# Clustering Items From Adaptively Collected Inconsistent Feedback

**Shubham Gupta**
IBM Research
Orsay, France

**Peter Staar**
IBM Research
Rüschlikon, Switzerland

**Christian de Sainte Marie**
IBM France Lab
Orsay, France

## Abstract

We study clustering in a query-based model where the learner can repeatedly query an oracle to determine if two items belong to the same cluster. However, these queries are costly and the oracle's responses are marred by inconsistency and noise. The learner's goal is to adaptively make a small number of queries and return the correct clusters for *all* $n$ items with high confidence. We develop efficient algorithms for this problem using the sequential hypothesis testing framework. We derive high probability upper bounds on their sample complexity (the number of queries they make) and complement this analysis with an information-theoretic lower bound. In particular, we show that our algorithm for two clusters is nearly optimal when the oracle's error probability is a constant. Our experiments verify these findings and highlight a few shortcomings of our algorithms. Namely, we show that their sample complexity deviates from the lower bound when the error probability of the oracle depends on $n$. We suggest an improvement based on a more efficient sequential hypothesis test and demonstrate it empirically.

## 1 INTRODUCTION

This paper explores a *query-based clustering* problem. The learner receives $n$ items to cluster and has access to an *oracle* that, given a pair of items, can answer the question "are these items in the same cluster?" We consider a *noisy* and *inconsistent* variant of this oracle: when asked the same query multiple times, it independently flips a biased coin *each time*, returning the correct answer with probability $p$ for same-cluster items and an incorrect answer with probability $q < p$ for different-cluster items. This is in contrast with *consistent* oracles that sample a noisy response *once* for each item pair, but return the same value every time this pair is queried (Mazumdar and Saha, 2017a). The learner's goal is to return the underlying ground-truth clusters with high confidence while minimizing the number of queries made to the oracle.

As a motivating example, consider the problem of identifying plant species in a newly discovered *complex* (a collection of visually similar species). Saryan et al. (2020) advocate using clustering for a data-driven solution to this problem. For instance, one can show two plant samples to a taxonomist and ask them if they belong to the same species. Unfortunately, as Saryan et al. (2020) point out, the answers to these queries are neither reliable nor consistent across experts as the similarity between the samples baffles them as well. In such a case, one stands to gain by aggregating information from multiple experts before returning the final clusters. The challenge lies in doing this with as few queries as possible to save both time and money.

Besides species delimitation, this model may provide a simple theoretical abstraction for many other applications. For example, clustering *compatible* molecules is useful for developing more potent drugs (Twarog et al., 2020). This compatibility is tested via drug-interaction experiments that are noisy due to natural biological variations in the samples. One can view each experiment as a query to a noisy oracle and use our approach for clustering molecules. This pattern also extends to some crowdsourcing applications where workers make errors. Aggregating information across repeated queries in a principled manner is crucial for accurate results (see Firmani et al. (2016); Luo et al. (2018); Vaughan (2018) and the references therein). We explore an alternative motivation for our model in Appendix A with an eye towards privacy. There, we argue why one might want to simulate our model, even if the actual item-item relationships are more complex.

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

Davidson et al. (2014) studied this problem in the special case where $p = \frac{1}{2} + \epsilon$, $q = \frac{1}{2} - \epsilon$, and $\epsilon \in (0, 1/2]$ is known to the learner. This assumes that the oracle is correct more than 50% of the time, and the learner knows the error rate. We remove both of these restrictions. Our oracle can be arbitrarily noisy as long as $q < p$ and we do not require $p$ and $q$ to be known. Yun and Proutiere (2014) also treat $p$ and $q$ as unknowns, but with a different end goal. They want to minimize the clustering error after a fixed number of queries, while we want to minimize the number of queries to get zero clustering error with a desired confidence level.

Mazumdar and Saha (2017a) studied this problem under the *consistent* oracle assumption mentioned above (same answer for repeated queries). This oracle is more suitable for problems like finding clusters in a social network where each edge, once observed, is either present or absent. However, in many applications like the ones mentioned above, it is possible to gain additional information from repeated queries (e.g., if the first drug-interaction experiment fails, it does not mean that all subsequent experiments will also fail). An inconsistent oracle is a better model in such cases.

To see this further, note that a consistent oracle yields one bit of information for each item pair. The noisier an oracle, the more bits the learner needs to effectively cluster the items. However, the maximum number of bits that can be collected is constrained by the number of item pairs. This becomes problematic when dealing with a small set of items or with clusters that only have a few items. For instance, the error probability of Mazumdar and Saha (2017a)'s algorithm depends on $n$. They also assume that each cluster has at least $\Omega(\ln n)$ items. Therefore, assuming the oracle to be consistent when it is not adds avoidable limitations.

We restrict our focus to the cases where repeated queries are allowed and the oracle is inconsistent. We present a statistically sound and nearly optimal clustering method that works for arbitrarily small item sets with a constant error probability irrespective of the cluster size and the noise level in the oracle.

**Contributions and Results:** The total number of queries made by an algorithm is called its *sample complexity*. Here is a summary of our contributions.

**1)** In Section 3, we establish a $\Omega(n/d(p,q))$ information theoretic lower bound on the sample complexity of any algorithm for our problem. Here, $d(p,q) = p\ln(p/q) + (1-p)\ln((1-p)/(1-q))$ is the Kullback-Leibler (KL) divergence between two Bernoulli distributions with success probability $p$ and $q$, respectively. Our proof adapts ideas from the best-arm identification literature (Kaufmann et al., 2016), and is valid for all $0 < q < p < 1$.

**2)** In Section 4, we present a hypothesis testing based meta-algorithm for dividing items into two clusters. We explain how a circular dependence between $p$, $q$ and the cluster memberships makes this problem challenging when $p$ and $q$ are unknown. Our algorithm for two clusters, `SplitItemsH`, implements this meta-algorithm using a test based on Hoeffding's lemma. For $k \geq 2$ clusters, our algorithm, `QBClusterH`, repeatedly calls `SplitItemsH` on leftover items until everything is clustered, without knowing $k$ in advance.

**3)** In Section 5, we analyze the correctness and sample complexity of our algorithms. Specifically, we show the near optimality of `SplitItemsH` for constant $p$ and $q$.

**4)** In Section 6, we present numerical evidence to support our analysis and evaluate the robustness of our algorithms to model misspecification. We also highlight a few limitations of our work. In particular, we show that the sample complexity of `SplitItemsH` deviates from the lower bound when $p$ and $q$ depend on $n$. Experiments confirm that this is easily fixed by using a stronger hypothesis test in our meta-algorithm.

**Related Work:** Clustering algorithms have long used different oracles to aid the otherwise unsupervised clustering process. For example, some oracles label a small subset of items directly (Ashtiani and Ben-David, 2015; Gadde et al., 2016). Others provide feedback on the clusters by recommending merge or split operations (Balcan and Blum, 2008; Awasthi et al., 2017). We consider yet another type of oracle that can noisily tell if a pair of items belong to the same cluster (Davidson et al., 2014; Yun and Proutiere, 2014).

Ashtiani et al. (2016) used a noise free variant of this oracle to reduce the computational cost of distance-based clustering algorithms like $k$-means (MacQueen, 1967)). Several follow-up works have strengthened their results for different variants of distance-based clustering (Ailon et al., 2018b; Gamlath et al., 2018; Bressan et al., 2020a; Li et al., 2021), correlation clustering (Ailon et al., 2018a; Saha and Subramanian, 2019; Bressan et al., 2020b), and graph partitioning (Mazumdar and Saha, 2017b). In all cases, the oracle is noise free whereas we consider a noisy oracle.

Mazumdar and Saha (2017a) introduced a noisy but consistent oracle, as explained earlier. They assumed that $p$ and $q$ are *known* and centered around 1/2, and presented two algorithms, one computationally efficient but query sub-optimal, and the other inefficient but query optimal. Algorithms that are both efficient and optimal were later developed, first for two clusters (Larsen et al., 2020), and then for the general case (Peng and Zhang, 2021; Xia and Huang, 2022). Others have studied variants where the oracle can also say "I don't know" (Kim and Ghosh, 2017) or return

an adversarially chosen answer (Pia et al., 2022). Also see (Mitzenmacher and Tsourakakis, 2016, 2020) for related problems. In contrast, our oracle is inconsistent and supports *unknown* and arbitrary $q < p$. The tradeoffs of these choices are explained above in detail.

Our ideas are inspired from the literature on best-arm identification (BAI) (Bubeck et al., 2009; Lattimore and Szepesvári, 2020). Think of each possible clustering of items as an arm and the ground-truth clustering as the best arm. Then, the goal is to use queries to quickly find the best arm. However, we can't simply use existing BAI algorithms for two reasons: **1)** The number of possible clusterings (and hence arms) grows exponentially with the number of items and clusters. **2)** Our oracle provides *local* feedback on an item pair instead of *global* feedback on an entire configuration of clusters. Therefore, we cannot "pull" an arm in the traditional sense. See Gentile et al. (2014) and Yang et al. (2022) for examples of clustering arms where the feedback from each arm is directly observable.

Finally, our oracle is reminiscent of the planted-partition model for graph partitioning (Abbe, 2018). Assume that the items correspond to the nodes in a graph. The oracle adds an edge between two items with probability $p$ (resp. $q$) if they belong to the same (resp. different) cluster. However, unlike the traditional planted-partition model, our oracle allows item pairs to be queried multiple times, which is useful in applications like species delimitation where the queries are noisy but repeatable. Yun and Proutiere (2014) also use a similar model and highlight this distinction, though, as explained above, for a different problem.

## 2 PROBLEM SETTING

Let $[n]$ be the set $\{1, 2, \ldots, n\}$ for any integer $n > 0$ and $\mathbf{I}\{\texttt{prop}\}$ indicate the truth value of a proposition `prop`. We consider the problem of partitioning a given set of $n$ items by adaptively choosing items pairs to query a noisy oracle that tells if these items are in the same cluster. We call this the *Query-Based Clustering* (`QBC`) problem, and define it formally below.

**Definition 2.1** (`QBC`)**.** *An instance of the Query-Based Clustering (`QBC`) problem is specified by a five tuple $\nu = (n, \rho, k, p, q)$, where $n$ is the number of items, $\rho : [n] \to [k]$ maps items to one of the $k \geq 2$ non-empty clusters, and $0 \leq q < p \leq 1$ are model parameters. Querying a pair of items $\{i, j\}$, where $i \neq j$, returns a binary feedback $O_{ij}$ sampled as follows:*

$$\mathrm{P}\left(O_{ij} = 1\right) = \begin{cases} p & \text{if } \rho(i) = \rho(j) \\ q & \text{otherwise.} \end{cases} \tag{1}$$

The learner only knows $n$, while $k$, $p$, $q$, and $\rho$ are unknown. Its goal is to return a cluster assignment function $\hat{\rho} : [n] \to [\hat{k}]$ that *matches* the ground-truth clusters in $\rho$ with high probability. That is, $\hat{k} = k$ and there is a permutation function $\pi : [k] \to [k]$ such that $\pi(\hat{\rho}(i)) = \rho(i)$ for all items $i$. Denoting this event by $\hat{\rho} = \rho$, the learner must execute a $\delta$-probably approximately correct ($\delta$-PAC) algorithm defined below.

**Definition 2.2** ($\delta$-PAC algorithm)**.** *An algorithm is called $\delta$-PAC with confidence parameter $\delta \in [0, 1]$ if, for any `QBC` instance $\nu = (n, \rho, k, p, q)$, it terminates after a finite number of queries $\tau$ and returns a $\hat{\rho}$ that satisfies $\hat{\rho} = \rho$ with probability at least $1 - \delta$.*

The number of queries $\tau$ made by an algorithm is called its *sample complexity*. Our goal is to develop $\delta$-PAC algorithms with a low sample complexity. In the next section, we begin with a lower bound on the sample complexity of any $\delta$-PAC algorithm for our problem.

## 3 LOWER BOUND

The learner receives feedback from the oracle according to eq. (1) each time it makes a query. We want to ask if the evidence collected until time $t$ is enough to ensure with high probability that the observations are from a `QBC` instance $\nu$ with cluster assignment $\rho$ and not any other instance $\nu'$ with a different assignment $\rho' \neq \rho$. To test the statistical limits of how fast this question can be answered, we consider the log-likelihood ratio of the observations under $\nu$ and $\nu'$.

More concretely, let $O(1), \ldots, O(t)$ be the observations received by an algorithm. We use $\ell_\nu(O(1), \ldots, O(t))$ to denote the likelihood of these observations under instance $\nu$, and define the log-likelihood ratio as $L_{\nu,\nu'}(t) \coloneqq \ln \frac{\ell_\nu(O(1),\ldots,O(t))}{\ell_{\nu'}(O(1),\ldots,O(t))}$. Intuitively, one can be confident that the observations are coming from $\nu$ instead of $\nu'$ if $L_{\nu,\nu'}(t)$ is high, and vice versa. Therefore, deriving an upper and a lower bound on $L_{\nu,\nu'}(t)$ provides a limit on the minimum number of queries needed before $L_{\nu,\nu'}(t)$ becomes high (or low) enough to identify the correct instance with high confidence.

