Optimal Clustering with Noisy Queries via Multi-Armed Bandit

Jinghui Xia [1]   **Zengfeng Huang** [1] [2]

Abstract

Motivated by many applications, we study clustering with a faulty oracle. In this problem, there are n items belonging to k unknown clusters, and the algorithm is allowed to ask the oracle whether two items belong to the same cluster or not. However, the answer from the oracle is correct only with probability $\frac{1}{2} + \frac{\delta}{2}$. The goal is to recover the hidden clusters with minimum number of noisy queries. Previous works have shown that the problem can be solved with $O(\frac{nk \log n}{\delta^2} + \text{poly}(k, \frac{1}{\delta}, \log n))$ queries, while $\Omega(\frac{nk}{\delta^2})$ queries is known to be necessary. So, for any values of k and δ, there is still a non-trivial gap between upper and lower bounds. In this work, we obtain the first matching upper and lower bounds for a wide range of parameters. In particular, a new polynomial time algorithm with $O(\frac{n(k + \log n)}{\delta^2} + \text{poly}(k, \frac{1}{\delta}, \log n))$ queries is proposed. Moreover, we prove a new lower bound of $\Omega(\frac{n \log n}{\delta^2})$, which, combined with the existing $\Omega(\frac{nk}{\delta^2})$ bound, matches our upper bound up to an additive $\text{poly}(k, \frac{1}{\delta}, \log n)$ term. To obtain the new results, our main ingredient is an interesting connection between our problem and multi-armed bandit, which might provide useful insights for other similar problems.

1. Introduction

In this paper, we study clustering with noisy queries, a.k.a. clustering with a faulty oracle. In this problem, there are n vertices (or items) belonging to k unknown clusters, and the algorithm is allowed to ask the oracle whether two vertices belong to the same cluster or not. However, the answer from the oracle is correct only with probability $\frac{1}{2} + \frac{\delta}{2}$. The goal is to recover the hidden clusters with minimum number of noisy queries. This elegant theoretical model is first proposed in (Mazumdar & Saha, 2017a), which is motivated by many application scenarios, e.g., *crowdsourced entity resolution* (CER). Entity resolution aims to identify all records in a database that refer to the same underlying entity, which is a fundamental data mining task (Fellegi & Sunter, 1969; Elmagarmid et al., 2007; Getoor & Machanavajjhala, 2012). Based on crowdsourcing platforms like Amazon Mechanical Turk, leveraging human knowledge to improve the accuracy of automated ER techniques is a promising direction (Karger et al., 2011; Wang et al., 2012; Dalvi et al., 2013; Vesdapunt et al., 2014; Mazumdar & Saha, 2017b). The answers from crowd workers are often very noisy (Prelec et al., 2017) and the goal of CER is to recover entities with minimum number of queries. Thus, clustering with noisy queries well captures CER. See (Mazumdar & Saha, 2017a) for discussions on more real applications, e.g., signed edge prediction and correlation clustering.

On the theory side, the model is intimately connected with the stochastic block model (SBM). SBM is a popular random graph model for studying clustering and community detection algorithms, and plays a key role in studying statistical and computational tradeoffs for many statistical problems (McSherry, 2001; Abbe, 2018). In SBM, n vertices belong to k hidden clusters; each inter-cluster edge exists with probability q and each intra-cluster edge exists with probability p, where $0 \le q < p \le 1$. Given a graph sampled from the above model, the goal is to recover all hidden clusters. When $q = \frac{1}{2} - \frac{\delta}{2}$ and $p = \frac{1}{2} + \frac{\delta}{2}$, SBM is equivalent to clustering with noisy queries, except that SBM always observes all $O(n^2)$ queries while our goal is to use $o(n^2)$ queries.

Problem definition. Given a set V of n vertices (or items), there is an unknown partition $\{V_1, ..., V_k\}$ of V and each vertex i belongs to a unique cluster $V_{c(i)}$. Define a function $\tau : V \times V \to \pm 1$ such that $\tau(u, v) = 1$ if u, v belong to the same cluster and $\tau(u, v) = -1$ if u, v belong to different clusters. To know whether a pair (u, v) is in the same cluster, the algorithm can make queries to an oracle, but the oracle will only return the correct answer with probability $\frac{1}{2} + \frac{\delta}{2}$ for some $\delta \in (0, 1)$. Equivalently, the algorithm has access to a noisy version of τ, denoted as $\tilde{\tau}$:

$$\tilde{\tau}(u, v) = \sigma_{u,v} \cdot \tau(u, v),$$

---

[1]School of Data Science, Fudan University, Shanghai, China [2]Shanghai Key Lab of Intelligent Information Processing. Correspondence to: Zengfeng Huang <huangzf@fudan.edu.cn>.

where $\sigma_{u,v}$ is a $\{\pm 1\}$ random variable attaining $+1$ with probability $\frac{1}{2} + \frac{\delta}{2}$. We assume that $\sigma_{u,v}$ is independent across different pairs and the oracle always returns the same answer for queries to the same pair. The goal is to recover V_i, $i = 1, ..., k$ with high probability by making minimum number of queries to the oracle.

1.1. Previous Results

Mazumdar & Saha (2017a) gave an super-polynomial time algorithm with $O(\frac{nk \log n}{\delta^2})$ queries, which recovers all clusters of size $\Omega(\frac{\log n}{\delta^2})$. Then they presented a polynomial time algorithm using $O(\frac{nk \log n}{\delta^2} + \min\{\frac{nk^2 \log n}{\delta^4}, \frac{k^5 \log^2 n}{\delta^8}\})$ queries, which recovers all clusters of size $\Omega(\frac{k \log n}{\delta^4})$. In the same paper, an information-theoretic lower bound of $\Omega(\frac{nk}{\delta^2})$ queries is proved. Later, Larsen et al. (2020) focused on the two-cluster case, and proposed an algorithm with query complexity $O(\frac{n \log n}{\delta^2} + \frac{\log^2 n}{\delta^6})$. Very recently, Peng & Zhang (2021) focused on improving the dependency on δ, and gave a polynomial time algorithm with query complexity $O(\frac{nk \log n}{\delta^2} + \frac{k^{10} \log^2 n}{\delta^4})$, which recovers all clusters of size $\Omega(\frac{k^4 \log n}{\delta^2})$. Compared to (Mazumdar & Saha, 2017a; Larsen et al., 2020), the dependence on δ in the second term is improved to $1/\delta^4$. Note, the lower bound suggests that the setting is meaningful only if $\frac{1}{\delta} \leq n^{1/2}$, since otherwise $\Omega(n^2)$ queries are required. Thus, $\frac{1}{\delta^4} \leq \frac{n}{\delta^2}$, and if we ignore k and $\log n$, the dependence on δ is optimal.

However, for any k and δ, there is still a non-trivial gap between upper and lower bounds. In particular, whether the first term $O(\frac{nk \log n}{\delta^2})$ can be improved, even without running time constraint, remains an unanswered question. As this term dominates the complexity for small $k, 1/\delta$ (which is likely the case), completely settling the question is critical to understanding the nature of the problem. In this paper, we fully resolve this question.

1.2. Our Contribution

We propose a new polynomial time algorithm with $O(\frac{n(k+\log n)}{\delta^2} + \frac{k^8 \log^3 n}{\delta^4})$ queries, which is the first improvement on the first term since (Mazumdar & Saha, 2017a). The dependence on k in the second term is also slightly improved.

Theorem 1.1. *There exists a polynomial time algorithm that recovers all the clusters of size $\Omega(\frac{k^4 \log n}{\delta^2})$ with success probability $1 - o_n(1)$. The total number of queries to the faulty oracle is $O(\frac{n(k+\log n)}{\delta^2} + \frac{k^8 \log^3 n}{\delta^4})$.*

Since $\Omega(\frac{nk}{\delta^2})$ is a lower bound, our algorithm is optimal when $k = \Omega(\log n)$. But for $k = o(\log n)$, there is still a non-negligible gap, and it is previously unknown whether $\frac{n \log n}{\delta^2}$ is necessary for constant k. To this end, we prove a new information-theoretic lower bound for any $k \geq 2$,

which matches our upper bound for all k.

Theorem 1.2. *For $k \geq 2$, any (randomized) algorithm must make $\Omega(\frac{n \log n}{\delta^2})$ expected number of queries to recover the correct clusters with probability at least $\frac{7}{8}$.*

To obtain those improved results, we introduce several new techniques. For the upper bound, we follow the framework of (Peng & Zhang, 2021), but make several critical changes. The most important new ingredient is to utilize multi-armed bandit algorithms to identify true clusters of vertices. Our lower bound proof is also quite different from (Mazumdar & Saha, 2017a). Specifically, we introduce a new variant of best arm identification, and prove a tight lower bound on the sample complexity; then we give a novel reduction from this problem to clustering with noisy queries.

1.3. Other Related Work

Clustering is a central problem in data mining and machine learning; numerous algorithms have been proposed, e.g., (McLachlan & Basford, 1988; Karypis & Kumar, 1998; Ng et al., 2001; Brandes et al., 2003; Arthur & Vassilvitskii, 2007). However, clustering problems are often computationally hard. With the emergence of crowdsourcing platforms, the humans-in-the-loop approaches become a popular research direction. One line closely related to ours is the semi-supervised active clustering framework (Ashtiani et al., 2016). Their goal is to incorporate same-cluster queries in specific clustering problems to alleviate the computational hardness, e.g., k-means (Chien et al., 2018; Bressan et al., 2020; Li et al., 2021), correlation clustering (Ailon et al., 2018; Saha & Subramanian, 2019).

2. Preliminaries and Tools

We first introduce two important definitions.

Definition 2.1. Let $b \in [0, 1]$ and V be a vertex set. We call a partition $V_1, ..., V_k$ of V b-balanced, if $|V_i| \geq \frac{bn}{k}, \forall i \in [k]$. An input instance V is b-balanced if its hidden clusters form a b-balanced partition.

Definition 2.2. Let $\eta \in [0, \frac{1}{2}]$. Let $C = V_i$ be a true cluster of V for some $i \in [k]$. A set of vertices $B \subseteq V$ is called (η, C)-biased if at least $1/2 + \eta$ fraction of the vertices in B belong to C, i.e. $|B \cap C| \geq (1/2 + \eta) \cdot |B|$. When $\eta = 1/2$, i.e., B is a subset of some cluster, B is called a sub-cluster.

