 3	3500 ± 100
	6	18×10^{13}	$\mathbf{16} \times 10^{13}$	89%	85 ± 4	2700 ± 300
	7	317×10^{13}	$\mathbf{59} \pm 1 \times 10^{13}$	19%	356 ± 7	$28 \pm 1 \times 10^4$
	8	238×10^{13}	$\mathbf{34} \times 10^{13}$	14%	307 ± 1	$20 \pm 1 \times 10^4$

Table 2: Experiments with larger graphs, with $\epsilon = 0.001$. Standard deviations are omitted when they are smaller than 1% of the average. Shortest paths and matching experiments were repeated 100 and 10 times, respectively.

ϵ	Sample size Naive	CSALE	Sample size ratio (CSALE/ Naive)	CSALE Oracle calls	CSALE Time (millisec)	CSALE Accepted early
0.50000	29×10^8	$\mathbf{28} \pm 4 \times 10^8$	97%	48 ± 14	$13 \pm 3 \times 10^2$	$17\% \pm 19\%$
0.25000	12×10^9	$\mathbf{10} \pm 2 \times 10^9$	83%	46 ± 14	$12 \pm 3 \times 10^2$	$31\% \pm 26\%$
0.12500	47×10^9	$\mathbf{33} \pm 13 \times 10^9$	70%	42 ± 14	$11 \pm 3 \times 10^2$	$51\% \pm 33\%$
0.06250	188×10^9	$\mathbf{94} \pm 60 \times 10^9$	50%	37 ± 13	$10 \pm 3 \times 10^2$	$66\% \pm 32\%$
0.03125	75×10^{10}	$\mathbf{19} \pm 21 \times 10^{10}$	25%	30 ± 13	832 ± 296	$85\% \pm 28\%$
0.01000	73×10^{11}	$\mathbf{13} \pm 22 \times 10^{11}$	18%	20 ± 13	592 ± 295	$85\% \pm 27\%$
0.00500	294×10^{11}	$\mathbf{49} \pm 91 \times 10^{11}$	17%	16 ± 12	494 ± 285	$85\% \pm 27\%$
0.00100	73×10^{13}	$\mathbf{12} \pm 23 \times 10^{13}$	16%	16 ± 12	518 ± 333	$85\% \pm 28\%$

Table 3: p2p-Gnutella08 network results. Each experiment was repeated 100 times.

References

Ben-David, N.; and Sabato, S. 2021. Active Structure Learning of Bayesian Networks in an Observational Setting. *arXiv preprint arXiv:2103.13796*.

Cao, T.; and Krishnamurthy, A. 2019. Disagreement-based combinatorial pure exploration: Sample complexity bounds and an efficient algorithm. In *Conference on Learning Theory*, 558–588.

Chaudhuri, A. R.; and Kalyanakrishnan, S. 2019. PAC identification of many good arms in stochastic multi-armed bandits. In *International Conference on Machine Learning*, 991–1000.

Chen, L.; Gupta, A.; and Li, J. 2016. Pure exploration of multi-armed bandit under matroid constraints. In *Conference on Learning Theory*, 647–669.

Chen, L.; Gupta, A.; Li, J.; Qiao, M.; and Wang, R. 2017. Nearly optimal sampling algorithms for combinatorial pure exploration. In *Conference on Learning Theory*, 482–534.

Chen, S.; Lin, T.; King, I.; Lyu, M. R.; and Chen, W. 2014. Combinatorial pure exploration of multi-armed bandits. *Advances in neural information processing systems*, 27: 379–387.

Chen, W.; Du, Y.; Huang, L.; and Zhao, H. 2020. Combinatorial pure exploration for dueling bandit. In *International Conference on Machine Learning*, 1531–1541.

Gabillon, V.; Lazaric, A.; Ghavamzadeh, M.; Ortner, R.; and Bartlett, P. 2016. Improved learning complexity in combinatorial pure exploration bandits. In *Artificial Intelligence and Statistics*, 1004–1012.

Galil, Z. 1986. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys*, 18(1): 23–38.

Hagberg, A.; Swart, P.; and Schult, D. 2008. Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Kalyanakrishnan, S.; Tewari, A.; Auer, P.; and Stone, P. 2012. PAC subset selection in stochastic multi-armed bandits. In *International Conference on Machine Learning*, volume 12, 655–662.

Marcotte, R. J.; Wang, X.; Mehta, D.; and Olson, E. 2020. Optimizing multi-robot communication under bandwidth constraints. *Autonomous Robots*, 44(1): 43–55.

Schumann, C.; Lang, Z.; Foster, J.; and Dickerson, J. 2019. Making the cut: A bandit-based approach to tiered interviewing. In *Advances in Neural Information Processing Systems*, volume 32.

Talebi, M. S.; Zou, Z.; Combes, R.; Proutiere, A.; and Johansson, M. 2017. Stochastic online shortest path routing: The value of feedback. *IEEE Transactions on Automatic Control*, 63(4): 915–930.

Xue, Y.; Zhou, P.; Mao, S.; Wu, D.; and Zhou, Y. 2018. Pure-exploration bandits for channel selection in mission-critical wireless communications. *IEEE Transactions on Vehicular Technology*, 67(11): 10995–11007.

Zhou, Y.; Chen, X.; and Li, J. 2014. Optimal PAC multiple arm identification with applications to crowdsourcing. In *International Conference on Machine Learning*, 217–225.