In Appendix B, we compute an expression for the log-likelihood ratio and derive these bounds using a *change-of-measure* argument from the bandits literature (Lai and Robbins, 1985; Kaufmann et al., 2016). We then find a set of "challenging" instances $\nu'$ that match $\nu$ in the cluster assignment of all but one item, and get the fundamental limit summarized below.

**Theorem 3.1** (Lower bound)**.** *Let $\nu = (n, \rho, k, p, q)$ be a `QBC` instance with $0 < q < p < 1$. Assume that each cluster has at least two items. The sample complexity $\tau$ of any $\delta$-PAC algorithm satisfies $\mathrm{E}_\nu[\tau] \geq \frac{n}{2 \max\left\{d(p,q), \frac{d(q,p)}{k-1}\right\}} \ln \frac{1}{2.4\delta}$.*

We described how our model relates to the planted partition model towards the end of Section 1. The fundamental limits for clustering under that model assume only one observation for each item pair but study the case with infinite items. They show that clustering eventually becomes impossible as $p$ and $q$ get close, even if one queries infinitely many item pairs (each pair queried only once). For example, when $p = A\frac{\ln n}{n}$ and $q = B\frac{\ln n}{n}$ for some $A > B > 0$, exact recovery is possible with probability $1-o_n(1)$ iff $\sqrt{A}-\sqrt{B} > \sqrt{k}$ (Abbe and Sandon, 2015). The remark below describes how Theorem 3.1 complements these fundamental limits.

**Remark 3.1** (Finite items, repeated queries)**.** Instead of saying that even infinite queries will not help, Theorem 3.1 finds the minimum number of possibly repeated queries that will in fact allow cluster recovery for any fixed $n$. This is achieved by shifting the querying budget from one query for an infinite number of item pairs to several queries for finitely many pairs.

# 4 ALGORITHMS

With a lower bound at hand, we now move towards developing efficient and nearly-optimal algorithms. Section 4.1 considers the simpler case of two clusters. Section 4.2 extends these ideas to the general case.

## 4.1 `SplitItemsH` for Two Clusters

Let $\mathcal{N}$ denote the set of items to be partitioned into two clusters. We face two challenges.

**1) Oracle noise:** Our oracle is noisy. If it had no noise, we could have simply fixed an *anchor* item $a$ and, for each item $i \in \mathcal{N}\backslash\{a\}$, queried $\{a, i\}$ to check if $a$ and $i$ are in the same cluster. This partitions $\mathcal{N}$ into two clusters using $O(n)$ queries, an improvement over the naive baseline that ignores the transitivity of cluster memberships and queries all $O(n^2)$ item pairs.

**2) Unknown $p$ and $q$:** Even with noise, determining item $i$'s cluster would have been a relatively simple exercise if $p$ and $q$ were known. One can query $\{a, i\}$ repeatedly until the hypothesis $\mathrm{E}[O_{ai}] = p$ is accepted or rejected with high confidence, where $O_{ai}$ is the feedback from eq. (1). Unfortunately, the *unknown* value of $p$ is needed to even formulate this hypothesis.

A natural instinct is to first estimate $p$ and $q$ and then test the hypothesis described above. For instance, one can divide the observations for $p$ and $q$ in separate buckets and average each bucket. However, dividing the observations requires knowing the cluster memberships. This creates a circular dependency between $(p, q)$ and the clusters. As an alternative, one may randomly select a small subset of items and repeatedly query all pairs in this set to estimate $p$ and $q$ (Abbe et al., 2020). However, this assumes that all clusters are sufficiently large. For example, when one of the two clusters is very small, the first step may not select any item from it, leading to incorrect estimates.

The meta-algorithm below proposes a different sequential hypothesis testing strategy that does not require knowing $p$ or $q$ and works as long as all clusters have at least two items. We define $\alpha_{ai} \coloneqq \mathrm{E}[O_{ai}]$.

**Meta-Algorithm 1.** Assume that each cluster has at least two items. Pick an anchor $a$ and query item pairs $\{a, i\}$ for all $i \in \mathcal{N}\backslash\{a\}$ at each step. Let $\mathcal{H}_{ij}$ be the hypothesis that $\alpha_{ai} > \alpha_{aj}$. The algorithm terminates when, for each item $i \in \mathcal{N}\backslash\{a\}$, there is an item $j_i \notin \{a, i\}$ for which $\mathcal{H}_{ij_i}$ can be accepted/rejected with high confidence. Put item $i$ in the same cluster as $a$ if $\mathcal{H}_{ij_i}$ was accepted and in the other cluster otherwise.

Testing $\alpha_{ai} > \alpha_{aj}$ instead of $\alpha_{ai} = p$ eliminates the need to know $p$ in advance. Because we know that $\alpha_{ai}$ takes only two values $p$ and $q$, $\alpha_{ai}$ must be $p$ and $\alpha_{aj}$ must be $q$ if the test $\alpha_{ai} > \alpha_{aj}$ succeeds, making the cluster assignments in Meta-Algorithm 1 valid.

The final missing piece is how to safely test these hypotheses. The remark below first presents a best-arm identification (BAI) view of this problem (Bubeck et al., 2009). We then describe a test based on Hoeffding's inequality, and an algorithm based on this test.

**Remark 4.1** (Best-arm identification)**.** Think of the pair $\{a, i\}$ as an arm with expected reward $\alpha_{ai}$. Testing the hypothesis $\mathcal{H}_{ij}$ is equivalent to solving a BAI problem between arms $\{a, i\}$ and $\{a, j\}$. In this sense, Meta-Algorithm 1 simultaneously solves $|\mathcal{N}| - 2$ BAI problems for each item, assigning them to a cluster as soon as at least one of these problems terminate.

A common approach to solving a two-arm BAI problem involves using the samples to construct confidence intervals for the arm means. A winner is declared once these intervals stop overlapping. Our first algorithm, `SplitItemsH`, uses confidence intervals $\mathcal{I}_{ai}(t) = [l_{ai}(t), u_{ai}(t)]$ derived from a Hoeffding-style inequality. See Appendix C for a proof of the lemma below.

**Lemma 4.1.** *Define* $\beta(t, \delta) = \ln \frac{2(t+1)t^2|\mathcal{N}|^2}{\delta}$. *Let* $a$ *be the anchor chosen by* `SplitItemsH`. *Set*

$$\hat{\mu}_{ai}(t) = \frac{1}{t}\sum_{s=1}^{t} O_{ai}(s), \;\; l_{ai}(t) = \hat{\mu}_{ai}(t) - \sqrt{\frac{\beta(t,\delta)}{2t}},$$
$$\text{and } \; u_{ai}(t) = \hat{\mu}_{ai}(t) + \sqrt{\frac{\beta(t,\delta)}{2t}}, \tag{2}$$

*where* $O_{ai}(s)$ *is the outcome of querying item pair*

**Algorithm 1** `SplitItemsH`$(\mathcal{N}, \delta)$ for $k = 2$

1: **Input:** Set of items $\mathcal{N}$, confidence parameter $\delta$
2: Sample an anchor $a \overset{\text{unif}}{\sim} \mathcal{N}$
3: Set $t = 1$, $\mathcal{C}_1 = \{a\}$, $\mathcal{C}_2 = \phi$, and $\mathcal{U}(1) = \mathcal{N} \backslash \{a\}$
4: **while** $|\mathcal{U}(t)| > 0$ **do**
5: `# Query the oracle`
6: Sample item pairs $\{a, i\}$ for all $i \in \mathcal{N} \backslash \{a\}$
7: Update $\hat{\mu}_{ai}(t)$, $l_{ai}(t)$ and $u_{ai}(t)$ using eq. (2)
8: Update $\hat{p}_l(t)$ and $\hat{q}_u(t)$ using eq. (3)
9:
10: `# Update the clusters`
11: $\mathcal{C}_1 = \mathcal{C}_1 \cup \{i \in \mathcal{U}(t) : l_{ai}(t) > \hat{q}_u(t)\}$
12: $\mathcal{C}_2 = \mathcal{C}_2 \cup \{i \in \mathcal{U}(t) : u_{ai}(t) < \hat{p}_l(t)\}$
13:
14: `# Update unassigned items`
15: $\mathcal{U}(t+1) = \mathcal{U}(t) \backslash (\mathcal{C}_1 \cup \mathcal{C}_2)$; $t = t + 1$
16: **end while**
17: **Return:** $\mathcal{C}_1$ and $\mathcal{C}_2$

$\{a, i\}$ *at time* $s$. *Further, define,*

$$\hat{p}_l(t) \coloneqq \max_{i \in \mathcal{N} \backslash \{a\}} l_{ai}(t) \quad \textit{and} \quad \hat{q}_u(t) \coloneqq \min_{i \in \mathcal{N} \backslash \{a\}} u_{ai}(t). \tag{3}$$

*Then, the event* $\{\forall t \geq 1,\ \ p \geq \hat{p}_l(t) \wedge q \leq \hat{q}_u(t) \wedge \forall i \in \mathcal{N} \backslash \{a\},\ \alpha_{ai} \in [l_{ai}(t), u_{ai}(t)]\}$ *happens with probability at least* $1 - \delta$.

Hypothesis $\mathcal{H}_{ij}$ is accepted at time $t$ if $l_{ai}(t) > u_{aj}(t)$ (i.e., the confidence intervals stop overlapping). Notice how our confidence intervals are valid at *all* steps. This bounds the probability of *any* hypothesis $\mathcal{H}_{ij}$ being mistakenly accepted/rejected at *any* time $t$ by $\delta$. Following Meta-Algorithm 1, we assign item $i$ to the same cluster as $a$ if there is a $j \in \mathcal{N} \backslash \{a, i\}$ such that $\mathcal{H}_{ij}$ is accepted, or equivalently when $l_{ai}(t) > \hat{q}_u(t)$. Similarly, $i$ is assigned to the other cluster if $u_{ai}(t) < \hat{p}_l(t)$. Algorithm 1 summarizes this procedure.

Lines 11 and 12 in Algorithm 1 show that `SplitItemsH` uses $\hat{p}_l(t)$ and $\hat{q}_u(t)$ as proxies for $p$ and $q$, respectively. Because the problem becomes much simpler if $p$ and $q$ were known, it is natural to quantify the penalty, if any, that we pay for using $\hat{p}_l(t)$ and $\hat{q}_u(t)$ instead of $p$ and $q$. The next remark addresses this issue. See Algorithm 1 for the definition of clusters $\mathcal{C}_1$ and $\mathcal{C}_2$.

**Remark 4.2** (Penalty for not knowing $p$ and $q$)**.** Consider two cases. **1)** If we know $p$ and $q$, we can assign item $i$ to $\mathcal{C}_1$ (resp. $\mathcal{C}_2$) when $l_{ai}(t) > q$ (resp. $u_{ai}(t) < p$). This requires $u_{ai}(t) - l_{ai}(t) < p - q$ in the worst case; e.g., when $\alpha_{ai} = p$ and $u_{ai} \approx p$. **2)** If we use $\hat{p}_l(t)$ and $\hat{q}_u(t)$ instead of $p$ and $q$, the worst case happens when $u_{ai}(t) \approx p$ for items with $\alpha_{ai} = p$ and $l_{ai}(t) \approx q$ for items with $\alpha_{ai} = q$. Now, we need $u_{ai}(t) - l_{ai}(t) < (p - q)/2$ before the confidence intervals separate. Therefore, not knowing $p$ and $q$ requires us to shrink the confidence intervals by twice as much, which amounts to four times the queries by Lemma 4.1. Hence, `SplitItemsH` pays only a constant multiplicative penalty in Case 2.

Appendix C shows that it is possible to stop querying an item pair $\{a, i\}$ as soon as item $i$ is assigned to a cluster. However, this makes the algorithm more complex without improving its worst case guarantees. Next, we describe our algorithm for $k \geq 2$ clusters.

### 4.2 `QBClusterH` for $k \geq 2$ Clusters

`QBClusterH` works in phases. The $j^{th}$ phase begins with a set of unclustered items $\mathcal{N}^j \subseteq \mathcal{N}$ (with $\mathcal{N}^1 = \mathcal{N}$), and calls `SplitItemsH` to divide them into two groups $\mathcal{C}_1^j$ and $\mathcal{C}_2^j$. $\mathcal{C}_1^j$ has all items in the same cluster as the anchor $a^j \in \mathcal{N}^j$ chosen in `SplitItemsH` and $\mathcal{C}_2^j$ has the remaining items. The *pure* cluster $\mathcal{C}_1^j$ is added as the $j^{th}$ output cluster $\mathcal{C}_j$ and the set of unclustered items is set to $\mathcal{N}^{j+1} = \mathcal{C}_2^j$. We need two more details.