Balanced clustering in stochastic block model. Fix a k-partition $V_1, ..., V_k$ of V and a parameter $\delta \in [0, 1)$, the distribution $\mathcal{D}(V_1, ..., V_k, \delta)$ samples a random graph as follows: for any two vertices $u, v \in V$, if they are in the same cluster, then there is an edge between them with probability $\frac{1}{2} + \frac{\delta}{2}$, otherwise there is an edge between them with probability $\frac{1}{2} - \frac{\delta}{2}$. This distribution is a special case of the well-studied stochastic block model (SBM). When δ is not too

small, there exist polynomial time algorithms that recover $V_1, ..., V_k$ from the random graph $G \sim \mathcal{D}(V_1, ..., V_k, \delta)$ with high probability. Here, we will need the following result from (Peng & Zhang, 2021), which instantiates a general theorem from (Vu, 2018).

Lemma 2.3 ((Vu, 2018)). *Let* $\delta \in (0, 1]$, $n = |V|$ *and* $G \sim \mathcal{D}(V_1, ..., V_k, \delta)$. *Suppose the partition* $V_1, ..., V_k$ *is* *b-balanced for some* $b \in (0, 1]$. *Then there exists an algorithm, denoted by* BalSBM(G, k, δ, b), *that recovers all the clusters* $V_1, ..., V_k$ *of* G *in polynomial time with probability at least* $1 - n^{-8}$, *if* $n \geq \frac{c_0 k^2 \log n}{b^2 \delta^2}$ *for some large enough constant* c_0.

Best arm identification. We considered a special case of best arm identification (BAI). In this problem, there are k different arms $a_1, ..., a_k$. When the algorithm pulls a_i, it receives a reward $r \in \{0, 1\}$ from a_i. Assume the rewards from each a_i are i.i.d. Bernoulli(μ_i), where $\mu_i \in [0, 1], \forall i$. The goal is to identify $i^* = \arg \max_i \mu_i$ using minimum number of pulls. We often use sample complexity to denote the total number of pulls conducted by an algorithm. In our application of BAI, the input instance always satisfies the following conditions: $i^* \in [k]$ is unique, $\mu_{i^*} \geq \frac{1}{2} + \frac{\delta}{2}$, and $\mu_i \leq \frac{1}{2} - \frac{\delta}{2}$ for all $i \neq i^*$. BAI is well-studied and the following results will be used, which is a special case of Theorem 10 from (Even-Dar et al., 2006).

Lemma 2.4. *Given* $\delta \in (0, 1)$, $\alpha \in (0, 1)$ *and a* k-BAI *instance defined above, there exists an algorithm* ME(k, δ, α) *that identifies the best arm with probability at least* $1 - \alpha$, *and its sample complexity is* $O(\frac{k}{\delta^2} \log \frac{1}{\alpha})$.

3. Algorithm for Nearly Balanced Instances

In this section, we provide an algorithm for nearly balanced instances (see Definition 2.1), which will be used as a subroutine in the algorithm for general instances. For this easier case, the algorithm is simpler, and the intuition behind using BAI to improve the sample complexity is clearer.

Theorem 3.1. *Let* $b \in (0, 1]$ *and* $n \geq \frac{c_0 k^2 \log^2 n}{b^2 \delta^2}$ *for some constant* c_0. *Suppose* V *is a b-balanced instance. There exists a polynomial time algorithm that recovers all the clusters with success probability* $1 - o_n(1)$. *The total number of queries to the faulty oracle is* $O(\frac{n(k + \log n)}{\delta^2} + \frac{k^4 \log^2 n}{b^4 \delta^4})$.

Our algorithm and the algorithm of (Peng & Zhang, 2021) both consist of two phases, and their first phase is the same.

In the first phase, k sub-clusters is recovered with high probability (see Algorithm 1). In this algorithm, a subset T of vertices is sampled from V uniformly at random, and then query all pairs of vertices in T using $O(|T|^2)$ queries. A graph $H_T = (T, E_T)$ is constructed, where E_T contains all pairs $(u, v) \in T^2$ such that $u \neq v$ and $\tilde{\tau}(u, v) = 1$. Applying standard concentration inequalities, one can show T is

$b/2$-balanced, and H_T satisfies the premises of Lemma 2.3 when $|T| = \Omega(\frac{k^2 \log n}{b^2 \delta^2})$. Thus, the outputs of BalSBM are the true clusters of T, denoted as $T_1, ..., T_k$, where $T_i = T \cap V_i$.

Algorithm 1 SubCluster(V, k, δ, b): sub-clusters recovery

1: Randomly sample $T \subset V$ of size $|T| = \frac{c_0 k^2 \log n}{b^2 \delta^2}$ for large enough constant c_0.
2: Construct H_T using $|T|^2$ queries to the oracle
3: Apply BalSBM$(H_T, k, \delta, b/2)$ to obtain $X_1, ..., X_k$
4: **output** $X_1, ..., X_k$

Lemma 3.2 ((Peng & Zhang, 2021)). *If the input instance* V *is b-balanced, then, with probability* $1 - |T|^{-7}$, *Algorithm 1 correctly outputs the true clusters* $T_1, ..., T_k$ *of* T. *Moreover,* $|T_i| \geq \frac{b|T|}{2k} = \frac{ck \log n}{b\delta^2}$ *for some constant* c *and the sample complexity is* $O(\frac{k^4 \log^2 n}{b^4 \delta^4})$.

In the second phase, each $u \in V \setminus T$ identifies its true label. In (Peng & Zhang, 2021; Mazumdar & Saha, 2017a), the algorithm picks $\Theta(\log n/\delta^2)$ distinct vertices from each T_i, and query u with them. If the majority of these answers are $+1$, then add u to T_i. The correctness of this method can be proved by applying the Chernoff bound together with a union bound. Note the sample complexity is $O(\frac{nk \log n}{\delta^2})$. Although simple, further improving it seems difficult a priori. Next, we show how to cross this barrier using BAI algorithms.

3.1. True Cluster Identification via BAI

After the first phase, we get k disjoint sets $\{X_1, ..., X_k\}$, each of size $\Omega(\frac{k \log n}{b\delta^2})$; and each X_i is a sub-cluster, i.e., $X_i \subset V_i$. Our key observation is that, for any vertex $u \in V$, identifying the cluster of u can be modeled as a best arm identification problem. To see this, each X_i corresponds to the ith arm a_i in BAI. Let $\mathcal{A}$ be an algorithm for k-BAI. When $\mathcal{A}$ pulls an arm a_i, we pick a fresh vertex $v \in X_i$ and query (u, v). Then $\mathcal{A}$ gets a reward 1 if the oracle returns $+1$, and a reward 0 otherwise.

Algorithm 2 TrueClusterId$(u, X_1, ..., X_k, \delta, \alpha)$

1: Simulate ME(k, δ, α):
2: **repeat**
3:     Receive an arm pull request from ME, say a_i
4:     Pick a random new vertex $v \in X_i$, and query (u, v) to get the answer $\tilde{\tau}(u, v)$ from the oracle
5:     Return reward $r(a_i) := \frac{\tilde{\tau}(u,v)+1}{2}$ to ME
6: **until** ME terminates and outputs an arm a_{i^*}
7: **Output** Index i^*

Lemma 3.3. *Assume* X_i *is a sub-cluster and* $|X_i| \geq \Omega(\frac{k}{\delta^2} \log \frac{1}{\alpha})$ *for all* i. *Then for any* $u \in V \setminus T$, *Algorithm 2*

correctly identifies the true cluster of u with probability $1 - \alpha$ and the sample complexity is $O(\frac{k}{\delta^2} \log \frac{1}{\alpha})$.

Proof. We simulate ME on a standard BAI instance, where the best arm corresponds to the true cluster that u belongs to; each arm pull receives an i.i.d. reward with mean $\frac{1}{2} + \frac{\delta}{2}$ for the best arm and with mean $\frac{1}{2} - \frac{\delta}{2}$ for the other arms. The simulation works as long as there are enough number of vertices in each X_i to generate i.i.d. samples, which is satisfied by our assumption. Then the lemma directly follows from Lemma 2.4. □

3.2. Identify the True Clusters of All Vertices

Lemma 3.3 is only for a fixed vertex, and our goal is to identify the true clusters for all vertices simultaneously. An immediate idea is to set $\alpha = 1/n$, so that each of the n vertices find the correct cluster with probability at least $1 - n^{-1}$. Then by a union bound, the goal is achieved with constant probability. However, the total number of queries is $\Theta(\frac{nk \log n}{\delta^2})$, which is the same as in previous work.

Instead, we use a recursive strategy to identify true clusters in multiple rounds. In the first round, we apply Algorithm 2 on each $u \in V$ with $\alpha = 1/4$, and each u gets an index $\tilde{c}(u) \in [k]$. By Lemma 3.3, $\tilde{c}(u)$ indicates the true cluster of u with probability at least $3/4$. By the Chernoff-Hoeffding bound, except for an exponentially (in $|V|$) small probability, at least $1/2$-fraction of V find their correct clusters. Then, we remove them and repeat the same process on the remaining vertices. Since we don't know which nodes have been labelled correctly, we need a verification step to verify whether $\tilde{c}(u)$ is indeed the correct cluster for each u.

For this step, there is only one candidate cluster to verify, so a standard majority test suffices (Algorithm 3). We pick $\Theta(\frac{\log n}{\delta^2})$ fresh vertices from $T_{\tilde{c}(u)}$ and query u with them. If the majority of the answers are 1, then we conclude $\tilde{c}(u)$ is the true cluster and remove u in the next round. Otherwise, u is kept for the next round. By the Chernoff-Hoeffding bound, this verification step is correct with probability $1 - \frac{1}{n^c}$. By union bound, with probability at least $1 - \frac{1}{n^{c-2}}$, all verification steps ever executed are correct, and we assume this event happens.

Algorithm 3 ClusterVerify(v, B)

1: For each $u \in B$, query (v, u) and get answer $\tilde{\tau}(v, u)$
2: Compute $d(v, B) = \sum_{u \in B} \mathbb{1}_{\{\tilde{\tau}(v,u)=1\}}$
3: **if** $d(v, B) \geq \frac{1}{2} |B|$ **then**
4:    Output **TRUE**
5: **else**
6:    Output **FALSE**
7: **end if**

Lemma 3.4. *Let B be a (η, V_i)-biased set of size at least*

$\frac{16 \log n}{\eta^2 \delta^2}$ *for some cluster V_i, then with probability at least $1 - n^{-6}$, the following holds for all $v \in V$ and all clusters V_i simultaneously,*
*(1) if $v \in V_i$, ClusterVerify(v, B) returns **TRUE**;*
*(2) if $v \notin V_i$, ClusterVerify(v, B) returns **FALSE**.*

Proof. Lemma B.1 and union bound for all v and i complete the proof. □

3.3. Query Complexity of the Second Phase

According to the discussion above, the number of remaining vertices after each round decreases by half, i.e., the number of vertices considered in the ith round is at most $\frac{n}{2^{i-1}}$. Since we always set $\alpha = 1/4$, Algorithm 2 incurs $O(\frac{nk}{2^{i-1}\delta^2})$ queries in round i by Lemma 2.4. The number of queries for verification is $O(\frac{n \log n}{2^{i-1}\delta^2})$, and therefore there are $O(\frac{n(k+\log n)}{2^{i-1}\delta^2})$ queries in the ith round. The total number over all rounds is

$$\sum_{i \geq 1} O(\frac{n(k+\log n)}{2^{i-1}\delta^2}) = O(\frac{n(k+\log n)}{\delta^2}).$$

3.4. Put It All Together

The pseudo code of our algorithm for balanced instances is presented in Algorithm 4.