*First*, we must set the confidence parameter $\delta^j$ for the $j^{th}$ call to `SplitItemsH`. We use $\delta^j = \delta / j(j+1)$, where $\delta$ is confidence in the final output expected by the user. This ensures that *all* calls to `SplitItemsH` succeed (and hence the final output is correct) with probability at least $1 - \sum_j \delta^j \geq 1 - \delta$.

*Second*, we need to address a boundary condition for the last cluster. When there are $k$ clusters, all the items in $\mathcal{N}^k$ will belong to the same cluster with high probability, making $\alpha_{a^k i} = p$ for all $a^k, i \in \mathcal{N}^k$. In this case, we will make endless queries as the assignment conditions in Lines 11 and 12 in Algorithm 1 will never be satisfied. To fix this, we modify $\hat{p}_l(t)$ to be the highest $l_{ai}(t)$ value that we have seen thus far *across all phases and anchor-item pairs*, and make a corresponding change to $\hat{q}_u(t)$. Appendix C shows that this does not affect our confidence bound. Crucially, the assignment conditions will now be eventually satisfied as $\hat{p}_l(t)$ and $\hat{q}_u(t)$ are not reset in each phase.

It is noteworthy that `QBClusterH` only requires $n$ and $\delta$ as input. If you additionally provide $k$, the number of clusters, it can also work with a more general oracle, as described in the remark below. See Appendix C for the pseudocode of `QBClusterH`.

**Remark 4.3** (A more general oracle)**.** Consider an oracle that is more accurate for some clusters than others. That is, it has $\mathrm{P}(O_{ij} = 1) = p_\ell$ when $i$ and $j$ are in the $\ell^{th}$ cluster and $\mathrm{P}(O_{ij} = 1) = q$ otherwise, where $p_\ell > q$ for all $\ell \in [k]$. When $k$ is known in advance, one can set $\delta^j = \delta/(k-1)$ in `QBClusterH`, and stop after $k-1$ phases, assigning all leftover items to the last cluster. In this case, the boundary con-

dition discussed above is irrelevant, and the calls to `SplitItemsH` can be completely independent, allowing them to find clusters with different $p_\ell$ values with the total probability of error at most $\sum_{j=1}^{k} \delta^j = \delta$. Here, the assumption that $k$ is known is important. As $p_\ell$ can be arbitrarily close to $q$, this extra information is useful to differentiate the case where $p_\ell \approx q$ for the chosen anchor from the case where all remaining items are in the same cluster. One can alternatively assume that $p_\ell - q > \phi$ for all $\ell$, where $\phi$ is known.

In what follows, we focus on the simpler oracle from eq. (1) for which the algorithm need not know $k$. Our guarantees can be trivially adapted to the oracle in Remark 4.3 when $k$ is known.

# 5 ANALYSIS

Suppose we have a sequential hypothesis test $\Psi(\epsilon)$ and paired samples $(x_t, y_t)$ from two Bernoulli random variables $X \sim \text{Ber}(\mu_1)$ and $Y \sim \text{Ber}(\mu_2)$. We say that $\Psi(\epsilon)$ has succeeded if it never terminates when $\mu_1 = \mu_2$, and uses a random but finite number of samples to correctly verify if $\mu_1 > \mu_2$ otherwise. We assume that $\Psi(\epsilon)$ succeeds with probability at least $1-\epsilon$.

We first describe the intuition behind our analysis in terms of Meta-Algorithm 1. Let $\Psi_{ij}$ be an instance of the test $\Psi(\epsilon)$ for the hypothesis $\mathcal{H}_{ij}$. Assume that $\epsilon$ is set so that *all* tests $\Psi_{ij}$ succeed with probability at least $1-\delta$. Under this event, it is easy to see that: **1)** all cluster assignments are correct, and **2)** the algorithm terminates in finite time. This makes it $\delta$-PAC.

For the sample complexity, we focus on tests $\Psi_{ij}$ where items $i$ and $j$ are in different ground-truth clusters. These items will be assigned a cluster as soon as at least one such test succeeds. The key is to find bounds $T_{ij}(p, q, \delta)$ such that all of these tests are guaranteed to succeed after collecting $T_{ij}(p, q, \delta)$ samples with probability at least $1-\delta$. In this case, the algorithm runs for at most $T = \max_{i,j} T_{ij}(p, q, \delta)$ steps, making one query for each item $i \in \mathcal{N} \backslash \{a\}$ in each step. This results in an $O(nT)$ bound on the sample complexity with high probability. Note that the max above is over item pairs in different ground-truth clusters. Next, we apply these ideas to our algorithms.

**Correctness:** In Appendix D.1, we use Lemma 4.1 to show that all tests in `SplitItemsH` succeed with probability $1-\delta$, and use this to argue that the algorithm is $\delta$-PAC. `QBClusterH` adjusts the confidence level in each phase to ensure that *all* calls to `SplitItemsH` succeed with probability $1-\delta$. In this event, the algorithm runs for exactly $k$ phases, where $k$ is the (unknown) number of clusters, and isolates one cluster correctly in each phase. This makes it $\delta$-PAC.

**Theorem 5.1** (Correctness of the algorithms)**.** *Assume that each ground-truth cluster has at least two items. Then,* `SplitItemsH` *and* `QBClusterH` *are $\delta$-PAC for $k = 2$ and $k \geq 2$ clusters, respectively.*

**Sample Complexity:** For `SplitItemsH`, suppose items $i$ and $j$ have $\alpha_{ai} = p$ and $\alpha_{aj} = q$ for the chosen anchor $a$. We use Lemma 4.1 in Appendix D.2 to show that $\Psi_{ij}$ succeeds with high probability after collecting at most $T_{ij}(p, q, \delta) = O\left(\frac{1}{(p-q)^2} \ln \frac{n^2}{\delta(p-q)^2}\right)$ samples, resulting in the following bound.

**Theorem 5.2** (Sample complexity of `SplitItemsH`)**.** *Let $\nu = (n, \rho, 2, p, q)$ be an arbitrary* `QBC` *instance with two clusters, each with at least two items. The number of queries made by* `SplitItemsH` *is bounded by $O\left(\frac{n}{(p-q)^2} \ln \frac{n^2}{\delta(p-q)^2}\right)$ with probability at least $1-\delta$.*

For `QBClusterH`, we again focus on the event where all calls to `SplitItemsH` have succeeded. This happens with probability at least $1-\delta$. Moreover, the algorithm runs for exactly $k$ phases under this event, one for each cluster. The sample complexity of `QBClusterH` is then the sum of the sample complexity of each phase. In the worst case, smaller clusters are eliminated first, and each phase still sees roughly $O(n)$ items. Using the result from Theorem 5.2 and noting that the smallest confidence level is the one from the last phase, we get the following worst-case sample complexity.

**Theorem 5.3** (Sample complexity of `QBClusterH`)**.** *Let $\nu = (n, \rho, k, p, q)$ be an arbitrary* `QBC` *instance with $k \geq 2$ clusters, each with at least two items. The worst-case sample complexity of* `QBClusterH` *is bounded by $O\left(\frac{nk}{(p-q)^2} \ln \frac{n^2k^2}{\delta(p-q)^2}\right)$ with probability at least $1-\delta$.*

In Appendix D.4, we derive a more general bound for `QBClusterH` that explicitly depends on the size of each cluster. The next two remarks discuss these results in the context of our lower bound.

**Remark 5.1** (Optimality of `SplitItemsH`)**.** For fixed values of $p$ and $q$, the bound in Theorem 5.2 matches our lower bound up to logarithmic factors. However, when $p$ and $q$ also depend on $n$, (as in Remark 3.1), $d(p, q) - 2(p-q)^2$ becomes an $n$ dependent positive quantity by Pinsker's inequality[1], widening the gap between the two bounds. We empirically validate this in our experiments and show that using a more efficient hypothesis test in Meta-Algorithm 1 fixes this issue.

[1]Pinsker's inequality states $\sqrt{\frac{1}{2} d(p, q)} \geq |p - q|$ in this context. The quantity on the right is the total variation distance between two Bernoulli distributions with success probability $p$ and $q$, respectively.

**Remark 5.2** (Optimality of `QBClusterH`)**.** Suppose $d(p,q) < d(q,p)/(k-1)$. This, for example, happens when $q = 0.5$ and $p$ is close to 1. In this case, for fixed $p$ and $q$, the bound in Theorem 5.3 matches our lower bound up to logarithmic factors. This means that our bounds are nearly tight in the worst-case sense.

As in Remark 5.1, a more efficient hypothesis test improves the sample complexity of `QBClusterH` when $p$ and $q$ vary with $n$. We demonstrate this in Section 6.

Also interesting is the case where $d(p,q) \geq d(q,p)/(k-1)$. Here, the bound in Theorem 5.3 differs from our lower bound by a multiplicative $k$ factor. We believe that Theorem 3.1 can be strengthened for these values of $p$ and $q$. After all, even in the noise free setting, each item needs at least one query per cluster in the worst case, leading to a $\Omega(nk)$ queries. The noisy case should only be worse.

# 6 EXPERIMENTS

This section has three objectives: **1)** empirically validate Theorems 5.2 and 5.3, **2)** show that a more efficient hypothesis test reduces the gap between our upper and lower bounds when $p$ and $q$ vary with $n$, and **3)** demonstrate the robustness of our algorithms to a simple model misspecification. All our experiments use $\delta = 0.05$. The error bars in the plots correspond to standard deviation[2].

**Validating Theorems 5.2 and 5.3:** We generate `QBC` instances by taking $n$ items and randomly dividing them into $k$ clusters, each with at least two items. We then fix $p = 0.2$ and $q = 0.1$, and simulate our oracle via eq. (1). This oracle is wrong 80% of the time in answering queries about items from the same cluster.

Let us first focus on the plots related to `SplitItemsH` and `QBClusterH` in Figure 1. `SplitItemsS` and `QBClusterS` are different algorithms that will be introduced in the next part of this section. In Figure 1a, we fix $k = 2$ and show that the sample complexity of `SplitItemsH` varies as $O(n \ln n)$. `QBClusterH` exhibits a similar pattern in Figure 1b when $k = 4$. The $O(k \ln k)$ variation for a fixed $n = 80$ is demonstrated in Figure 1c. Together, these observations empirically validate Theorems 5.2 and 5.3.

**Using a more efficient test:** The confidence intervals in Lemma 4.1 use Hoeffding's inequality. As a consequence, they approximately shrink as $O(1/\sqrt{t})$, which explains why the sample complexities in Theorems 5.2 and 5.3 vary as $1/(p-q)^2$. We argued in Remark 5.1 that this dependence on $p$ and $q$ can be improved. One way to achieve this is discussed below.

Recall that Meta-Algorithm 1 solves several BAI problems by constructing confidence intervals for arm means using Lemma 4.1 (see Remark 4.1). While these intervals are easy to understand and analyze, they are not the most sample efficient. We develop two new algorithms, called `SplitItemsS` and `QBClusterS`, that are identical to the corresponding algorithms from Section 4, except that they use better confidence intervals borrowed from Kaufmann et al. (2016).

More specifically, let $\psi(t,\epsilon) = \ln \frac{t(\ln(3t))^2}{\epsilon}$, and define

$$\begin{aligned} l_{ai}(t) &= \inf\{z < \hat{\mu}_{ai}(t) : td(\hat{\mu}_{ai}(t), z) \leq \psi(2t,\epsilon)\}, \\ u_{ai}(t) &= \sup\{z > \hat{\mu}_{ai}(t) : td(\hat{\mu}_{ai}(t), z) \leq \psi(2t,\epsilon)\}, \end{aligned} \tag{4}$$

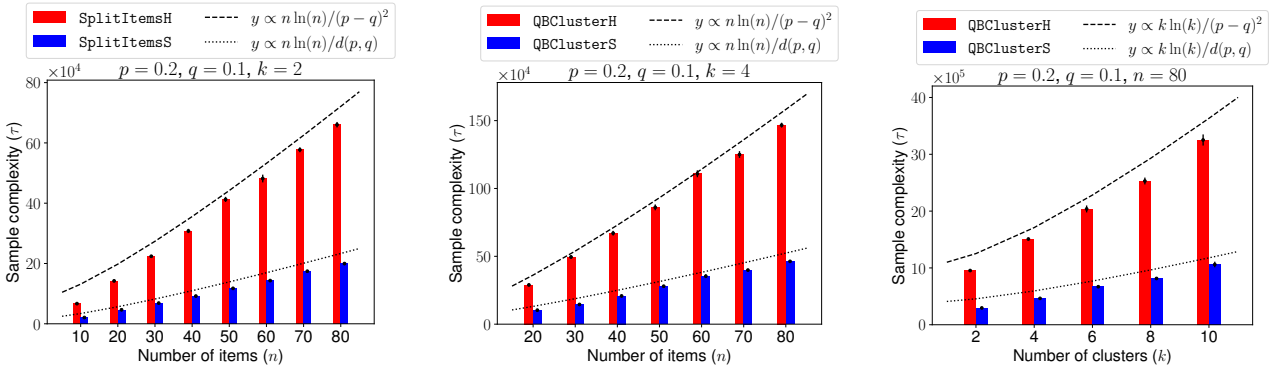
where $\hat{\mu}_{ai}(t)$ is the empirical mean for item pair $\{a,i\}$ defined in Lemma 4.1. Kaufmann et al. (2016) showed that these confidence intervals are asymptotically optimal. Informally speaking, what this means in our context is that the sample complexity of a test based on these intervals approximately varies with $d(p,q)$ instead of $(p-q)^2$. In Appendix E, we use a result from Kaufmann and Kalyanakrishnan (2013) to argue that an analog of Lemma 4.1 is valid for these confidence intervals when $\epsilon = \delta/|\mathcal{N}|^2$. We also make a more precise statement about their optimality and provide the pseudocode for the new algorithms.