Algorithm 4 BalNoisyClustering(V, k, δ, b)

1: Let $\alpha = 1/4$. Initialize $U = V$
2: Invoke SubCluster(V, k, δ, b) to obtain sub-clusters $X_1, ..., X_k$
3: For each $u \in V$, set $D_u = \{X_1, ..., X_k\}$ {Candidate clusters of u}
4: For each X_i, pick arbitrary $X_i' \subseteq X_i$ of size $\frac{1600 \log n}{\delta^2}$
5: **while** $|U| \geq \frac{c_0 k^2 \log n}{b^2 \delta^2}$ **do**
6:    **for** each $v \in U$ **do**
7:       $j = $ TrueClusterId(v, D_v, δ, α)
8:       $o = $ ClusterVerify(v, X_j')
9:       **if** $o = $**TRUE** **then**
10:          Update $X_j = X_j \cup \{v\}, U = U \setminus \{v\}$
11:       **else**
12:          Update $D_v = D_v \setminus \{X_j\}$
13:       **end if**
14:    **end for**
15: **end while**
16: **for** each $v \in U$ **do**
17:    **for** each $X_j \in D_v$ **do**
18:       **if** ClusterVerify$(v, X_j') = $ **TRUE then**
19:          Update $X_j = X_j \cup v$
20:       **end if**
21:    **end for**
22: **end for**
output k clusters $X_1, ..., X_k$

Now we are ready to prove the main theorem for the nearly balanced instances, i.e. Theorem 3.1.

Proof of Theorem 3.1. First, it holds with probability $1 - o_n(1)$ that $X_1, ..., X_k$ are always sub-clusters (i.e., $X_i \subset V_i$) during the execution of Algorithm 4. To see this, with probability $1 - o_n(1)$, it is true after invoking SubCluster in line 2 (Lemma 3.2). Conditioned on this, $X'_i \subset V_i$ for each i (X'_i is $(1/2, V_i)$-biased). Each time we add a vertex v to X_i, it holds that $\mathsf{ClusterVerify}(v, X'_j) = \mathbf{TRUE}$, which means by Lemma 3.4, with probability $1 - \frac{1}{n^6}$, $v \in V_i$ and after adding v, X_i is still a sub-cluster of V_i. The claim follows by applying a union bound over all updates to X_i. In the cleanup stage (line 16-22 in Algorithm 4), all remaining vertices will eventually find their correct clusters with probability at least $1 - \frac{1}{n^6}$ (again by Lemma 3.4), therefore the correctness of the algorithm is proved.

For the query complexity, in the first phase SubCluster incurs $O(\frac{k^4 \log^2 n}{b^4 \delta^4})$ queries (Lemma 3.2). The query complexity of the second phase is analyzed in subsection 3.3, which is $O(\frac{n(k+\log n)}{\delta^2})$. In the cleanup stage, each of the remaining vertices invokes ClusterVerify less than k times, and thus the total number of queries is $O(\frac{k^3 \log^2 n}{b^2 \delta^4})$. To sum up the query complexiy of Algorithm 4 is at most

$$O\left(\frac{n(k+\log n)}{\delta^2} + \frac{k^4 \log^2 n}{b^4 \delta^4}\right).$$

And the running time of our algorithm is polynomial in $n, k, 1/\delta$. $\square$

4. Algorithm for General Instances

For general instances, let $s_1, ..., s_k$ be the sizes of $V_1, ..., V_k$ respectively. W.l.o.g, we assume $s_1 \geq s_2 \cdots \geq s_k$. Similar as for balanced instances, we still randomly sample a subset T of size $\mathrm{poly}(k, 1/\delta, \log n)$ and query all pairs of vertices in T. As before, let $H_T = (T, E_T)$ be the graph with positive answers and define $T_i = T \cap V_i$. Now T might be extremely unbalanced and we cannot apply Lemma 2.3.

The idea of Peng & Zhang (2021) is to first prune all vertices in T that belong to small clusters. Let T' be the remaining vertices, which is then a balanced instance. Peng & Zhang (2021) observe that if u belongs to a small cluster, then its degree in H_T is small, and vice versa. The next lemma provides a formal statement of this intuition, which can be proved by the Chernoff-Hoeffding bound.

Lemma 4.1 ((Peng & Zhang, 2021)). *Let $b \in [0, 1/2], h \in [k]$, and $d_h = \frac{t}{2} - \left(\frac{1}{2} - \frac{1}{k} + \frac{(h+1/2)b}{k^2}\right)\delta t$. Assume $t = |T| = \frac{64 k^4 \log n}{b^2 \delta^2}$ and $u \in T_i$, then it holds with probability $1 - \frac{1}{n^7}$: (1) If $|V_i| \leq \frac{n}{k} - \frac{b(h+1)n}{k^2}$, then the degree of u in H_T is smaller than d_h; (2) If $|V_i| \geq \frac{n}{k} - \frac{bhn}{k^2}$, then the degree of u in H_T is larger than d_h.*

A natural idea is to remove all vertices in T with degree smaller than d_h and let T' be the remaining vertices. We are guaranteed that, if $|V_i| \leq \frac{n}{k} - \frac{b(h+1)n}{k^2}$ then $V_i \cap T' = \emptyset$; and if $|V_i| \geq \frac{n}{k} - \frac{bhn}{k^2}$, then $T_i \subset T'$. However, T' can still be unbalanced, since for V_i with size in between, $|V_i \cap T'|$ could be some positive but small integer. Peng & Zhang (2021) then prove that if $s_k < bn/k$ (the input is unbalanced), then there must exists $h < k$ such that $s_{h+1} < \frac{n}{k} - \frac{b(h+1)n}{k^2}$ and $s_h \geq \frac{n}{k} - \frac{bhn}{k^2}$. Then, after pruning vertices with degree smaller than d_h,[1] T' only contains vertices from $V_1, ..., V_h$ and is $\frac{h}{2k}$-balanced, whose clusters can be recovered by applying BalSBM (Lemma 2.3) and let $X_1, ..., X_h$ be the clusters of T'.

Next, we grow $X_1, ..., X_h$ to $V_1, ..., V_h$ by identifying the true cluster of each $u \in V \setminus T'$ as before. In (Peng & Zhang, 2021), this is achieved by querying u with $\Theta(\frac{\log n}{\delta^2})$ vertices from each X_i and checking whether the majority is positive. After getting $V_1, ..., V_h$, the remaining clusters are solved recursively. Again the query complexity is suboptimal and we wish to improve it using BAI.

4.1. A New Size-gap Lemma

Similar as in the balanced case, for each $u \in V \setminus T'$, we invoke $\mathsf{TrueClusterId}(u, X_1, ..., X_h, \delta, 1/4)$ (Algorithm 2) to identify the cluster label of u, and then invoke ClusterVerify (Algorithm 3) to verify the cluster. However, the same argument as for the balanced case does not work any more. Let n_i be the number of remaining vertices in the beginning of the ith round. In the balanced case, at least half of the n_i vertices will be settled in the ith round. For the unbalanced case, this is not true when $h < k$, since it is possible that the majority of the n_i vertices belongs to $V_{h+1}, ..., V_k$; such vertices will never be settled in this round.

To overcome the above obstacle, we first strengthen the size-gap lemma of (Peng & Zhang, 2021). We show that for any unbalanced instance, there exists $h < k$ such that there is a gap between s_h and s_{h+1}. Moreover, the number of vertices in $V_1, ..., V_h$ is $\Theta(n)$.

Lemma 4.2. *If $s_k < \frac{n}{4k}$, then there exists $h < k$ such that*

$$s_h \geq \frac{n}{2k} - h \cdot \frac{n}{4k^2} \text{ and } s_{h+1} < \frac{n}{2k} - (h+1) \cdot \frac{n}{4k^2}.$$

Moreover, $\sum_{i \leq h} s_i \geq n/2$.

Note this is still not sufficient. For example, it is possible that $h = \Theta(k)$ and $\sum_{i > h} s_i = \Theta(n)$; for this case, each TrueClusterId incurs $\Theta(h/\delta^2) = \Theta(k/\delta^2)$ queries and in each round TrueClusterId is executed on $\Theta(n)$ vertices, and thus $\Theta(nk/\delta^2)$ queries. Note it needs $\Omega(\log n)$ rounds to ensure all vertices in $V_1, ..., V_h$ have been identified, which

---

[1]The only problem is that such h exists but is unknown, which will be discussed later.

is required for the success of the algorithm from (Peng & Zhang, 2021). We take a different strategy. Given sub-clusters, we do not try to recover all vertices belongs to those clusters, but stop after one round of filtering. By the new size-gap lemma, this removes a constant fraction of vertices. Then we apply the same procedure on the rest of vertices recursively. Details are provided in Section 4.4.

4.2. Recovering Sub-clusters

Based on the idea outlined above, we describe a subroutine, which outputs h sub-clusters assuming h is given. We first randomly sample a set $T \subset V$ of size $t = \frac{ck^4 \log n}{\delta^2}$, and then query all pairs $u, v \in T$ and construct the graph $H_T = (T, E_T)$. Next, we remove small-degree vertices and apply BalSBM (Lemma 2.3) on the remaining graph (see Algorithm 5).

Algorithm 5 GapSBM(T, h, δ)

1: Remove all vertices in H_T with degree less than $d_h :=$ $(\frac{1}{2} - \frac{\delta}{2})t + \delta \cdot (\frac{1}{2k} - \frac{h+1/2}{4k^2})t$
2: Let T'_h be the set of remaining vertices and $H_{T'_h}$ be the graph induced by T'_h
3: Invoke BalSBM$(H_{T'_h}, h, \delta, \frac{h}{4k})$ to obtain clusters $X_1, ..., X_h$
4: **output** $X_1, ..., X_h$

The algorithm is essentially the same as in (Peng & Zhang, 2021), however, equipped with our new size-gap lemma, the guarantee is stronger.