Next, we want to verify if these new intervals actually help. To do so, we simulate our oracle as before, but use $p = 2\frac{\ln n}{n}$ and $q = \frac{\ln n}{n}$. As $p$ and $q$ now depend on $n$, so does the ratio $d(p,q)/(p-q)^2$, which introduces a gap between our upper and lower bounds (see Remark 5.1). In particular, one can easily use these values of $p$ and $q$ in Theorem 5.2 to verify that the sample complexity of `SplitItemsH` should increase as $O(n^3/\ln n)$ in this case. Figure 2 confirms this prediction. More importantly, it shows that `SplitItemsS` does not suffer from the same issue. Its sample complexity grows as $\tilde{O}\left(n/d(p,q)\right)$, which matches our lower bound up to logarithmic factors, even when $p$ and $q$ vary with $n$.

It is important to emphasize that this gap between our bounds in relevant only when $p$ and $q$ are not constants. Figure 1 shows that `SplitItemsS` and `QBClusterS` perform better than `SplitItemsH` and `QBClusterH` even for fixed $p$ and $q$, but only by a constant multiplicative factor. The order of sample complexity changes only when $p$ and $q$ vary with $n$.

**Model misspecification:** In this part, we study the consequences of having an oracle that does not behave as expected. Let $\eta \geq 0$ be a perturbation param-

[2]Code available at: `https://github.com/IBM/aistats-24-inconsistent-feedback-clustering`




(a) $\tau$ vs $n$, $k = 2$ clusters (b) $\tau$ vs $n$, $k = 4$ clusters (c) $\tau$ vs $k$, $n = 80$ items

Figure 1: Variation in the sample complexity of our algorithms with respect to the number of items ($n$) and the number of clusters ($k$). These results empirically validate our upper bounds in Theorem 5.2 and Theorem 5.3.

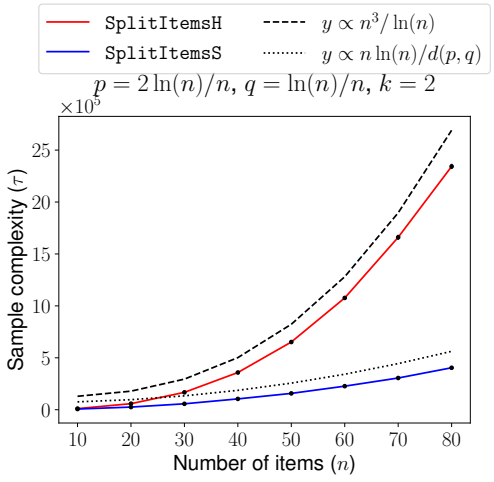



Figure 2: Sample complexity of `SplitItemsH` and `SplitItemsS` when $p$ and $q$ vary with $n$.

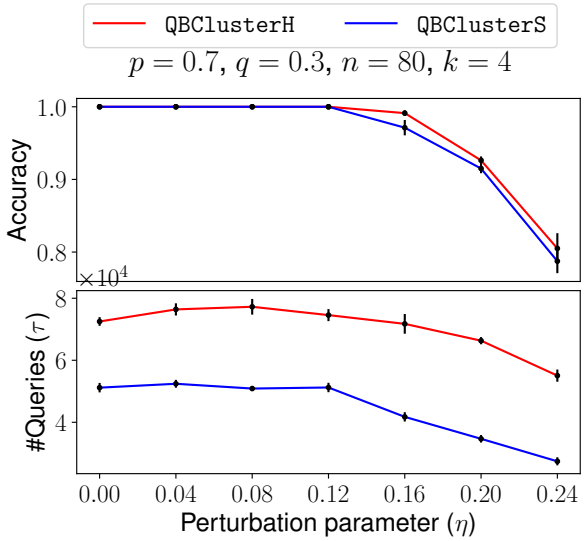



Figure 3: Variation in the performance of `QBClusterH` and `QBClusterS` vs the perturbation in the oracle.

eter. We use it to generate a random perturbation $\delta_{ij} \sim \text{Uniform}[-\eta, +\eta]$ for each pair of items. Instead of using the feedback model in eq. (1), we now simulate an oracle that uses the following feedback model.

$$\mathrm{P}\left(O_{ij} = 1\right) = \begin{cases} p + \delta_{ij} & \text{if } \rho(i) = \rho(j) \\ q + \delta_{ij} & \text{otherwise.} \end{cases} \tag{5}$$

As before, $\rho : [n] \to [k]$ encodes the ground-truth clusters. This model reduces to the one in eq. (1) when $\eta = 0$. In other cases, it uses different probability values across item pairs. We set $p = 0.7$ and $q = 0.3$.

Figure 3 shows the accuracy and query complexity of our algorithms as a function of the robustness parameter $\eta$. *First*, consider the accuracy plot and notice that the algorithms make no mistakes for $\eta \leq 0.1$. To see why this happens, note that the probability values used by the oracle fall in one of the two clusters, one centered around $p$ (the $p$-cluster) and the other centered around $q$ (the $q$-cluster). Let $p_{\min}$ and $p_{\max}$ be the smallest and largest values in the $p$-cluster. Analogously define $q_{\min}$ and $q_{\max}$ for the $q$-cluster. If $p_{\min} - q_{\max} < p_{\max} - p_{\min}$, the confidence intervals for $p_{\min}$ and $p_{\max}$ may stop overlapping before the intervals for $p_{\min}$ and $q_{\max}$ do so, which leads to an incorrect cluster assignment. However, as $p = 0.7$ and $q = 0.3$, this will never happen with high probability unless $\eta > 0.1$. Even for $\eta > 0.1$, the accuracy of the algorithms degrades gracefully, showing their robustness to this type of model misspecification. The $p$ and $q$ cluster actually overlap for $\eta > 0.2$, but, even then, we get more than 75% accuracy.

*Next*, consider the sample complexity. As $q_{\max}$ and $p_{\min}$ get closer, the algorithm needs more queries to resolve these values. This accounts for the increase in complexity as $\eta$ goes from 0 to 0.1. After this point, as discussed above, it gets easier to resolve $p_{\max}$ from $p_{\min}$ than it is to resolve $q_{\max}$ from $p_{\min}$, which leads to fewer queries, albeit at the cost of more mistakes.

*Finally*, it is important to highlight the *non-systematic* nature of the perturbations introduced above: they are independently selected from a uniform distribution for each item pair. One may encounter more *systematic* perturbations in practice which may lead to a less-than-elegant decline in performance. For example, if the ground truth clusters are hierarchical, sub-clusters will have increasing $\mathrm{P}(O_{ij} = 1)$ values as one gets deeper in the hierarchy. In this case, $\mathrm{P}(O_{ij} = 1)$ values are selected from different "levels" based on the distance between items $i$ and $j$ in the hierarchy. Running our algorithms in such scenarios would almost certainly fail since the cluster with the anchor need not be a *pure* cluster. Adapting the key ideas from our algorithms to such problems is an interesting direction for future work (Emamjomeh-Zadeh and Kempe, 2018; Ghoshdastidar et al., 2019).

# 7 CONCLUSION

This paper studies the problem of clustering items by querying a noisy and inconsistent oracle. We developed efficient algorithms using sequential hypothesis testing, and derived high-probability upper bounds on their sample complexity. We showed that these bounds match an information-theoretic lower bound up to logarithmic factors in many interesting cases. We also empirically demonstrated a way to close this gap and highlighted the robustness of our algorithms. This still leaves many avenues for improvement. Our lower bound can be strengthened for some values of $p$ and $q$ when $k > 2$ (see Remark 5.2). It would also be interesting to study oracles based on more realistic models like the Stochastic Block Model (Holland et al., 1983) or other type of clustering problems like hierarchical clustering (Emamjomeh-Zadeh and Kempe, 2018).

### Acknowledgements

We would like to thank Michele Dolfi for his help in making our code publicly available and Yagmur Gizem Cinar for her valuable feedback on an initial version of this paper.

### References

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

Emmanuel Abbe and Colin Sandon. Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 670–688, 2015.

Emmanuel Abbe, Jianqing Fan, Kaizheng Wang, and Yiqiao Zhong. Entrywise eigenvector analysis of random matrices with low expected rank. *The Annals of Statistics*, 48(3):1452–1474, 2020.

Nir Ailon, Anup Bhattacharya, and Ragesh Jaiswal. Approximate correlation clustering using same-cluster queries. *LATIN 2018: Theoretical Informatics*, pages 14–27, 2018a.

Nir Ailon, Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. Approximate clustering with same-cluster queries. *In Proceedings of the 9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, 94:40:1–40:21, 2018b.

Hassan Ashtiani and Shai Ben-David. Representation learning for clustering: A statistical framework. *In Proceedings of Uncertainty in Artificial (UAI)*, 2015.

Hassan Ashtiani, Shrinu Kushagra, and Shai Ben-David. Clustering with same-cluster queries. *Advances in Neural Information Processing Systems*, 29, 2016.

Pranjal Awasthi, Maria Florina Balcan, and Konstantin Voevodski. Local algorithms for interactive clustering. *Journal of Machine Learning Research*, 18(3):1–35, 2017.

Maria-Florina Balcan and Avrim Blum. Clustering with interactive feedback. *Algorithmic Learning Theory, Lecture Notes in Computer Science*, 5254, 2008.

Marco Bressan, Nicolò Cesa-Bianchi, Silvio Lattanzi, and Andrea Paudice. Exact recovery of mangled clusters with same-cluster queries. *Advances in Neural Information Processing Systems*, 33, 2020a.

Marco Bressan, Nicolò Cesa-Bianchi, Andrea Paudice, and Fabio Vitale. Correlation clustering with adaptive similarity queries. *Advances in Neural Information Processing Systems*, 32, 2020b.

Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *Algorithmic Learning Theory*, pages 23–37. Springer Berlin Heidelberg, 2009.

Serban D. Constantin and T.R.N. Rao. On the theory of binary asymmetric error correcting codes. *Information and Control*, 40(1):20–36, 1979.

Susan Davidson, Sanjeev Khanna, Tova Milo, and Sudeepa Roy. Top-k and clustering with noisy com-

parisons. *ACM Transactions on Database Systems*, 39(4):1–39, 2014.

Ehsan Emamjomeh-Zadeh and David Kempe. Adaptive hierarchical clustering using ordinal queries. *Proceedings of the 2018 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 415–429, 2018.

Donatella Firmani, Barna Saha, and Divesh Srivastava. Online entity resolution using an oracle. *In Proceedings of the VLDB Endowment*, 9(5):384–395, 2016.

Akshay Gadde, Eyal En Gad, Salman Avestimehr, and Antonio Ortega. Active learning for community detection in stochastic block models. *In Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT)*, pages 1889–1893, 2016.

Buddhima Gamlath, Sangxia Huang, and Ola Svensson. Semi-supervised algorithms for approximately optimal and accurate clustering. *In Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, 107: 57:1–57:14, 2018.

Claudio Gentile, Shuai Li, and Giovanni Zappella. Online clustering of bandits. *In Proceedings of the 31st International Conference on Machine Learning*, 32 (2):757–765, 2014.

Debarghya Ghoshdastidar, Michaël Perrot, and Ulrike von Luxburg. Foundations of comparison-based hierarchical clustering. *Advances in Neural Information Processing Systems*, 32, 2019.

Paul Holland, Kathryn B. Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5:109–137, 1983.

Emilie Kaufmann and Shivaram Kalyanakrishnan. Information complexity in bandit subset selection. *In Proceedings of the 26th Annual Conference on Learning Theory*, 30:228–251, 2013.

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

Taewan Kim and Joydeep Ghosh. Semi-supervised active clustering with weak oracles. *arXiv preprint*, arxiv/1709.03202, 2017.

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

Kasper Green Larsen, Michael Mitzenmacher, and Charalampos E. Tsourakakis. Clustering with a faulty oracle. *In Proceedings of The Web Conference 2020*, pages 2831–2834, 2020.

Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

Yi Li, Yan Song, and Qin Zhang. Learning to cluster via same-cluster qeries. *In Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 978–987, 2021.

Yucen Luo, Tian Tian, Jiaxin Shi, Jun Zhu, and Bo Zhang. Semi-crowdsourced clustering with deep generative models. *Advances in Neural Information Processing Systems*, 31, 2018.

James MacQueen. Some methods for classification and analysis of multivariate observations. *In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 5.1: Statistics:281–297, 1967.

Arya Mazumdar and Barna Saha. Clustering with noisy queries. *Advances in Neural Information Processing Systems*, 30, 2017a.

Arya Mazumdar and Barna Saha. Query complexity of clustering with side information. *Advances in Neural Information Processing Systems*, 30, 2017b.

Michael Mitzenmacher and Charalampos E. Tsourakakis. Predicting signed edges with $o(n^{1+o(1)} \log n)$ queries. *arXiv preprint*, arxiv/1609.00750, 2016.

Michael Mitzenmacher and Charalampos E. Tsourakakis. Joint alignment from pairwise differences with a noisy oracle. *Internet Mathematics*, 2020.

Pan Peng and Jiapeng Zhang. Towards a query-optimal and time-efficient algorithm for clustering with a faulty oracle. *In Proceedings of 34th Annual Conference on Learning Theory*, 134:3662–3680, 2021.

Alberto Del Pia, Mingchen Ma, and Christos Tzamos. Clustering with queries under semi-random noise. *In Proceedings of the 35th Annual Conference on Learning Theory*, 178:1–36, 2022.

Barna Saha and Sanjay Subramanian. Correlation clustering with same-cluster queries bounded by optimal cost. *In Proceedings of the 27th Annual European Symposium on Algorithms (ESA 2019)*, 144: 81:1–81:17, 2019.

Preeti Saryan, Shubham Gupta, and Vinita Gowda. Species complex delimitations in the genus hedychium: A machine learning approach for cluster discovery. *Appl Plant Sci*, 8(7):e11377, 2020.

David Siegmund. *Sequential Analysis*. Springer-Verlag, 1985.

Nathaniel R. Twarog, Michele Connelly, and Anang A. Shelat. A critical evaluation of methods to interpret

drug combinations. *Scientific Reports*, 10(5144), 2020.

Jennifer Wortman Vaughan. Making better use of the crowd: How crowdsourcing can advance machine learning research. *Journal of Machine Learning Research*, 18:1–46, 2018.

Jinghui Xia and Zengfeng Huang. Optimal clustering with noisy queries via multi-armed bandit. *In Proceedings of the 39th International Conference on Machine Learning*, 162:24315–24331, 2022.

Junwen Yang, Zixin Zhong, and Vincent Y. F. Tan. Optimal clustering with bandit feedback. *arxiv preprint*, arXiv/2202.04294, 2022.

Se-Young Yun and Alexandre Proutiere. Community detection via random and adaptive sampling. *In proceedings of the 27th Conference on Learning Theory*, 35:138–175, 2014.

## Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes. See Section 2 for problem setting, Section 4 for algorithms, Theorem 3.1 for a lower bound, and Theorems 5.2 and 5.3 for the upper bound.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes. We analyze the sample complexity of our algorithms in Theorems 5.2 and 5.3. Each step of our algorithms involves simple calculations for $O(n)$ items, and hence has complexity $O(n)$.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes. Our code is available on Github.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes. See Theorem 3.1 for a lower bound and Theorems 5.2 and 5.3 for the upper bound.
   (b) Complete proofs of all theoretical results. Yes. All proofs are included in the appendices.
   (c) Clear explanations of any assumptions. Yes. The assumptions are explained when they are used in the proofs.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes. Our code is available for reproducing all the results.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes. Section 6 includes all details needed for reproduction.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes, see Section 6
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). No. All our experiments were run on a MacBook Pro (2021). We did not use any other compute resources.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Not Applicable
   (b) The license information of the assets, if applicable. Not Applicable
   (c) New assets either in the supplemental material or as a URL, if applicable. Not Applicable
   (d) Information about consent from data providers/curators. Not Applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable

# Clustering Items From Adaptively Collected Inconsistent Feedback: Supplementary Material

## A AN ALTERNATIVE MOTIVATION FOR OUR ORACLE

One can reasonably argue that our feedback model in eq. (1) is too simplistic for many practical applications. For example, in distance-based clustering, items are often represented by vectors, which makes some items "more similar" than others. The error probability of the oracle should ideally be a function of the similarity between the queried items in this case. In Section 6, we experiment with such an oracle. We show that our algorithms work well, even when $p$ and $q$ values vary across item pairs, as long as these values are "well separated" between same- and different-cluster pairs. In this section, we argue that eq. (1) may itself be useful in some applications, especially when the oracle is perfect.

To see this, suppose that the oracle makes no errors: given a pair of items, it can always correctly tell if they are in the same cluster. Instead of the noise being present in the oracle, this time the noise comes from the communication channel that the learner uses to receive feedback. In particular, eq. (1) corresponds to the case of a Binary Asymmetric Channel (BAC) (Constantin and Rao, 1979). BACs are characterized by having different error probabilities for $0 \to 1$ and $1 \to 0$ type errors, and have been extensively studied in the literature. In our context, when the oracle sends a 1 (items are in the same cluster), the channel makes a $1 \to 0$ error with probability $1-p$, transmitting a 1 only with probability $p$. Similarly, it makes a $0 \to 1$ error with probability $q$, transforming a 0 sent by the oracle (items in different cluster) into a 1 with probability $q$. As before, the goal is to determine when enough evidence has been collected to correctly cluster the items without knowing the exact properties of the channel (i.e., the value of $p$ and $q$).

Perfect oracles are useful in many practical applications. They expose the clusters to the learners while hiding the details of how these clusters were obtained. For example, an online marketplace might want to expose market segments to an advertizing agency, while hiding fine-grained private information about the users' purchasing behavior that was used to arrive at these market segments. The advertizing agency pays for each query in this case. It is useful to simulate a perfect oracle in such privacy-aware applications, even if the underlying items exhibit a more intricate relationship.

## B ADDITIONAL DETAILS REGARDING THE LOWER BOUND

Let $\mathcal{V}$ be a family of `QBC` instances that share the same value of $n$, $p$, and $q$ but have non-matching cluster assignments. For any instance $\nu \in \mathcal{V}$ with cluster assignment $\rho$, define $C_{ij} \coloneqq \mathbf{I}\{\rho(i) = \rho(j)\}$. Further, let $\Delta \coloneqq \{\mathbf{W} \in [0,1]^{n \times n} : \forall i,j,\ \ W_{ij} = W_{ji} \ \text{ and } \ \sum_{i>j} W_{ij} = 1\}$ and $\tilde{W}_{ij} \coloneqq W_{ij}(C_{ij} - C'_{ij})\,[C_{ij}d(p,q) - (1-C_{ij})d(q,p)]$, where $d(x,y) \coloneqq x\ln(x/y) + (1-x)\ln((1-x)/(1-y))$ is the binary relative entropy function. As discussed in Section 3, the following lemma derives an expression for the expected log-likelihood ratio, and provides an upper bound on this quantity.

**Lemma B.1.** *Let $\mathcal{V}$ be a family of `QBC` instances that share the same value of $n$, $p$, and $q$ but have non-matching cluster assignments. Let $\mathcal{F}(t)$ be the $\sigma$-algebra generated by the item pair selections made by the algorithm and the feedback returned by the oracle up to and including time $t$.*

1. *For any $\nu \neq \nu' \in \mathcal{V}$ and almost-surely finite stopping time $\tau$ with respect to $(\mathcal{F}(t))_t$, the expected* log*-likelihood ratio under instance $\nu$ is given by*

$$\mathrm{E}_\nu[L_{\nu,\nu'}(\tau)] = \sum_{i>j} \mathrm{E}_\nu[T_{ij}(\tau)](C_{ij} - C'_{ij})\,[C_{ij}d(p,q) - (1-C_{ij})d(q,p)]\,,$$

*where, $T_{ij}(t) = \sum_{s=1}^{t} \mathbf{I}\{\text{Pair } \{i,j\} \text{ was sampled at time } s\}$, $C_{ij} = \mathbf{I}\{\rho(i) = \rho(j)\}$, $C'_{ij} = \mathbf{I}\{\rho'(i) = \rho'(j)\}$, and $d(x,y) \coloneqq x\ln(x/y) + (1-x)\ln((1-x)/(1-y))$ is the binary relative entropy function.*

*2. Let* $\Delta \coloneqq \{\mathbf{W} \in [0,1]^{n\times n} : \forall i,j,\ \ W_{ij} = W_{ji} \ \ \textit{and} \ \ \sum_{i>j} W_{ij} = 1\}$. *The following holds*

$$\inf_{\nu' \neq \nu \in \mathcal{V}} \mathrm{E}_\nu[L_{\nu,\nu'}(\tau)] \leq \mathrm{E}_\nu[\tau] \ \ \sup_{\mathbf{W}\in\Delta} \inf_{\nu' \neq \nu \in \mathcal{V}} \sum_{i>j} W_{ij}(C_{ij} - C_{ij}^{'}) \left[C_{ij} d(p,q) - (1-C_{ij}) d(q,p)\right]. \tag{6}$$

*Proof.* Let $\nu, \nu' \in \mathcal{V}$ be two `QBC` instances with cluster assignment functions $\rho$ and $\rho'$ respectively. The log-likelihood ratio $L_{\nu,\nu'}(t)$ of observations $(O(s))_{s\leq t}$ under instances $\nu$ and $\nu'$ is defined as

$$L_{\nu,\nu'}(t) = \ln \frac{\Pi_{s=1}^{t} \mathrm{P}_\nu(O(s)|i(s),j(s))}{\Pi_{s=1}^{t} \mathrm{P}_{\nu'}(O(s)|i(s),j(s))},$$

where $\{i(s), j(s)\}$ is the item pair queried by the algorithm at time $s$. Because $\mathrm{P}_\nu(O(s)|i(s),j(s)) = \left(p^{O(s)}(1-p)^{1-O(s)}\right)^{\mathbf{I}\{\rho(i(s))=\rho(j(s))\}} \left(q^{O(s)}(1-q)^{1-O(s)}\right)^{\mathbf{I}\{\rho(i(s))\neq\rho(j(s))\}}$ for any instance $\nu$, the expression above can be written as

$$L_{\nu,\nu'}(t) = \sum_{i>j} \sum_{s=1}^{t} \mathbf{I}\{\{i(s),j(s)\} = \{i,j\}\}(C_{ij} - C_{ij}^{'}) \left[O(s) \ln \frac{p}{q} + (1-O(s)) \ln \frac{1-p}{1-q}\right]. \tag{7}$$

Define a sequence of $i.i.d.$ random samples $(Y_{ij}(s))_{s\geq 1}$ for all item pairs $i > j$, obtained after successively probing these pairs. The log-likelihood ratio can be equivalently written as

$$L_{\nu,\nu'}(t) = \sum_{i>j} \sum_{s=1}^{T_{ij}(t)} (C_{ij} - C_{ij}^{'}) \left[Y_{ij}(s) \ln \frac{p}{q} + (1-Y_{ij}(s)) \ln \frac{1-p}{1-q}\right].$$

Under problem instance $\nu$, we have $\mathrm{P}(Y_{ij}(s) = 1) = C_{ij}p + (1-C_{ij})q$, and hence the expected log-likelihood is given by

$$\mathrm{E}_\nu\left[L_{\nu,\nu'}(t)\right] = \sum_{i>j} T_{ij}(t)(C_{ij} - C_{ij}^{'}) \left[C_{ij} d(p,q) - (1-C_{ij}) d(q,p)\right].$$

By Wald's lemma (Siegmund, 1985), for a random stopping time $\tau$ with respect to $(\mathcal{F}(t))_t$, for any $\delta$-PAC algorithm for which $\mathrm{P}(\tau < +\infty) = 1$, we get

$$\mathrm{E}_\nu\left[L_{\nu,\nu'}(\tau)\right] = \sum_{i>j} \mathrm{E}_\nu\left[T_{ij}(\tau)\right](C_{ij} - C_{ij}^{'}) \left[C_{ij} d(p,q) - (1-C_{ij}) d(q,p)\right].$$

This proves the first part of the lemma. For the second part, note that

$$\begin{aligned}
\inf_{\nu' \neq \nu \in \mathcal{V}} \mathrm{E}_\nu\left[L_{\nu,\nu'}(\tau)\right] &= \inf_{\nu' \neq \nu \in \mathcal{V}} \sum_{i>j} \mathrm{E}_\nu\left[T_{ij}(\tau)\right](C_{ij} - C_{ij}^{'}) \left[C_{ij} d(p,q) - (1-C_{ij}) d(q,p)\right] \\
&= \mathrm{E}_\nu\left[\tau\right] \inf_{\nu' \neq \nu \in \mathcal{V}} \sum_{i>j} \frac{\mathrm{E}_\nu\left[T_{ij}(\tau)\right]}{\mathrm{E}_\nu\left[\tau\right]} (C_{ij} - C_{ij}^{'}) \left[C_{ij} d(p,q) - (1-C_{ij}) d(q,p)\right] \\
&\leq \mathrm{E}_\nu\left[\tau\right] \sup_{\mathbf{W}\in\Delta} \inf_{\nu' \neq \nu \in \mathcal{V}} \sum_{i>j} W_{ij}(C_{ij} - C_{ij}^{'}) \left[C_{ij} d(p,q) - (1-C_{ij}) d(q,p)\right].
\end{aligned}$$

The last inequality uses the fact that the total sample complexity $\mathrm{E}_\nu[\tau] = \sum_{i>j} \mathrm{E}_\nu[T_{ij}(\tau)]$ is the sum of the number of times each item pair is pulled. This finishes the proof of the second part. □

In the bound above, $\tau$ is a random *stopping time* that depends only on the observations collected in the past. As we make no assumption on how $\tau$ is calculated as long as it depends only on the past observations, the inequality above is valid for any $\delta$-PAC algorithm. We analogously show a lower bound on $\inf_{\nu' \neq \nu \in \mathcal{V}} \mathrm{E}_\nu[L_{\nu,\nu'}(\tau)]$ that is satisfied by any $\delta$-PAC algorithm. This bound is a simple consequence of Lemma 1 from Kaufmann et al. (2016).