Lemma 4.3. *Let T be a random subset of V of size t. With probability $1 - O(k^{-24} \log^{-8} n)$, we have*
(1) If h is the index from Lemma 4.2. Then, the output of GapSBM are h sub-clusters of $V_1, ..., V_h$ whose sizes satisfy $s_i \geq t/4k$ and $\sum_{i \leq h} s_i \geq n/2$.
(2) For an arbitrary $h \in [k]$, all vertices in T that belongs to V_i with $s_i \geq \frac{n}{2k} - h \cdot \frac{n}{4k^2}$ must be contained in the output.

The proof follows the analysis of (Peng & Zhang, 2021). For completeness, we include a proof in Appendix C.

Before giving the full algorithm, there is another issue to deal with: h from Lemma 4.2 is unknown. We simply apply GapSBM(T, h, δ) for all $h \in [k]$ in decreasing order, and then perform a test on the output to check whether all subsets returned are indeed sub-clusters. This idea was successfully implemented in (Peng & Zhang, 2021). Our algorithm GapSubcluster (Algorithm 9) follows the same idea and the proof of the following lemma is given in Appendix C. Our proof is a non-trivial extension due to part (c).

Lemma 4.4. *Let T be a random subset of V of size $t = \frac{ck^4 \log n}{\delta^2}$. GapSubcluster$(T, k, \delta)$ outputs $X_1, ..., X_h$ for some $h \in [k]$ or **Fail**. If $s_k < |V|/4k$, then with proba-*

bility $1 - O(k^{-24} \log^{-8} n)$, it does not output **Fail**. If the algorithm does not **fail** then

(a) $|X_i| \geq \frac{t}{4k}$.

(b) X_i is $(0.1, V_{g(i)})$-biased for some $g(i) \in [k]$.

(c) $\sum_{j:g^{-1}(j) \neq \emptyset} |V_j| \geq n/5$.

The query complexity is $O(t^2)$.

4.3. True Cluster Identification: One Round

By Lemma 4.4, GapSubcluster outputs $X_1, ..., X_h$ such that X_i is $(0.1, C)$-biased for some cluster C in $V_1, ..., V_k$. Similar as in Section 3.1, we use TrueClusterId$(u, X_1, ..., X_h, \delta, 1/4)$ to identify the cluster of u. Here the situation is slightly different, since now each X_i is not a strict sub-cluster. But the same algorithm still works, except that we need to use a smaller δ, denoted as δ' to be determined later. To simulate an arm a_i for $i \in [h]$, we randomly sample a vertex v from X_i and query (u, v), and the reward is $r(a_i) = \frac{\tilde{\tau}(u,v)+1}{2}$. Suppose $u \in V_{g(i)}$, the expected reward of pulling a_i is at least

$$(\frac{1}{2} + 0.1)(\frac{1}{2} + \frac{\delta}{2}) + (\frac{1}{2} - 0.1)(\frac{1}{2} - \frac{\delta}{2}) = \frac{1}{2} + \frac{\delta}{10}.$$

Similarly, if $u \notin V_{g(i)}$, the reward is at most $\frac{1}{2} - \frac{\delta}{10}$. So, with probability $3/4$, the true cluster of u is identified by ME$(h, \frac{\delta}{5}, 1/4)$, and the query complexity of which is the same up to a constant. One subtlety is that the random vertex v may have been sampled before, and the resulting rewards may not satisfy i.i.d. condition. Note, if X_i is a strict sub-cluster, we could just pick any v that has not been queried before. However, the correctness of BAI on such a reward distribution still holds up to a constant blowup in the query complexity by a more careful concentration argument.

Lemma 4.5. *Let $X_1, ..., X_h$ be the subsets obtained in Lemma 4.4. if u belongs to $V_{g(i^*)}$ for some i^*, then TrueClusterId$(u, X_1, ..., X_h, \delta/5, \alpha)$ returns i^* with probability $1 - \alpha$.*

Similarly, we use ClusterVerify to verify whether the cluster label for each u is correct. By (c) of Lemma 4.4, at least a constant fraction of V will pass this verification and are added to the correct clusters. The query complexity of this part is still $O(\frac{|V|k}{\delta^2} + \frac{|V| \log |V|}{\delta^2})$.

4.4. Iterative Clustering

The above true cluster identification step successfully labels a constant fraction of vertices. Let U be the set of vertices remain unlabelled. To make U a valid noisy clustering instance, we also exclude T from U, since all pairs from T have been queried. Then, we apply the same procedure on U. After at most $O(\log n)$ rounds, the size of U becomes too small, and then we do a cleanup step.

Merging clusters of different rounds. Note that each round may produce as many as k sub-clusters and there are totally $O(k \log n)$ over all rounds, denoted as $C_i, ..., C_\ell$. So we need to merge sub-clusters from the same V_i. This is another noisy clustering problem, but easier. Given two sub-clusters C_i and C_j, to test whether both belong to the same cluster, we just pick one vertex u from C_i and $m = O(\frac{\log n}{\delta^2})$ vertices $v_1, ..., v_m$ from C_j, and query (u, v_i) for all $i \in [m]$. If the majority of the answer is positive, then C_i, C_j are merged. We do this testing for all $k^2 \log^2 n$ pairs of clusters. By the Chernoff-Hoeffding bound and union bound, with probability at least $1 - n^{-6}$, the merging result is correct. The sample complexity of the merging process is at most $O(\frac{k^2 \log^3 n}{\delta^2})$.

4.5. The Final Algorithm

Algorithm 6 NoisyClustering(V, k, δ)

1: Let c be a sufficiently large constant. Initialize $R = \emptyset$, $C_i = \emptyset$ for $i \in [k]$, $r = 1$, $U = V$.
2: **while** $|U| \geq \frac{ck^4 \log n}{\delta^2}$ **do**
3:     Randomly sample $T^r \subset U$ of size $|T^r| = \frac{ck^4 \log n}{\delta^2}$
4:     Inkove GapSubcluster(T^r, k, δ) to obtain sub-clusters $X_1, ..., X_h$.
5:     **if** GapSubcluster output **Fail then**
6:         Invoke BalNoisyClustering$(U, k, \delta, 1/4)$ to get k clusters $X_1, ..., X_k$
7:         $U = \emptyset$
8:         Merge $X_1, ..., X_k$ into C_i's as described in 4.4
9:         **break**
10:     **end if**
11:     For each i, pick a subset $X_i' \subseteq X_i$ of size $\frac{c \log n}{\delta^2}$
12:     $U = U \setminus T^r$, $R = R \cup T^r$, $Y_i = \emptyset$
13:     **for** each $v \in U$ **do**
14:         $j =$ TrueClusterId$(v, X_1, ..., X_h, \delta, 1/8)$
15:         $o =$ ClusterVerify(v, X_j')
16:         **if** $o =$ **TRUE then**
17:             Update $Y_j = Y_j \cup \{v\}$, $U = U \setminus \{v\}$
18:         **end if**
19:     **end for**
20:     Merge $Y_1, ..., Y_k$ into C_i's as described in 4.4
21:     $r = r + 1$
22: **end while**
23: {Cleanup stage}
24: **for** each C_i **do**
25:     Pick a subset $C_i' \subseteq C_i$ of size $\frac{16 \log n}{\delta^2}$
26:     Let $\tilde{C}_i = \{v \in U \cup R : $ ClusterVerify$(v, C_i') = $ **TRUE**$\}$
27:     Update $C_i = C_i \cup \tilde{C}_i$
28: **end for**
output all the clusters C_i's

Now the outline of our algorithm for general case is com-plete, and we present the pseudo code of the final algorithm in Algorithm 6. The proof of the main theorem, i.e. Theorem 1.1, is provided in Appendix C.4.

5. Lower bound

In (Mazumdar & Saha, 2017a), they proved a $\Omega(\frac{nk}{\delta^2})$ lower bound on the number of queries. Here we prove that $\Omega(\frac{n \log n}{\delta^2})$ is also a lower bound and thus our algorithm is optimal.

Best arm identification with failure. Before proving this lower bound, we define a variant of the best 2-arm identification problem. We call this problem *best arm identification with failure*, denoted as (δ, α)-BAIF. Given 2 arms of expected reward $\frac{1}{2} + \frac{\delta}{2}$ and $\frac{1}{2} - \frac{\delta}{2}$ respectively for some $\delta \in (0, 1)$, suppose the rewards are binary i.e. $\{0, 1\}$. We still want to find the optimal arm, but now we allow the algorithm to fail with small probability. More specifically, the algorithm can return FAIL (not any one of the 2 arms) with probability at most $1/8$, or return an arm otherwise (NOT FAIL). When the algorithm does return an arm, then it should be the optimal one with probability at least $1 - \alpha$ (conditioned on NOT FAIL).

We claim that any algorithm that solves (δ, α)-BAIF, has a sample complexity lower bound $\Omega(\frac{1}{\delta^2} \log \frac{1}{\alpha})$, which is the same as the well known lower bound for the original best arm identification problem (Mannor & Tsitsiklis, 2004). The proof of Lemma 5.1 is in Appendix D.

Lemma 5.1. *Any (randomized) algorithm that solves (δ, α)-BAIF has $\Omega(\frac{1}{\delta^2} \log \frac{1}{\alpha})$ sample complexity.*

Reduction from BAIF to clustering with noisy queries. Now we show that BAIF can be reduced to the problem of identifying the cluster of one vertex in the faulty oracle model with $k = 2$ clusters. Suppose $\mathcal{A}$ is an algorithm for clustering with noisy queries with success probability at least $\frac{7}{8}$, then we can derive an algorithm $\mathcal{A}_i'$ indexed by a vertex i that solves $(\delta, 1/n)$-BAIF. The query complexity of $\mathcal{A}_i'$ is the number of queries in $\mathcal{A}$ that involve vertex i. Lemma 5.1 implies that $\mathcal{A}$ must query i at least $\Omega(\frac{1}{\delta^2} \log n)$ times. So if we can show that there are $\Omega(n)$ such i's, the query complexity of $\mathcal{A}$ is $\Omega(\frac{n}{\delta^2} \log n)$.

Consider a graph distribution $\mathcal{D}$ of n vertices, which samples a random graph as follows. First let each vertex belongs to one of the two clusters C_0, C_1 randomly with equal probability. Then fix the clusters and sample a SBM-type graph, i.e. assign an edge to each pair of vertices from the same cluster with probability $\frac{1}{2} + \frac{\delta}{2}$, and assign an edge to each pair of vertices from different clusters with probability $\frac{1}{2} - \frac{\delta}{2}$.