**Lemma B.2.** *With the same notation as Lemma B.1, any $\delta$-PAC algorithm satisfies the following for any instance $\nu \in \mathcal{V}$*

$$\inf_{\nu' \neq \nu \in \mathcal{V}} \mathrm{E}_\nu[L_{\nu,\nu'}(\tau)] \geq \ln \frac{1}{2.4\delta}.$$

*Proof.* The following lemma is equivalent to Lemma 19 from Kaufmann et al. (2016), and can be derived in exactly the same way by using the definition of $L_{\nu,\nu'}(t)$ from eq. (7).

**Lemma B.3** (Lemma 19 from Kaufmann et al. (2016))**.** *With the same notation as Lemma B.1, for every event $\mathcal{E} \in \mathcal{F}(\tau)$ and instances $\nu \neq \nu' \in \mathcal{V}$, we have* $\mathrm{E}_\nu[L_{\nu,\nu'}(\tau)] \geq d(\mathrm{P}_\nu(\mathcal{E}), d(\mathrm{P}_{\nu'}(\mathcal{E})))$.

Pick an arbitrary instance $\nu \in \mathcal{V}$. Let $\mathcal{E} = \{\hat{\rho} = \rho\}$ be the event that the cluster estimate $\hat{\rho}$ returned by a $\delta$-PAC algorithm upon termination matches the cluster assignment $\rho$ in instance $\nu$. Clearly, $\mathcal{E} \in \mathcal{F}(\tau)$. Moreover, by Definition 2.2, $\mathrm{P}_\nu(\mathcal{E}) \geq 1 - \delta$ when the observations are drawn from instance $\nu$ and $\mathrm{P}_{\nu'}(\mathcal{E}) < \delta$ when the observations are drawn from an arbitrary instance $\nu' \neq \nu$ with a different cluster assignment function $\rho'$. For this event $\mathcal{E}$,

$$d(\mathrm{P}_\nu(\mathcal{E}), \mathrm{P}_{\nu'}(\mathcal{E})) \geq d(1-\delta, \delta) \geq \ln \frac{1}{2.4\delta},$$

where the last inequality can be easily verified. Using this event in Lemma B.3 gives

$$\mathrm{E}_\nu[L_{\nu,\nu'}(\tau)] \geq \frac{1}{2.4\delta}.$$

As the above is true for an arbitrary instance $\nu' \neq \nu$, taking an infimum over all such $\nu'$ on the left hand side yields the desired result.

□

We are now ready to prove our main result on the lower bound in Theorem 3.1.

Combining lemmas B.1 and B.2 gives a lower bound on $\mathrm{E}_\nu[\tau]$ in terms of $\delta$, $p$, and $q$. The expression on the right hand side of eq. (6) is a bit cumbersome. We simplify it in proof of Theorem 3.1 in the next section by restricting the infimum to a smaller class of "challenging" alternative instances $\nu'$ that differ from the instance $\nu$ in the cluster assignment of only one item.

### B.1 Proof of Theorem 3.1

*Proof.* Fix an arbitrary instance $\nu = (n, \rho, k, p, q)$ from $\mathcal{V}$. For any $\delta$-PAC algorithm, combining Lemma B.1 with Lemma B.2 results in

$$\mathrm{E}_\nu[\tau] \geq \frac{1}{\sup_{\mathbf{W}\in\Delta} \inf_{\nu'\neq\nu\in\mathcal{V}} \sum_{i>j} W_{ij}(C_{ij} - C'_{ij})\left[C_{ij}d(p,q) - (1-C_{ij})d(q,p)\right]} \ln \frac{1}{2.4\delta}. \tag{8}$$

While this is a valid lower bound, the quantity on the right hand side of the equation above is not easy to compute. We constrain the infimum in the denominator further to make the calculations easier while still preserving the inequality. Towards this end, define $\bar{\mathcal{V}}_\nu \subseteq \mathcal{V}$ to be a class of instances that agree with $\nu$ on the cluster assignment of all but one items while still having $k$ clusters.

$$\bar{\mathcal{V}}_\nu \coloneqq \left\{ \nu^\ell_m = (n, \rho^\ell_m, k, p, q) \in \mathcal{V} : \rho^\ell_m(i) = \begin{cases} \ell & \text{if } i = m \\ \rho(i) & \text{otherwise.} \end{cases}, \ m \in [n],\ \ell \in [k]\backslash\{\rho(m)\} \right\}$$

Clearly, all instances in $\bar{\mathcal{V}}_\nu$ have a different cluster assignment function compared to $\nu$ as item $m$ is transferred from its original cluster $\rho(m)$ in $\nu$ to the $\ell^{th}$ cluster in $\nu^\ell_m$, where $\ell \neq \rho(m)$. There are still $k$ clusters in $\nu^\ell_m$ because we assume that each cluster has at least two items. For any function $f(\mathbf{W}, \nu, \nu')$,

$$\sup_{\mathbf{W}\in\Delta} \inf_{\nu'\neq\nu\in\mathcal{V}} f(\mathbf{W}, \nu, \nu') \leq \sup_{\mathbf{W}\in\Delta} \inf_{\nu'\in\bar{\mathcal{V}}_\nu} f(\mathbf{W}, \nu, \nu'). \tag{9}$$

To see this, consider any solution $\mathbf{W}^*$ of the optimization problem on the left hand side. For this $\mathbf{W}^*$,

$$\sup_{\mathbf{W}\in\Delta} \inf_{\nu'\neq\nu\in\mathcal{V}} f(\mathbf{W}, \nu, \nu') = \inf_{\nu'\neq\nu\in\mathcal{V}} f(\mathbf{W}^*, \nu, \nu') \leq \inf_{\nu'\in\bar{\mathcal{V}}_\nu} f(\mathbf{W}^*, \nu, \nu') \leq \sup_{\mathbf{W}\in\Delta} \inf_{\nu'\in\bar{\mathcal{V}}_\nu} f(\mathbf{W}, \nu, \nu').$$

Applying eq. (9) to $f(\mathbf{W}, \nu, \nu') = \sum_{i>j} W_{ij}(C_{ij} - C'_{ij})\left[C_{ij}d(p,q) - (1-C_{ij})d(q,p)\right]$ in eq. (8) gives

$$\mathrm{E}_\nu[\tau] \geq \frac{1}{\sup_{\mathbf{W}\in\Delta} \inf_{\nu'\in\bar{\mathcal{V}}_\nu} \sum_{i>j} W_{ij}(C_{ij} - C'_{ij})\left[C_{ij}d(p,q) - (1-C_{ij})d(q,p)\right]} \ln \frac{1}{2.4\delta}. \tag{10}$$

Let $C_{ij}^{m,\ell} = \mathbf{I}\{\rho_m^\ell(i) = \rho_m^\ell(j)\}$ for any instance $\nu_m^\ell \in \bar{\mathcal{V}}_\nu$. For all $m \in [n]$ and $\ell \in [k]\backslash\{\rho(m)\}$, the difference $C_{ij} - C_{ij}^{m,\ell}$ satisfies

$$C_{ij} - C_{ij}^{m,\ell} = \begin{cases} 1 & \text{if } ((i = m \wedge \rho(j) = \rho(m)) \vee (j = m \wedge \rho(i) = \rho(m))) \wedge (i \neq j) \\ -1 & \text{if } ((i = m \wedge \rho(j) = \ell) \vee (j = m \wedge \rho(i) = \ell)) \wedge (i \neq j) \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Therefore, for any $\mathbf{W} \in \Delta$,

$$\begin{aligned} &\inf_{\nu' \in \bar{\mathcal{V}}_\nu} \sum_{i>j} W_{ij}(C_{ij} - C_{ij}^{'})[C_{ij}d(p,q) - (1 - C_{ij})d(q,p)] \\ &\qquad = \min_{\substack{m \in [n] \\ \ell \neq \rho(m)}} \sum_{i>j} W_{ij}(C_{ij} - C_{ij}^{m,\ell})\left[C_{ij}d(p,q) - (1 - C_{ij})d(q,p)\right] \\ &\qquad = \min_{\substack{m \in [n] \\ \ell \neq \rho(m)}} \left( d(p,q) \sum_{i \neq m} W_{im}\mathbf{I}\{\rho(i) = \rho(m)\} + d(q,p) \sum_{i \neq m} W_{im}\mathbf{I}\{\rho(i) = \ell\} \right) \\ &\qquad = \min_{\substack{m \in [n] \\ \ell \neq \rho(m)}} \left( d(p,q)H_{m\rho(m)} + d(q,p)H_{m\ell} \right). \end{aligned}$$

The first equality follows from the definition of $\bar{\mathcal{V}}_\nu$, the second from eq. (11) and the symmetry of $W_{ij}$ and $C_{ij}$, and the third by defining $H_{m\ell}$ to be the total weight in $\mathbf{W}$ assigned by item $m$ to items in cluster $\ell$ for all $m \in [n]$ and $\ell \in [k]$. That is,

$$H_{m\ell} = \sum_{i \neq m} W_{mi}\mathbf{I}\{\rho(i) = \ell\} = \sum_{i \neq m} W_{im}\mathbf{I}\{\rho(i) = \ell\}.$$

Let $\mathcal{H} = \{\mathbf{H} \in [0,1]^{n \times k} : H_{m\ell} = \sum_{i \neq m} W_{mi}\mathbf{I}\{\rho(i) = \ell\}\ ,\ \mathbf{W} \in \Delta\}$ be the set of all matrices $\mathbf{H}$ whose entries have the form specified above. The optimization problem in eq. (10) can now be equivalently written as

$$\begin{aligned} &\sup_{\mathbf{W} \in \Delta} \inf_{\nu' \in \bar{\mathcal{V}}_\nu} \sum_{i>j} W_{ij}(C_{ij} - C_{ij}^{'})\left[C_{ij}d(p,q) - (1 - C_{ij})d(q,p)\right] \\ &\qquad = \sup_{\mathbf{H} \in \mathcal{H}} \min_{\substack{m \in [n] \\ \ell \neq \rho(m)}} d(p,q)H_{m\rho(m)} + d(q,p)H_{m\ell}. \end{aligned}$$

It is easy to see that an optimal solution $\mathbf{H}^* \in \mathcal{H}$ for the optimization problem above must ensure that $d(p,q)H^*_{m\rho(m)} + d(q,p)H^*_{m\ell} = c$ for all $m \in [n]$ and $\ell \in [k]\backslash\{\rho(m)\}$ for the largest possible constant $c$. The optimal value corresponding to such a $\mathbf{H}^*$ would then be $c$. Moreover, because of the constraints on $\mathbf{W}$, the matrix $\mathbf{H}^*$ must also satisfy $\sum_{m=1}^{n}\sum_{\ell=1}^{k} H^*_{m\ell} = 2$. We consider three cases, mentioning an optimal solution $\mathbf{H}^*$ in each case that satisfies all the requirements above.

**Case 1 -** $d(p,q) > d(q,p)/(k-1)$**:** $H^*_{m\ell} = \frac{2}{n}$ if $\ell = \rho(m)$ and 0 otherwise with optimal $c = \frac{2d(p,q)}{n}$.

**Case 2 -** $d(p,q) < d(q,p)/(k-1)$**:** $H^*_{m\ell} = \frac{2}{n(k-1)}$ if $\ell \neq \rho(m)$ and 0 otherwise with optimal $c = \frac{2d(q,p)}{n(k-1)}$.

**Case 3 -** $d(p,q) = d(q,p)/(k-1)$**:** $H^*_{m\ell} = \frac{2}{nk}$ for all $m$, $\ell$. The value of optimal $c = \frac{2d(q,p)}{n(k-1)} = \frac{2d(p,q)}{n}$.

In all cases, the optimal value $c$ is bounded above by $\frac{2}{n}\max\{d(p,q), d(q,p)/(k-1)\}$. Using this bound on the optimal value of the optimization problem in eq. (10) gives

$$\mathrm{E}_\nu[\tau] \geq \frac{n}{2\max\left\{d(p,q), \frac{d(q,p)}{k-1}\right\}} \ln\frac{1}{2.4\delta},$$

which completes the proof.

□

# C ADDITIONAL DETAILS REGARDING THE ALGORITHM

We prove Lemma 4.1 in Appendix C.1, provide more details about `QBClusterH` in Appendix C.2, and show how the learner can safely stop querying item pair $\{a, i\}$ as soon as item $i$ is assigned to a cluster in Appendix C.3.

## C.1 Confidence Intervals in `SplitItemsH`

We prove a slightly more general result than Lemma 4.1, which will be useful in Appendix C.3. Towards this end, the following concentration bound for the sum of a random number of independent sub-Gaussian random variables will be useful.

**Lemma C.1.** *Let $X_1, X_2, \ldots$ be independent sub-Gaussian random variables with $\ln \mathrm{E}\left[e^{\lambda X_t}\right] \leq \lambda^2/8$ for all $t \geq 1$. Define $Z_1, Z_2, \ldots$ to be Bernoulli random variables such that $Z_{t+1}$ is $\mathcal{F}_t$-measurable, where $\mathcal{F}_t \coloneqq \sigma(X_1, X_2, \ldots, X_t)$ is the $\sigma$-algebra generated by $X_1, \ldots, X_t$. The following holds:*

$$\mathrm{P}\left(\forall t \geq 1, \ \left|\sum_{s=1}^{t} X_s Z_s\right| \leq \sqrt{\frac{\sum_{s=1}^{t} Z_s}{2} \ln \frac{2t^2(t+1)}{\delta}}\right) \geq 1 - \delta.$$

*Proof.* Fix an arbitrary $t \geq 1$ and define $S(t) = \sum_{s=1}^{t} X_s Z_s$ and $T(t) = \sum_{s=1}^{t} Z_s$. Further, for all $t \geq 1$ and a constant $\lambda \in \mathbb{R}$, define $M(t)$ as

$$M(t) = \exp\left(\lambda S(t) - \frac{\lambda^2 T(t)}{8}\right).$$

$(M(t))_t$ is a super-martingale with respect to $(\mathcal{F}_t)_t$. To see this, note that

$$\begin{aligned}
\mathrm{E}[M(t+1)|\mathcal{F}_t] &= \mathrm{E}\left[\exp\left(\lambda S(t+1) - \frac{\lambda^2 T(t+1)}{8}\right)\Big|\mathcal{F}_t\right] \\
&= M(t)\ \left(\mathrm{E}\left[\exp\left(\lambda X_{t+1} - \frac{\lambda^2}{8}\right)\right]\mathbf{I}\{Z_{t+1} = 1\} + \mathbf{I}\{Z_{t+1} = 0\}\right) \leq M(t),
\end{aligned}$$

where the last inequality follows from our assumption that $\mathrm{E}\left[e^{\lambda X_{t+1}}\right] \leq \exp(\lambda^2/8)$. As $(M(t))_t$ is a super-martingale, $\mathrm{E}[M(t)] \leq \mathrm{E}[M(1)] \leq 1$. Now consider the event $\mathcal{E}_\ell(t) = \{S(t) \geq \epsilon \wedge T(t) = \ell\}$ for a fixed natural number $\ell$ between 1 and $t$. Using a Markov style inequality,

$$\begin{aligned}
\mathrm{P}(\mathcal{E}_\ell(t)) &\leq e^{-\lambda\epsilon}\, \mathrm{E}\left[\exp\left(\lambda S(t)\right)\ \mathbf{I}\{T(t) = \ell\}\right] \\
&\leq e^{-\lambda\epsilon + \frac{\lambda^2 \ell}{8}}\ \mathrm{E}\left[\exp\left(\lambda S(t) - \frac{\lambda^2 T(t)}{8}\right)\ \mathbf{I}\{T(t) = \ell\}\right] \\
&\leq e^{-\lambda\epsilon + \frac{\lambda^2 \ell}{8}}\ \mathrm{E}\left[M(t)\right] \\
&\leq \exp\left(-\lambda\epsilon + \frac{\lambda^2 \ell}{8}\right).
\end{aligned}$$

Taking a supremum over all values of $\lambda \in \mathbb{R}$ gives $\mathrm{P}(\mathcal{E}_\ell(t)) \leq \exp\left(-\frac{2\epsilon^2}{\ell}\right)$. Therefore, as $T(t) \leq t$,

$$\mathrm{P}\left(S(t) \geq \sqrt{\frac{T(t)}{2} \ln \frac{2t^2(t+1)}{\delta}}\ \wedge\ T(t) = \ell\right) \leq \frac{\delta}{2t^2(t+1)} \qquad \ell = 1, 2, \ldots, t.$$

As $T(t) = 0$ implies that $Z_s = 0$ for all $s \leq t$, we must have $S(t) = 0$ whenever $T(t) = 0$. Therefore, $\mathrm{P}(\mathcal{E}_0(t)) = 0$ for any $\epsilon > 0$. Taking a union bound over the events $\mathcal{E}_\ell(t)$ for all possible values of $\ell$ gives

$$\mathrm{P}\left(S(t) \geq \sqrt{\frac{T(t)}{2} \ln \frac{2t^2(t+1)}{\delta}}\right) \leq \sum_{\ell=0}^{t} \frac{\delta}{2t^2(t+1)} = \frac{\delta}{2t(t+1)}.$$

One can analogously show a high probability lower bound on $S(t)$. Combining the two bound gives

$$\mathrm{P}\left(|S(t)| \geq \sqrt{\frac{T(t)}{2} \ln \frac{2t^2(t+1)}{\delta}}\right) \leq \frac{\delta}{t(t+1)}.$$

Taking a union bound over all time steps $t = 1, 2, \ldots$ and noting that $\sum_{t\geq 1} \frac{1}{t(t+1)} = 1$ produces the desired result.

□

We now prove a more general variant of Lemma 4.1 that does not assume that each item pair of the form $\{a, i\}$ is sampled at each step by the algorithm. While `SplitItemsH` does sample every such item pair at each step, we later show in Appendix C.3 that this can be avoided in practice, slightly reducing the number of queries made by the algorithm. The result below will be useful over there.

**Lemma C.2.** *Define $\beta(t,\delta) = \ln \frac{2(t+1)t^2|\mathcal{N}|^2}{\delta}$. Let $a$ be an anchor item. Define $\mathcal{N}\backslash\{a\} = \mathcal{J}(1) \supseteq \mathcal{J}(2) \supseteq \mathcal{J}(3)\ldots$ such that item pair $\{a, i\}$ is queried at time $t$ if and only if $i \in \mathcal{J}(t)$. Define $Z_{ai}(t) = \mathbf{I}\{i \in \mathcal{J}(t)\}$ and $T_{ai}(t) = \sum_{s=1}^{t} Z_{ai}(s)$, and set*

$$\hat{\mu}_{ai}(t) = \frac{1}{T_{ai}(t)} \sum_{s=1}^{T_{ai}(t)} O_{ai}(s), \;\; l_{ai}(t) = \hat{\mu}_{ai}(t) - \sqrt{\frac{\beta(t,\delta)}{2T_{ai}(t)}}, \;\; \text{and} \;\; u_{ai}(t) = \hat{\mu}_{ai}(t) + \sqrt{\frac{\beta(t,\delta)}{2T_{ai}(t)}}, \tag{12}$$

*where $O_{ai}(s)$ is the outcome of querying item pair $\{a, i\}$ at time $s$. Further, define,*

$$\hat{p}_l(t) \coloneqq \max_{i\in\mathcal{N}\backslash\{a\}} l_{ai}(t) \qquad \text{and} \qquad \hat{q}_u(t) \coloneqq \min_{i\in\mathcal{N}\backslash\{a\}} u_{ai}(t). \tag{13}$$

*Assume that $\mathcal{J}(t)$ is fully determined by the queries and observations made until time $t$. Then, the event $\{\forall t \geq 1, \;\; p \geq \hat{p}_l(t) \;\wedge\; q \leq \hat{q}_u(t) \;\wedge\; \forall i \in \mathcal{N}\backslash\{a\}, \; \alpha_{ai} \in [l_{ai}(t), u_{ai}(t)]\}$ happens with probability at least $1 - \delta$.*

*Proof.* Consider a fixed anchor $a$ and another item $i \in \mathcal{N}\backslash\{a\}$. Define $(Y_{ai}(t))_{t\geq 1}$ to be a sequence of $i.i.d.$ observations of the random variable $O_{ai}$ from eq. (1), and let $\tilde{Y}_{ai}(t) = Y_{ai}(t) - \alpha_{ai}$, where recall that $\alpha_{ai} = \mathrm{E}[O_{ai}]$. As $\mathcal{J}(t+1) \subseteq \mathcal{J}(t)$, the learner observes $O_{ai}(t) = Y_{ai}(t)$ if the item pair $\{a, i\}$ is queried at time $t$.

Let $\mathcal{F}(t) \coloneqq \sigma(a, (Y_{aj}(s))_{s\leq t, j\in\mathcal{N}\backslash\{a\}}, (Z_{aj}(s))_{s\leq t, j\in\mathcal{N}\backslash\{a\}})$. As the decision to keep an item in $\mathcal{J}(t)$ only depends on the samples collected till time $t$, the random variable $Z_{ai}(t+1)$ is $\mathcal{F}(t)$-measurable. Define $S_{ai}(t) = \sum_{s=1}^{t}(O_{ai}(s) - \alpha_{ai})Z_{ai}(s) = \sum_{s=1}^{t}(Y_{ai}(s) - \alpha_{ai})Z_{ai}(s) = \sum_{s=1}^{t} \tilde{Y}_{ai}(s)Z_{ai}(s)$. Note that $(\tilde{Y}_{ai}(t))_{t\geq 1}$ are independent and it can be easily verified that they satisfy $\ln \mathrm{E}\left[\exp(\lambda \tilde{Y}_{ai}(t))\right] \leq \lambda^2/8$. By Lemma C.1,

$$\mathrm{P}\left(\forall t \geq 1, \;\; \left|\frac{S_{ai}(t)}{T_{ai}(t)}\right| \leq \sqrt{\frac{1}{2T_{ai}(t)} \ln \frac{2t^2(t+1)|\mathcal{N}|^2}{\delta}}\right) \geq 1 - \frac{\delta}{|\mathcal{N}|^2}.$$

Moreover,

$$\frac{S_{ai}(t)}{T_{ai}(t)} = \frac{1}{T_{ai}(t)} \sum_{s=1}^{t} (O_{ai}(s) - \alpha_{ai})Z_{ai}(s) = \frac{1}{T_{ai}(t)} \sum_{s=1}^{T_{ai}(t)} (O_{ai}(s) - \alpha_{ai}) = \hat{\mu}_{ai}(t) - \alpha_{ai}.$$

With $l_{ai}(t) = \hat{\mu}_{ai}(t) - \sqrt{\frac{\beta(t,\delta)}{2T_{ai}(t)}}$ and $u_{ai}(t) = \hat{\mu}_{ai}(t) + \sqrt{\frac{\beta(t,\delta)}{2T_{ai}(t)}}$, we get

$$\mathrm{P}(\forall t \geq 1, \;\; \alpha_{ai} \in [l_{ai}(t), u_{ai}(t)]) \geq 1 - \frac{\delta}{|\mathcal{N}|^2}.$$

The bound above is true for arbitrary choices of a fixed anchor item $a$ and another item $i \in \mathcal{N}\backslash\{a\}$. Taking a union bound over all possible choices of $a$ and $i$ gives

$$\mathrm{P}(\forall t \geq 1, i \in \mathcal{N}\backslash\{a\}, \;\; \alpha_{ai} \in [l_{ai}(t), u_{ai}(t)]) \geq 1 - \delta.$$

The anchor $a$ in the equation above is randomly chosen.