Let $\mathcal{A}$ be any algorithm that solves clustering with noisy

queries with probability at least $\frac{7}{8}$. We next provide an algorithm for $(\delta, 1/n)$-BAIF (Algorithm 7).

Algorithm 7 $\mathcal{A}'_s$: algorithm for BAIF

Require: two arms a_0, a_1, suppose a_{opt} is the best arm
1: Assign vertices $v_1, ..., v_{s-1}, v_{s+1}, ..., v_n$ to the cluster C_0 or C_1 randomly with equal probability, let $c(i)$ be the cluster of v_i.
2: For the special vertex v_s, its cluster is indicated by the (unknown) index of the best arm, i.e. opt.
   ## Simulate $\mathcal{A}$ on the instance constructed above:
3: **repeat**
4:    Receive an query request from $\mathcal{A}$, say $(v_i, v_{i'})$
5:    **if** $i, i' \neq s$ **then**
6:       **if** $v_i, v_{i'}$ are in the same cluster **then**
7:          $\beta \sim \text{Bernoulli}(\frac{1}{2} + \frac{\delta}{2})$
8:       **else**
9:          $\beta \sim \text{Bernoulli}(\frac{1}{2} - \frac{\delta}{2})$
10:      **end if**
11:      answer $\tilde{\tau}(v_i, v_{i'}) = 2\beta - 1$
12:   **else if** $i = s$ (or $i' = s$) **then**
13:      pull arm $a_{c(i')}$ (or $a_{c(i)}$) get a reward $r \in \{0, 1\}$
14:      answer $\tilde{\tau}(v_i, v_{i'}) = 2r - 1$
15:   **end if**
16: **until** $\mathcal{A}$ terminates and outputs the cluster labels $\tilde{c}(1), ..., \tilde{c}(n)$
17: **if** $c(i) = \tilde{c}(i)$ for $i = 1, ..., s - 1$ **then**
18:    output $\tilde{c}(s)$
19: **else**
20:    output FAIL
21: **end if**

Let $\mathcal{E}$ denote the event that $\mathcal{A}$ correctly recovers all the clusters for an instance sampled from $\mathcal{D}$, and $\mathcal{E}_i, i \in [n]$ denote the event that the algorithm correctly identifies the cluster for vertex v_i, i.e., $\tilde{c}(i) = c(i)$. Note $\text{Pr}_{\mathcal{D},\mathcal{A}}[\mathcal{E}] \geq 7/8$. To simplify notation, we omit the subscripts, but the probability is always over both $\mathcal{D}$ and the randomness of $\mathcal{A}$.

Definition 5.2. We call index $s \in [n]$ good, if the conditional probability $\text{Pr}[\mathcal{E}_s | \mathcal{E}_1, ..., \mathcal{E}_{s-1}] \geq 1 - 1/n$.

Lemma 5.3. *There are at least $3n/4$ good indices.*

Proof. Suppose there are less than $3n/4$ good indices, then, by the chain rule,

$$\text{Pr}[\mathcal{E}] = \prod_{i=1}^{n} \text{Pr}[\mathcal{E}_i | \mathcal{E}_1, ..., \mathcal{E}_{i-1}] < (1 - \frac{1}{n})^{\frac{n}{4}} \leq e^{-\frac{1}{4}} < \frac{7}{8},$$

which is a contradiction to the correctness of $\mathcal{A}$. So there must be at least $3n/4$ good indices in $[n]$. $\square$

Lemma 5.4. *Suppose s is good, then $\mathcal{A}'_s$ solves $(\delta, 1/n)$-BAIF. The sample complexity of $\mathcal{A}'_s$ is the same as the number of queries used in $\mathcal{A}$ that involve s.*

Proof. Let us assume a_0 and a_1 both have probability $1/2$ being the better arm. This is not a restriction, since we can always randomly permute the two arms in the beginning. Then, the clustering instance for $\mathcal{A}$ constructed in $\mathcal{A}'_s$ has the same distribution as $\mathcal{D}$, since each v_i, including v_s, belongs to C_0 or C_1 with equal probability and edges are also sampled accordingly. We observe that the cluster label of v_s in the graph is the same as the index of the best arm.

The event that $\mathcal{A}'_s$ outputs FAIL implies that $\mathcal{A}$ is wrong for at least one vertex, which happens with probability at most $1/8$. On the other hand, if $\mathcal{A}'_s$ does not output FAIL, then $\mathcal{E}_1, ..., \mathcal{E}_{i-1}$ all happens. By the assumption that s is good, i.e., $\text{Pr}[\mathcal{E}_s | \mathcal{E}_1, ..., \mathcal{E}_{s-1}] \geq 1 - 1/n$, we conclude that $\tilde{c}(s)$ is the correct cluster label of v_s, and hence the index of the best arm, with probability at least $1 - 1/n$.

From the description of Algorithm 7, each arm pull in $\mathcal{A}'_s$ corresponds to a query of $\mathcal{A}$ that involves s, which finishes the proof the lemma. $\square$

Now we are ready to prove the main theorem.

Proof of Theorem 1.2. We have shown that, given any algorithm for the noisy clustering problem, we can construct n algorithms $\mathcal{A}'_s$, $s = 1, ..., n$, for BAIF. Let S be the sample complexity of $\mathcal{A}$, and S_i be the number of queries that involve i. We have

$$\mathbb{E}_{\mathcal{D},\mathcal{A}}[S] = \frac{1}{2} \sum_{i=1}^{n} \mathbb{E}_{\mathcal{D},\mathcal{A}}[S_i] \geq \frac{1}{2} \sum_{i:i \text{ is good}} \mathbb{E}_{\mathcal{D},\mathcal{A}}[S_i]. \quad (1)$$

For a good i, by Lemma 5.4, the algorithm $\mathcal{A}'_s$ solves $(\delta, 1/n)$-BAIF and $\mathbb{E}_{\mathcal{D},\mathcal{A}}[S_i]$ is the expected query complexity of $\mathcal{A}'_s$. Then, by Lemma 5.1, $\mathbb{E}_{\mathcal{D},\mathcal{A}}[S_i] = \Omega(\frac{1}{\delta^2} \log n)$. Since there are at least $3n/4$ good indices (Lemma 5.3), then by (1), the expected number of query complexity of $\mathcal{A}$ is $\mathbb{E}_{\mathcal{D},\mathcal{A}}[S] = \Omega(\frac{n \log n}{\delta^2})$. $\square$

6. Conclusion

In this paper, we study clustering with a faulty oracle, a.k.a. clustering with noisy queries, where the algorithm is allowed to ask a noisy oracle whether two items belong to the same cluster or not. We provide a new algorithm with improved query complexity and also prove a new information-theoretic lower bound that matches our upper bound for a wide range of parameters. Along the way, we introduce several new techniques. For the upper bound, the most important new ingredient is to utilize multi-armed bandit algorithms to identify true clusters of vertices. Our lower bound proof also relies on a new connection to the best arm identification with failure problem. We believe our new techniques and insights are of independent interests and might be useful for other problems in noisy query models.

Acknowledgements

References

Abbe, E. Community detection and stochastic block models: Recent developments. *Journal of Machine Learning Research*, 18(177):1–86, 2018.

Ailon, N., Bhattacharya, A., and Jaiswal, R. Approximate correlation clustering using same-cluster queries. In *Latin American Symposium on Theoretical Informatics*, pp. 14–27. Springer, 2018.

Arthur, D. and Vassilvitskii, S. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035. Society for Industrial and Applied Mathematics, 2007.

Ashtiani, H., Kushagra, S., and Ben-David, S. Clustering with same-cluster queries. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, volume 29, pp. 3216–3224. Curran Associates, Inc., 2016.

Brandes, U., Gaertler, M., and Wagner, D. Experiments on graph clustering algorithms. In *Proceedings of the 11th European Symposium on Algorithms*, pp. 568–579. Springer, 2003.

Bressan, M., Cesa-Bianchi, N., Lattanzi, S., and Paudice, A. Exact recovery of mangled clusters with same-cluster queries. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, volume 33, pp. 9324–9334. Curran Associates, Inc., 2020.

Chien, I. E., Pan, C., and Milenkovic, O. Query k-means clustering and the double dixie cup problem. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 6650–6659, Red Hook, NY, USA, 2018. Curran Associates Inc.

Dalvi, N., Dasgupta, A., Kumar, R., and Rastogi, V. Aggregating crowdsourced binary ratings. In *Proceedings of the 22nd International Conference on World Wide Web*, pp. 285–294, New York, NY, USA, 2013. Association for Computing Machinery.

Durrett, R. *Probability: Theory and Examples*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 5 edition, 2019.

Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

Even-Dar, E., Mannor, S., and Mansour, Y. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 7(39):1079–1105, 2006.

Fellegi, I. P. and Sunter, A. B. A theory for record linkage. *Journal of the American Statistical Association*, 64(328): 1183–1210, 1969.

Getoor, L. and Machanavajjhala, A. Entity resolution: Theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5(12):2018–2019, 2012.

Karger, D. R., Oh, S., and Shah, D. Iterative learning for reliable crowdsourcing systems. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, pp. 1953–1961, Red Hook, NY, USA, 2011. Curran Associates Inc.

Karypis, G. and Kumar, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

Larsen, K. G., Mitzenmacher, M., and Tsourakakis, C. Clustering with a faulty oracle. In *Proceedings of The Web Conference 2020*, pp. 2831–2834. Association for Computing Machinery, 2020.

Li, Y., Song, Y., and Zhang, Q. Learning to cluster via same-cluster queries. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 978–987, New York, NY, USA, 2021. Association for Computing Machinery.

Mannor, S. and Tsitsiklis, J. N. The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research*, 5:623–648, 2004.

Mazumdar, A. and Saha, B. Clustering with noisy queries. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5790–5801, Red Hook, NY, USA, 2017a. Curran Associates Inc.

Mazumdar, A. and Saha, B. A theoretical analysis of first heuristics of crowdsourced entity resolution. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 970–976. AAAI Press, 2017b.

McLachlan, G. J. and Basford, K. E. *Mixture Models: Inference and Applications to Clustering*. Statistics, textbooks and monographs. M. Dekker, 1988.

McSherry, F. Spectral partitioning of random graphs. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pp. 529–537. IEEE Computer Society, 2001.

Ng, A. Y., Jordan, M. I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 15th International Conference on Neural Information Processing Systems: Natural and Synthetic*, pp. 849–856, Cambridge, MA, USA, 2001. MIT Press.

Peng, P. and Zhang, J. Towards a query-optimal and time-efficient algorithm for clustering with a faulty oracle. In Belkin, M. and Kpotufe, S. (eds.), *Proceedings of Thirty Fourth Conference on Learning Theory*, volume 134 of *Proceedings of Machine Learning Research*, pp. 3662–3680. PMLR, 2021.

Prelec, D., Seung, H. S., and McCoy, J. A solution to the single-question crowd wisdom problem. *Nature*, 541 (7638):532–535, 2017.

Saha, B. and Subramanian, S. Correlation Clustering with Same-Cluster Queries Bounded by Optimal Cost. In *Proceedings of the 27th Annual European Symposium on Algorithms (ESA 2019)*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 81:1–81:17, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Vesdapunt, N., Bellare, K., and Dalvi, N. Crowdsourcing algorithms for entity resolution. *Proceedings of the VLDB Endowment*, 7(12):1071–1082, 2014.

Vu, V. A simple svd algorithm for finding hidden partitions. *Combinatorics, Probability and Computing*, 27(1): 124–140, 2018.

Wainwright, M. J. *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019.

Wang, J., Kraska, T., Franklin, M. J., and Feng, J. Crowder: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11):1483–1494, 2012.

A. Preliminaries

We will use the following well known inequalities. See, e.g., Proposition 2.5 in (Wainwright, 2019) and Theorem 2.5.5 in (Durrett, 2019).

Theorem A.1 (Chernoff-Hoeffding bound). *Let $n \geq 1$. Suppose there are n independent random variables $X_1, ..., X_n$ that are distributed in $[0, 1]$. Let $X := \sum_{i=1}^{n}$. Then for all $\lambda > 0$,*

$$\mathbb{P}\left(X > \mathbb{E}[X] + \lambda\right), \mathbb{P}\left(X < \mathbb{E}[X] - \lambda\right) \leq e^{-2\lambda^2/n}.$$

Theorem A.2 (Kolmogorov's inequality). *Let $n \geq 1$. Suppose there are n independent random variables $X_1, ..., X_n$ such that $\mathbb{E}[X_i] = 0$ and $\mathrm{Var}(X_i) < \infty$. Let $S_k = \sum_{i=1}^{k} X_i, k = 1, ..., n$. Then for all $\lambda > 0$,*

$$\mathbb{P}\left(\max_{1 \leq k \leq n} |S_k| \geq \lambda\right) \leq \lambda^{-2} \mathrm{Var}(S_n).$$

B. Deferred Proofs from Section 3

Define function $d : V \times 2^V$,

$$d(v, X) = \sum_{u \in X} \mathbb{1}_{\{\tilde{\tau}(v,u)=+1\}}$$

to be the number of positive edges between v and vertices in set X, where $\mathbb{1}$ is the indicator function. This quantity has the following property.

Lemma B.1. *Let B be a (η, C)-biased set of size at least $\frac{16 \log n}{\eta^2 \delta^2}$, for some cluster C. Then with probability at least $1 - n^{-8}$, for any vertex $v \in V$,*

$$d(v, B) \geq \left(\frac{1}{2} + \frac{1}{2}\eta\delta\right)|B|, \ \text{if } v \in B,$$

$$d(v, B) \leq \left(\frac{1}{2} - \frac{1}{2}\eta\delta\right)|B|, \ \text{if } v \notin B.$$

Proof. Let C_v denote the true cluster v belongs to. Let $B_v = B \cap C_v$.

$$\mathbb{E}[d(v, B)] = \left(\frac{1}{2} + \frac{1}{2}\delta\right)|B_v| + \left(\frac{1}{2} - \frac{1}{2}\delta\right)|B \setminus B_v|$$

$$= \left(\frac{1}{2} - \frac{1}{2}\delta\right)|B| + \delta|B_v|.$$

Let $\lambda = \frac{1}{2}\eta\delta|B|$, then $\lambda^2/|B| \, 4\log n$. Since B is (η, C)-biased, then if $v \in C$, $|B_v| \geq (\frac{1}{2} + \delta)|B|$, so we have

$$\mathbb{E}[d(v, B)] \geq \left(\frac{1}{2} + \eta\delta\right)|B|.$$

By Chernoff-Hoeffding bound (see Lemma A.1), with probability at least $1 - \exp{-2\lambda^2/|B|} \geq 1 - n^{-8}$,

$$d(v, B) \geq \left(\frac{1}{2} + \frac{1}{2}\eta\delta\right)|B|.$$

Similarly if $v \notin C$, then $|B_v| \leq (\frac{1}{2} + \delta)|B|$, so with probability at least $1 - \exp{-2\lambda^2/|B|} \geq 1 - n^{-8}$,

$$d(v, B) \leq \left(\frac{1}{2} - \frac{1}{2}\eta\delta\right)|B|. \qquad \square$$

C. Deferred proofs from Section 4

C.1. Proof of Lemma 4.2

Proof. Let $f(i) = s_i - \left(\frac{n}{2k} - i \cdot \frac{n}{4k^2}\right)$, then

$$f(1) = s_1 - \left(\frac{n}{2k} - \frac{n}{4k^2}\right) \geq \frac{n}{k} - \frac{n}{2k} + \frac{n}{4k^2} > 0,$$
$$f(k) = s_k - \left(\frac{n}{2k} - k \cdot \frac{n}{4k^2}\right) < \frac{n}{4k} - \frac{n}{4k} \leq 0.$$

Let h be the largest index such that $f(h) \geq 0$. By the above two inequalities, $h \geq 1$ and $h \leq k - 1$. Then $s_h \geq \frac{n}{2k} - h \cdot \frac{n}{4k^2}$ and $s_{h+1} < \frac{n}{2k} - (h+1) \cdot \frac{n}{4k^2}$. Since $s_i \leq \frac{n}{2k}$, $\sum_{i>h} \leq n/2$, and thus the second part is proved. □

C.2. Proof of Lemma 4.3

Proof. Let $\lambda_1 = \frac{t}{16k^2}$, then $\lambda_1^2/t \geq 4\log n$. For any $i \leq h$, by Chernoff-Hoeffding bound, with probability at least $1 - n^{-8}$,

$$|T_i| \geq \left(\frac{1}{2k} - \frac{h+1/4}{4k^2}\right) t,$$

For any $i > h$, with probability at least $1 - n^{-8}$,

$$|T_i| < \left(\frac{1}{2k} - \frac{h+3/4}{4k^2}\right) t.$$

In the following, we assume the above condition holds for all $i \in [k]$, which occurs with probability at least $1 - n^{-7}$ by the union bound.

Let $\lambda_2 = \frac{t\delta}{16k^2}$, then $\lambda_2^2/t \geq 4\log n$. For any $i \leq h$, for any vertex $v \in T_i$, by Chernoff-Hoeffding bound, with probability at least $1 - n^{-8}$, its degree in H_T

$$d(v, T) \geq \left(\frac{1}{2} - \frac{\delta}{2}\right) t + \delta \cdot \left(\frac{1}{2k} - \frac{h+1/2}{4k^2}\right) t.$$

For any $i > h$, for any vertex $v \in T_i$, with probability at least $1 - n^{-8}$, its degree in H_T

$$d(v, T) < \left(\frac{1}{2} - \frac{\delta}{2}\right) t + \delta \cdot \left(\frac{1}{2k} - \frac{h+1/2}{4k^2}\right) t.$$

Let $d_h = (\frac{1}{2} - \frac{\delta}{2})t + \delta \cdot (\frac{1}{2k} - \frac{h+1/2}{4k^2})t$, then by union bound, with probability at least $1 - n^{-7}$, all vertices in $T_1, ..., T_h$ have degree at least d_h and all vertices in $T_{h+1}, ..., T_k$ have vertices less than d_h. Then by the description of the algorithm, $T' = \bigcup_{i \leq h} T_h$.

Now we consider $H_{T'}$, and the number of clusters in $H_{T'}$ is h. For each $i \leq h$,

$$|T_i| \geq \left(\frac{1}{2k} - \frac{h+1/4}{4k^2}\right) t \geq \frac{t}{4k} \geq \frac{|T'|}{4k} = \frac{h}{4k} \frac{|T'|}{h}.$$

Thus let $b'' = \frac{h}{4k}$, then the partition $T_1, ..., T_h$ of T' is b''-balanced, and $|T'| \geq \frac{ht}{4k} \geq \frac{256c_0k^3\log n}{\delta^2}$, $\log|T'| \leq \log n$. Thus,

$$\frac{|T'|}{\log|T'|} \geq 16kc_0\frac{16k^2}{h^2}\frac{h^2}{\delta^2} \geq \frac{c_0h^2}{(b'')^2\delta^2}.$$

Therefore, by Lemma 2.3, $\mathsf{BalSBM}(H_{T'}, k, \delta, \frac{h}{4k})$ can recover sub-clusters $T_1, ..., T_h$ with probability at least $1 - |T'|^{-8}$. □

C.3. Proof of Lemma 4.4

Proof. We first provide a subroutines TestBias from (Peng & Zhang, 2021).

Algorithm 8 TestBias(n, B, η): test if B is (η, C)-biased for some cluster C

1: **for** $i = 1, ..., \frac{16k \log n}{b}$ **do**
2:      Randomly sample a vertex $v_i \in V$ and query all pairs v_i, u for $u \in B$
3:      **if** $d(v, B) \geq (\frac{1}{2} + \frac{1}{2}\eta\delta) |B|$ **then**
4:          **output** *Yes*
5:      **end if**
6: **end for**
7: **output** *No*

Lemma C.1 ((Peng & Zhang, 2021))**.** *Let B be a vertex set of size at least $\frac{64 \log n}{\eta^2 \delta^2}$. TestBias$(n, B, \eta)$ makes $O(\frac{k \log n |B|}{b})$ queries and satisfies, with probability at least $1 - n^{-7}$,*
(1) return Yes, if B is (η, C)-biased for some cluster C of size at least $\frac{bn}{k}$, i.e., $|B \cap C| \geq (\frac{1}{2} + \eta) |B|$;
(2) return No, if B is not $(\frac{\eta}{4}, C)$-biased for any cluster C, i.e. $|B \cap C| \leq (\frac{1}{2} + \frac{\eta}{4}) |B|$.

Algorithm 9 GapSubcluster(T, k, δ): find a good index h and the corresponding sub-clusters

---

1: **for** $h = k - 1, ..., 1$ **do**
2:      Invoke GapSBM(T, h, δ) to get h clusters $X_1, ..., X_h$
3:      **if** $|X_i| < \frac{t}{4k}$ for some $i \leq h$ **then**
4:          **continue**
5:      **end if**
6:      **if** for all $i \in [h]$, TestBias$(n, X_i, 0.1)$ returns *Yes* **then**
7:          **output** $X_1, ..., X_h$
8:      **end if**
9: **end for**
10: **output Fail**

By Lemma 4.2, there is some $h^* \leq k - 1$ satisfying the size gap condition, and thus by (1) of Lemma 4.3, with probability $1 - O(k^{-24} \log^{-8} n)$, at least one execution of GapSBM in the for loop will output h^* sub-clusters of $V_1, ..., V_{h^*}$ whose sizes satisfies $s_i \geq t/4k$ and $\sum_{i \leq h^*} s_i \geq n/2$. In particular, all these sub-clusters will pass TestBias. Therefore, if $s_k < n/4k$, GapSubcluster will not output **Fail** with probability $1 - O(k^{-24} \log^{-8} n)$.

On the other hand, if GapSubcluster succeeds, then the size of each X_i in the output at least $t/4k$, and thus (a) is proved. Moreover, each X_i is $(0.1, V_{g(i)})$-biased for some $g(i) \in [k]$ since they passed TestBias(n, X_i, η), which implies (b).

To prove (c), let h be final index when GapSubcluster successfully terminates and let $t_i = |T \cap V_i|$. By (2) of Lemma 4.3, all vertices in T that belongs to V_i with $s_i \geq \frac{n}{2k} - h \cdot \frac{n}{4k^2}$ must be contained in the output of GapSBM. Let ℓ be the largest number such that $s_\ell \geq \frac{n}{2k} - h \cdot \frac{n}{4k^2}$. In other words, for all $i \leq \ell$, $|(\bigcup_{i=1}^h X_i) \cap V_i| = t_i$. By Chernoff bound, with probability $1 - \frac{1}{n^5}$, $t_i \leq 1.01 \frac{ts_i}{n}$ holds for all $i \leq \ell$. Let $x_i = |X_i|$. Note, when $h \neq h^*$, we have no control of the output of GapSBM(T, h, δ); it could even be possible that $g(i) = g(j)$ for some $i, j \leq h$ and $i \neq j$. Since X_i is $(0.1, V_{g(i)})$-biased, which means at least 0.6 fraction of X_i are from $V_{g(i)}$, then

$$0.6 \sum_{j \in g^{-1}(i)} x_j \leq t_i \leq 1.01 \frac{ts_i}{n}.$$

Then sum the above inequality over all i such that $g^{-1}(i) \neq \emptyset$, we have

$$\sum_{i: g^{-1}(i) \neq \emptyset} s_i \geq 0.5 \sum_{j=1}^h x_j \cdot \frac{n}{t}. \tag{2}$$

We emphasize that $\sum_{j=1}^h x_j \neq t$, since we have pruned vertices from T with low degrees. However, we can still show that $\sum_{j=1}^h x_j = \Theta(t)$. To see this, first observe that $h \geq h^*$ since we try all h in a decreasing order. Then $d_h \leq d_{h^*}$ and

$T'_{h^*} \subset T'_h$. By Lemma 4.3, all vertices from $V_1, ..., V_{h^*}$ are preserved in T'_{h^*}, i.e., $|T'_{h^*}| \geq \sum_{i=1}^{h^*} t_i$. By Chernoff bound, $t_i \geq 0.99 \frac{t s_i}{n}$ and By Lemma 4.2, $\sum_{i=1}^{h^*} s_i \geq n/2$. As a result,

$$\sum_{j=1}^{h} x_j = |T'_h| \geq |T'_{h^*}| \geq \sum_{i=1}^{h^*} t_i \geq 0.99 \frac{t}{n} \sum_{i=1}^{h^*} s_i \geq 0.4t.$$

Together with (2), we finish the proof of (c).

For the sample complexity, GapSBM makes t^2 queries. Each TestBias$(n, X_i, 0.1)$ makes $O(k|X_i| \log n) \leq O(kt \log n)$. There are at most k^2 calls of TestBias$(n, X_i, 0.1)$, so the total complexity of this part is $O(k^3 t \log n) \leq O(t^2)$. □

C.4. Proof of Theorem 1.1.

Let $U_1, ..., U_k$ be the clusters of U at the beginning of the r-th round, and $s_i = |U_i|$. W.l.o.g., assume $s_1 \geq s_2 \cdots \geq s_k$.

Case 1: $s_k < |U|/4k$. By Lemma 4.4, then with probability $1 - o_n(1)$, GapSubcluster(T^r, k, δ) does not **Fail**. Conditioned on this, each X_i is $(0.1, U_{g(i)})$-biased and $|X_i| \geq |T^r|/4k$ (Lemma 4.4). For each $v \in U \setminus T^r$ and $v \in U_{g(i)}$ for some i, it holds that $j = i$ (line 13) with probability $7/8$ (Lemma 4.5). Then, with probability $1 - 1/n^6$ ClusterVerify(v, X'_j) output true and v is added to X_i. On the other hand, if $v \notin U_{g(i)}$ for all i, v remains in U with probability $1 - 1/n^6$. For $s_k < |U|/4k$, we say the round succeeds if GapSubcluster(T^r, k, δ) does not **Fail**, and for all v TrueClusterId$(v, X_1, ..., X_h, \delta, 1/8)$ and ClusterVerify(v, X'_j) output correctly. By union bound, the round succeeds with probability $1 - o_n(1)$. For a successful round, all $v \in U \setminus T^r$ such that $v \in U_{g(i)}$ for some i are added to Y_i and all other vertices v in U remain in U for the next round. By (3) of Lemma 4.4, a successful round reduces the number of vertices in U by a factor of $4/5$. So there are at most $4 \log n$ rounds.

Case 2: $s_k \geq |U|/4k$. Note GapSubcluster(T^r, k, δ) may or may not **Fail** in this case. Suppose, it outputs **Fail**, then BalNoisyClustering$(U, k, \delta, 1/4)$ is called. Now U is $1/4$-balanced, by Theorem 3.1, $X_1, ..., X_k$ are the true clusters of U with probability $1 - o_n(1)$ (the round succeeds); then $U = \emptyset$ and directly goes to the cleanup stage. GapSubcluster may not fail and still output $X_1, ..., X_h$, which satisfies the guarantees in Lemma 4.4. If this happens, then everything is the same as in Case 1, and the round succeeds with probability $1 - o_n(1)$.

Case 3: $|U| < \frac{ck^4 \log n}{\delta^2}$. In this case, the algorithm does nothing in this round and goes to the cleanup stage. To simplify the analysis, we assume that the algorithm always runs for exactly $4 \log n$ rounds by adding dummy rounds after U becomes too small. A dummy round is considered successful.

Error probability and unlabelled vertices. For $r \in [4 \log n]$, let E_r denote the event that the r-round succeeds and the merging result is correct at the end of the round. We have from the analyses above and in Section 4.4, $\Pr[E_r] \geq 1 - o_n(1)$. Since this lower bound holds for any configuration at the beginning of the round, $\Pr[E_r \mid E_1, ..., E_{r-1}] \geq 1 - o_n(1)$ for all r (since $E_1, ..., E_{r-1}$ only affects the initial configuration of the r-th round). Then, by the chain rule, we have

$$\Pr[E_1, ..., E_{4 \log n}] \geq (1 - o_n(1))^{4 \log n},$$

which is still $1 - o_n(1)$ since the $o_n(1)$ term above is smaller than $k^{-28} \log^{-8} n$.

Next we assume $E_1, ..., E_{4 \log n}$ happen. Before the cleanup stage, we get $C_1, ..., C_k$, and assume $|C_1| \geq \cdots \geq |C_k|$. It is possible $C_i = \emptyset$, and let i be the largest index such that $|C_i| > 0$. Note, conditioned on $E_1, ..., E_{4 \log n}$, each nonempty C_i has size at least $t/4k = \frac{ck^3 \log n}{4\delta^2}$. After the cleanup stage, each nonempty C_i will grow to the full cluster V_j for some $j \in [k]$. However, there could be some vertices remain unidentified. In the worst case, all vertices in $U \cup R$ are unidentified. Since $|U| < \frac{ck^4 \log n}{\delta^2}$ and $|R| \leq 4 \log n \cdot \frac{ck^4 \log n}{\delta^2} = \frac{4ck^4 \log^2 n}{\delta^2}$, we have $|U \cup R| \leq \frac{5ck^4 \log^2 n}{\delta^2}$. Since each V_i is either fully recovered or $V_i \subset U \cup R$, all V_i with $|V_i| \geq \frac{5ck^4 \log^2 n}{\delta^2}$ must have been recovered. We remark that algorithm by (Peng & Zhang, 2021) recovers all the clusters of size $\Omega(\frac{k^4 \log n}{\delta^2})$ and algorithm by (Mazumdar & Saha, 2017a) recovers all the clusters of size $\Omega(\frac{k \log n}{\delta^4})$. By running their algorithms on $U \cup R$, we achieve the same guarantees on the size of recovered clusters.

Query Complexity. Now we bound the query complexity of the final algorithm. Recall that there are at most $4 \log n$ rounds. In each round r, GapSubcluster makes $t^2 = \frac{k^8 \log^2 n}{\delta^4}$ queries. Similar as in the balanced case, TrueClusterId and ClusterVerify make $O(\frac{|U|(k + \log n)}{\delta^2})$ queries. In the merging step, at most $2k$ sub-clusters are merged, which incurs $O(\frac{k^2 \log n}{\delta^2})$ queries. Since $|U|$ decreases exponentially, over all rounds, the total query complexity is

$$O\left(\frac{n(k + \log n)}{\delta^2} + \frac{k^8 \log^3 n}{\delta^4}\right).$$

This finish the proof of Theorem 1.1.

D. Deferred Proofs from Section 5

Lemma D.1 (Lemma 5.1 restated). *There exist positive constant c_1, c_2, δ_0 and α_0, such that for every $\delta \in (0, \delta_0)$ and $\alpha \in (0, \alpha_0)$ and for every algorithm that returns FAIL with probability at most $\frac{1}{8}$ and returns a δ-optimal arm with probability at least $1 - \alpha$ conditioned on NOT FAIL, there exists some problem instances such that the expected number of trials satisfies*

$$\mathbb{E}[T] \geq c_1 \frac{1}{\delta^2} \log \frac{c_2}{\alpha}.$$

In particular, δ_0 and α_0 can be taken equal to $1/8$ and $e^{-8}/4$

Proof. The structure of this proof is similar to the proof of Theorem 1 in (Mannor & Tsitsiklis, 2004). Let us consider the following two problem instances that we will refer to as "hypotheses" denoted by H_0 and H_1 respectively.

$$H_0 : \mu_0 = \frac{1}{2} + \frac{\delta}{2}, \quad \mu_1 = \frac{1}{2} - \frac{\delta}{2},$$
$$H_1 : \mu_0 = \frac{1}{2} - \frac{\delta}{2}, \quad \mu_1 = \frac{1}{2} + \frac{\delta}{2}.$$

So under H_0, arm a_0 is optimal, while under H_1 arm a_1 is optimal.

Let $\delta_0 = 1/8$ and $\alpha_0 = e^{-8}/4$, fix some $\delta \in (0, \delta_0)$, $\alpha \in (0, \alpha_0)$ and any algorithm that satisfies the condition in the lemma. We denote by $\mathbb{E}_l$ and $\mathbb{P}_l$ the expectation and probability under hypothesis H_l, respectively. Let T_i be the number of trails that involves arm a_i, so $T = T_0 + T_1$. We define t^* by

$$t^* = \frac{1}{c\delta^2} \log \frac{1}{4\alpha} = \frac{1}{c\delta^2} \log \frac{1}{\theta},$$

where $\theta = 4\alpha$ and c is an universal constant that we will specify its value later. Throughout the proof, we use E^C to denote the complement of event E.

Assume that $\mathbb{E}_0[T] \leq t^*$, and we claim that under this assumption, the probability that the algorithm returns arm a_0 under H_1 conditioned on NOT FAIL will exceeds α, thus violates the condition in the lemma.

We define some events A and C_0, C_1 where various random variables do not deviate significantly from their expected values. Define

$$A = \{T \leq 4t^*\}.$$

By Markov's inequality,

$$\mathbb{P}_0(A^C) = \mathbb{P}_0(T > 4t^*) \leq \frac{\mathbb{E}_0[T]}{4t^*} \leq \frac{t^*}{4t^*} = \frac{1}{4}.$$

It follows that

$$\mathbb{P}_0(A) \geq \frac{3}{4}.$$

Define $K_i^t = \sum_{s=1}^t r_i^s$ to be the total reward if arm a_i is tried t (not necessarily consecutive) times, where r_i^s is the reward

obtained at the s-th time arm a_i is tried. Define

$$C_0 = \left\{ \max_{1 \leq t \leq 4t^*} \left| K_0^t - (\frac{1}{2} + \frac{\delta}{2})t \right| < \sqrt{t^* \log(\frac{1}{\theta})} \right\},$$

$$C_1 = \left\{ \max_{1 \leq t \leq 4t^*} \left| K_1^t - (\frac{1}{2} - \frac{\delta}{2})t \right| < \sqrt{t^* \log(\frac{1}{\theta})} \right\},$$

and $C = C_0 \cap C_1$.

Lemma D.2 (Generalized Lemma 2 in (Mannor & Tsitsiklis, 2004)). *We have* $\mathbb{P}_0(C) > \frac{3}{4}$ *for* $i = 0, 1$.

Proof. We first prove a more general result. Suppose the expected reward of arm a_i is $\mu_i^{(l)}$ under H_l, let

$$C_i^{(l)} = \left\{ \max_{1 \leq t \leq 4t^*} \left| K_i^t - \mu_i^{(l)} t \right| < \sqrt{t^* \log(\frac{1}{\theta})} \right\}.$$

Note that $K_i^t - \mu_i^{(l)} t$ is a $\mathbb{P}_l$-martingale. By Kolmogorov's inequality,

$$\mathbb{P}_l \left((C_i^{(l)})^C \right) = \mathbb{P}_l \left(\max_{1 \leq t \leq 4t^*} \left| K_i^t - \mu_i^{(l)} t \right| \geq \sqrt{t^* \log(\frac{1}{\theta})} \right) \leq \frac{\mathbb{E}_l[(K_i^{4t^*} - 4t^* \mu_i^{(l)})^2]}{t^* \log(\frac{1}{\theta})}.$$

Since $\mathbb{E}_l[K_i^{4t^*}] = 4t^* \mu_i^{(l)}$, so $\mathbb{E}_l[(K_i^{4t^*} - 4t^* \mu_i^{(l)})^2] = 4t^* \mu_i^{(l)}(1 - \mu_i^{(l)})$. Note that $\theta < e^{-8}$ then we have

$$\mathbb{P}_l(C_i) \geq 1 - \frac{4t^* \mu_i^{(l)}(1 - \mu_i^{(l)})}{t^* \log \frac{1}{\theta}} \geq \frac{7}{8},$$

where the last inequality follows because $\mu_i^{(l)}(1 - \mu_i^{(l)}) \leq \frac{1}{4}$. Let $l = 0$, then $\mathbb{P}_0(C_i) \geq \frac{7}{8}$ for $i = 0, 1$, and by a simple union bound, $\mathbb{P}_0(C) \geq \frac{3}{4}$. $\quad\square$

Lemma D.3. *If* $0 \leq x \leq \frac{1}{\sqrt{2}}$ *and* $y \geq 0$, *then*

$$(1 - x)^y \geq e^{-dxy},$$

where $d = 1.78$.

Proof. Simple calculation shows that $\log(1 - x) + dx \geq 0$ for any $x \in [0, \frac{1}{\sqrt{2}}]$, rearranging terms and taking $y \geq 0$ gives the proof. $\quad\square$

Let D be the event that the algorithm returns FAIL. Let I be the arm that the algorithm returns if NOT FAIL. Let $B = \{I = a_0\}$, i.e. the algorithm returns arm a_0. Recall the condition in the lemma, we have $\mathbb{P}_0(D) \leq \frac{1}{8}$, $\mathbb{P}_0(B|D^C) \geq 1 - \alpha$, $\alpha < e^{-8}/4 < \frac{1}{8}$, so $\mathbb{P}_0(B) \geq (1 - \mathbb{P}_0(D))(1 - \alpha) > \frac{3}{4}$. Let $S = A \cap B \cap C$ be the event that A, B, C occur simultaneously. Since $\mathbb{P}_0(A) \geq \frac{3}{4}$, $\mathbb{P}_0(B) \geq \frac{3}{4}$, $\mathbb{P}_0(C) > \frac{3}{4}$, we then have $\mathbb{P}_0(S) > \frac{1}{4}$.

Now we prove that if $c \geq 400$ and $\mathbb{E}_0[T] \leq t^*$, then $\mathbb{P}_1(B) > \delta$.

Let W be the history of the process (the sequence of arms selected at each time and corresponding rewards) until the algorithm terminates. Define the likelihood function L_l by

$$L_l(w) = \mathbb{P}_l(W = w),$$

for every possible history w. Note that the choices of the algorithm only depend on the history, and are independent of which hypothesis is true. Let $K_0 := K_0^{T_0}$, $K_1 := K_1^{T_1}$. The likelihood ratio is given by

$$\frac{L_1(W)}{L_0(W)} = \frac{(\frac{1}{2} + \frac{\delta}{2})^{K_1 + T_0 - K_0}(\frac{1}{2} - \frac{\delta}{2})^{K_0 + T_1 - K_1}}{(\frac{1}{2} + \frac{\delta}{2})^{K_0 + T_1 - K_1}(\frac{1}{2} - \frac{\delta}{2})^{K_1 + T_0 - K_0}}$$

$$= \left(\frac{1 - \delta}{1 + \delta} \right)^{2(K_0 - \frac{1+\delta}{2}T_0) - 2(K_1 - \frac{1-\delta}{2}T_1) + \delta(T_0 + T_1)}$$

$$\geq (1 - 2\delta)^{2\left|K_0 - \frac{1+\delta}{2}T_0\right| + 2\left|K_1 - \frac{1-\delta}{2}T_1\right| + \delta T},$$

where the last inequality is because $1 > \frac{1-\delta}{1+\delta} \geq 1 - 2\delta$. We now lower bound the right-hand side above when S occurs. Since $S = A \cap B \cap C$ occurs, we have the followings

$$T \leq 4t^* = \frac{4}{c\delta^2} \log(\frac{1}{\theta}),$$

$$\left| K_1 - \frac{1-\delta}{2} T_1 \right| \leq 2\sqrt{t^* \log(\frac{1}{\theta})} = \frac{2}{\sqrt{c}\delta} \log(\frac{1}{\theta}),$$

$$\left| K_0 - \frac{1+\delta}{2} T_0 \right| \leq 2\sqrt{t^* \log(\frac{1}{\theta})} = \frac{2}{\sqrt{c}\delta} \log(\frac{1}{\theta}).$$

Thus,

$$\frac{L_1(W)}{L_0(W)} \geq (1 - 2\delta)^{2 \cdot \frac{2}{\sqrt{c}\delta} \log(\frac{1}{\theta}) + \delta \cdot \frac{4}{c\delta^2} \log(\frac{1}{\theta})}$$

$$= (1 - 2\delta)^{(\frac{4}{\sqrt{c}} + \frac{4}{c}) \frac{1}{\delta} \log(\frac{1}{\theta})}$$

$$\geq e^{-d \cdot 2\delta \cdot (\frac{4}{\sqrt{c}} + \frac{4}{c}) \frac{1}{\delta} \log(\frac{1}{\theta})}$$

$$= \theta^{\frac{8d}{\sqrt{c}} + \frac{8d}{c}},$$

where the last inequality is by Lemma D.3.

By picking sufficiently large c ($c = 400$ suffices), we obtain that $\frac{8d}{\sqrt{c}} + \frac{8d}{c} \leq 1$, thus $\frac{L_1(W)}{L_0(W)} \geq \theta = 4\alpha$. Note that this holds for every possible history $W = w$ whenever S occurs, i.e. more precisely,

$$\frac{L_1(W)}{L_0(W)} \mathbb{1}_S \geq 4\alpha \mathbb{1}_S,$$

where $\mathbb{1}_S$ is the indicator function of event S. Then

$$\mathbb{P}_1(B) \geq \mathbb{P}_1(S) = \mathbb{E}_1[\mathbb{1}_S] = \mathbb{E}_0 \left[\frac{L_1(W)}{L_0(W)} \mathbb{1}_S \right] \geq \mathbb{E}_0[4\alpha \mathbb{1}_S] = 4\alpha \mathbb{P}_0(S) > \alpha,$$

where the last inequality is because $\mathbb{P}_0(S) > \frac{1}{4}$.

To summarize, we have shown that when $c \geq 400$, if $\mathbb{E}_0[T] \leq \frac{1}{c\delta^2} \log(\frac{1}{4\alpha})$, then $\mathbb{P}_1(B|D^C) \geq \mathbb{P}_1(B) > \alpha$, i.e. the algorithm will return arm a_0 with probability larger than α under H_1, which is not a δ-optimal arm. So for every algorithm that satisfies the condition in the lemma, we must have $\mathbb{E}_0[T] > \frac{1}{c\delta^2} \log(\frac{1}{4\alpha})$ $\square$