Proving the validity of $\hat{p}_l(t)$ and $\hat{q}_u(t)$ is now easy. Let $\mathcal{E} = \{\forall t \geq 1, i \in \mathcal{N}\backslash\{a\}, \;\; \alpha_{ai} \in [l_{ai}(t), u_{ai}(t)]\}$ be the event that the confidence intervals above are valid for all $t \geq 1$. Then,

$$\begin{aligned} \mathrm{P}((\forall t \geq 1, \; \hat{p}_l(t) \leq p \;\wedge\; \hat{q}_u(t) \geq q) \;\wedge\; \mathcal{E}) &= \mathrm{P}(\forall t \geq 1, \; \hat{p}_l(t) \leq p \;\wedge\; \hat{q}_u(t) \geq q \mid \mathcal{E})\, \mathrm{P}(\mathcal{E}) \\ &\geq \mathrm{P}(\forall t \geq 1, \; \hat{p}_l(t) \leq p \;\wedge\; \hat{q}_u(t) \geq q \mid \mathcal{E})\, (1 - \delta) \\ &\geq 1 - \delta. \end{aligned} \tag{14}$$

**Algorithm 2** `QBClusterH`$(n, \delta)$ more than two clusters

**Input:** Set of items $\mathcal{N}$ and confidence level $\delta$
Arbitrarily order items in $\mathcal{N}$
Set $j = 1$ and $\mathcal{N}^j = \mathcal{N}$
**while** $|N^j| > 0$ **do**
    $\mathcal{C}_1^j, \mathcal{C}_2^j =$ `SplitItemsH`$(\mathcal{N}^j, \delta/j(j+1))$ where $a^j$ is the first item in $\mathcal{N}^j$ and eq. (15) is used
    Set $\mathcal{C}_j = \mathcal{C}_1^j$ and $\mathcal{N}^{j+1} = \mathcal{C}_2^j$
    Set $j = j + 1$
**end while**
**Return:** $\mathcal{C}_1, \ldots, \mathcal{C}_{j-1}$

The first inequality uses the fact that $\mathrm{P}(\mathcal{E}) \geq 1 - \delta$. To see the second inequality, note that under the event $\mathcal{E}$,

$$\hat{p}_l(t) = \min_{i \in \mathcal{N} \setminus \{a\}} l_{ai}(t) = \min \left\{ \min_{\substack{i \in \mathcal{N} \setminus \{a\} \\ \alpha_{ai} = p}} l_{ai}(t) \;\; , \;\; \min_{\substack{i \in \mathcal{N} \setminus \{a\} \\ \alpha_{ai} = q}} l_{ai}(t) \right\} \leq \min\{p, q\} \leq p.$$

One can similarly show that $\hat{q}_u(t) \geq q$ under the event $\mathcal{E}$, and hence the second inequality in eq. (14). This finishes the proof.

□

Lemma 4.1 turns out to be a special case of Lemma C.2. In Lemma 4.1, we set $\mathcal{J}(t) = \mathcal{N} \setminus \{a\}$ for all $t$ as `SplitItemsH` samples all pairs of the form $\{a, i\}$ for $i \in \mathcal{N} \setminus \{a\}$ at each step $t$. This clearly satisfies all the requirements imposed on $\mathcal{J}(t)$ in Lemma C.2. For this choice, we get $Z_{ai}(t) = 1$ and $T_{ai}(t) = t$ for all $i \in \mathcal{N} \setminus \{a\}$ and $t \geq 1$. Using these values in Lemma C.2 recovers Lemma 4.1.

### C.2 More Details About `QBClusterH`

As described in Section 4.2, `QBClusterH` proceeds in phases, calling `SplitItemsH` on the remaining items in each phase until everything is clustered. Recall that `SplitItemsH` uses the estimates $\hat{p}_l(t)$ and $\hat{q}_u(t)$ from Lemma 4.1 to calculate a lower bound on $p$ and an upper bound on $q$, respectively. In Lemma 4.1, these estimates are written for one call to `SplitItemsH`, and hence they use a fixed anchor $a$. We now present a way to compute these estimates by combining the observations collected across phases.

We index phases by $j$. Focus on phase $j$, and let $\mathcal{N}^j \subseteq \mathcal{N}$ be the set of items that `SplitItemsH` was asked to partition in this phase. For doing so, the algorithm choses an anchor $a^j \in \mathcal{N}^j$, runs for $T_j$ steps, and returns two clusters $\mathcal{C}_1^j, \mathcal{C}_2^j \subseteq \mathcal{N}^j$, where $\mathcal{C}_1^j$ has all items that are believed to be in the same cluster as $a^j$ and $\mathcal{C}_2^j = \mathcal{N}^j \setminus \mathcal{C}_1^j$ has the remaining items. We add a superscript $j$ to the estimates $\hat{p}_l(t)$ and $\hat{q}_u(t)$ used in `SplitItemsH` to clearly mark the phase. These estimates are now calculated as

$$\hat{p}_l^j(t) \coloneqq \max \left\{ \max_{i \in \mathcal{N}^j \setminus \{a^j\}} l_{a^j i}(t), \;\; \hat{p}_l^{j-1}(T_{j-1}) \right\} \;\text{ and }\; \hat{q}_u^j(t) \coloneqq \min \left\{ \min_{i \in \mathcal{N}^j \setminus \{a^j\}} u_{a^j i}(t), \;\; \hat{q}_u^{j-1}(T_{j-1}) \right\}. \tag{15}$$

For these to be valid, we define $\hat{p}_l^0(T_0) = -\infty$ and $\hat{q}_u^0(T_0) = +\infty$. The estimate $\hat{p}_l^j(t)$ now corresponds to the maximum $l_{ij}(t)$ value seen till now across all item pairs sampled in phases $j$ or earlier. One can make an analogous statement about $\hat{q}_u^j(t)$.

We additionally (arbitrarily) order all items in $\mathcal{N}$ at the beginning, and then always choose the first unclustered item in $\mathcal{N}^j$ as the anchor $a^j$ in phase $j$. Now, the item pairs queried at time $t$ in phase $j$ completely depend on the past observations (including observations from earlier phases). This ensures that all assumptions in Lemma C.2 are satisfied, and hence these estimates remain valid. Algorithm 2 summarizes the pseudocode for `QBClusterH`.

We want to make one final remark about this algorithm. Consider what happens at the beginning of the $(j+1)^{th}$ phase in `QBClusterH`. The algorithm has found $j$ clusters $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_j$ by this time with high confidence. One might be tempted to condition on the event that these clusters are correct, and calculate confidence intervals $[\hat{p}_l^j, \hat{p}_u^j]$ and $[\hat{q}_l^j, \hat{q}_u^j]$ around $p$ and $q$ by *combining* past observations from item pairs within and across clusters

**Algorithm 3** A more efficient implementation of `SplitItemsH`$(\mathcal{N}, \delta)$ for $k = 2$

**Input:** Set of items $\mathcal{N}$ and confidence parameter $\delta$
Sample an anchor $a \overset{\text{unif}}{\sim} \mathcal{N}$
Set $t = 1$, $\mathcal{C}_1 = \{a\}$, $\mathcal{C}_2 = \phi$, $\mathcal{U}(1) = \mathcal{N} \backslash \{a\}$, and $\mathcal{W} = \phi$
**while** $|\mathcal{U}(t)| > 0$ **do**
    `# Sample item pairs and update confidence intervals`
    Sample item pairs $\{a, i\}$ for all $i \in \mathcal{U}(t) \cup \mathcal{W}$
    Update $\hat{\mu}_{ai}(t)$, $l_{ai}(t)$ and $u_{ai}(t)$ for all $i \in \mathcal{N} \backslash \{a\}$ using Lemma C.2
    Update $\hat{p}_l(t)$ and $\hat{q}_u(t)$ using Lemma C.2

    `# Assign items to clusters`
    $\mathcal{C}_1 = \mathcal{C}_1 \cup \{i \in \mathcal{U}(t) : l_{ai}(t) > \hat{q}_u(t)\}$
    $\mathcal{C}_2 = \mathcal{C}_2 \cup \{i \in \mathcal{U}(t) : u_{ai}(t) < \hat{p}_l(t)\}$

    `# Record first additions to` $\mathcal{C}_1$ `and` $\mathcal{C}_2$
    **if** $t = \min\{s \leq t : \exists i, j \in \mathcal{U}(s), l_{ai}(s) > \hat{q}_u(s) \ \wedge \ u_{aj}(s) < \hat{p}_l(s)\}$ **then**
        Set $\mathcal{W} = \{i, j\}$ for randomly chosen $i \in \mathcal{C}_1 \backslash \{a\}$ and $j \in \mathcal{C}_2$
    **end if**

    $\mathcal{U}(t+1) = \mathcal{U}(t) \backslash (\mathcal{C}_1 \cup \mathcal{C}_2)$
    $t = t + 1$
**end while**
**Return:** $\mathcal{C}_1$ and $\mathcal{C}_2$

$\mathcal{C}_1, \ldots, \mathcal{C}_j$, respectively. That is, one can collect all old samples for item pairs that are now known to lie in the same cluster to get a sharper estimate of $p$, and do the same with items across clusters to get a sharper estimate of $q$. The hope is that such $\hat{p}_l^j$ and $\hat{q}_u^j$ will offer a tighter lower and upper bound on $p$ and $q$ as compared to eq. (15).

Unfortunately, we can't do this. The tool we have for computing such confidence intervals is Lemma C.1. Think of $Z_t$ in Lemma C.1 as indicating if observation $X_t$ should be included in the sum or not. In our case, we want to include an observation $O_{ai}(t)$ in estimating $p$ only if items $a$ and $i$ belong together in one of the already found clusters. Thus, the "$Z_t$ value" for the observation $O_{ai}(t)$ depends on whether item $i$ was at some point moved to the same cluster as the anchor $a$ by `SplitItemsH`. This decision, however, depends not just on $O_{ai}(t)$, but also on other observations for the pair $\{a, i\}$ that were potentially collected after the observation $O_{ai}(t)$. Therefore, this "$Z_t$ value" for $O_{ai}(t)$ is not completely determined by things that happen before time $t$, and hence we violate the assumption that $Z_t$ is $\mathcal{F}_{t-1}$-measurable in Lemma C.1

### C.3 Avoiding Unnecessary Queries in `SplitItemsH`

Algorithm 3 presents a slightly more efficient variant of `SplitItemsH`. There are two key changes. *First*, at each step $t$, Algorithm 3 only samples item pairs $\{a, i\}$ where $i \in \mathcal{U}(t) \cup \mathcal{W}$. Ignore the set $\mathcal{W}$ for now, this means that the algorithm keeps sampling an item pair $\{a, i\}$ only if item $i$ is currently unassigned. In contrast, Algorithm 1 samples pairs $\{a, i\}$ for all $i \in \mathcal{N} \backslash \{a\}$. The set $\mathcal{U}(t)$ is completely determined by the past observations. We show later that this is also true for the set $\mathcal{W}$. Therefore, we get valid confidence intervals by setting $\mathcal{J}(t) = \mathcal{U}(t) \cup \mathcal{W}$ in Lemma C.2. The correctness of this algorithm can be argued in the same way as `SplitItemsH` once the validity of these confidence intervals is established.

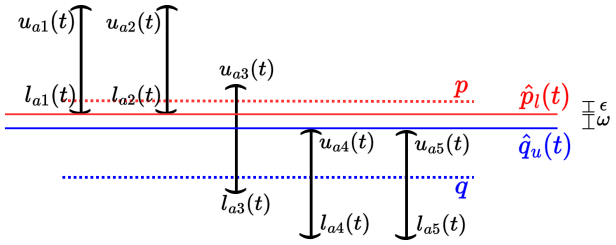



Figure 4: An example demonstrating the role of $\mathcal{W}$ in Algorithm 3.

The algorithm will sometimes run into problems if

it only samples items in $\mathcal{U}(t)$. The *second* addition in Algorithm 3 is the set $\mathcal{W}$ which addresses this issue. The utility of $\mathcal{W}$ is best explained through an example. Figure 4 shows such an example with six items. Let $a = 6$ be the anchor (not shown in the figure). The figure shows the confidence intervals $\mathcal{I}_{ai}(t)$ of the remaining five items with respect to the anchor. Assume that the ground truth clusters are $\mathcal{Y}_1 = \{1, 2, 3, 6\}$ and $\mathcal{Y}_2 = \{4, 5\}$. All confidence intervals correctly contain the respective $\alpha_{ai}$ values (marked by dotted lines in Figure 4). The figure also shows the current value of $\hat{p}_l(t)$ and $\hat{q}_u(t)$ as per Lemma C.2. Clearly, at this stage, the algorithm will have $\mathcal{C}_1 = \{1, 2, 6\}$, $\mathcal{C}_2 = \{4, 5\}$, and $\mathcal{U}(t) = \{3\}$ using the conditions in Lines 11 and 12 in Algorithm 3. Let $p - \hat{p}_l(t) = \epsilon$ and $\hat{p}_l(t) - \hat{q}_u(t) = \omega$. If we only sample the item pair $\{a, 3\}$ based on the contents of $\mathcal{U}(t)$, the value of $\hat{p}_l(t)$ and $\hat{q}_u(t)$ will not change. We may then need to collect enough samples to reduce the length of $\mathcal{I}_{a3}(t)$ to $\epsilon + \omega$ in the worst case[3] before $l_{a3}(t)$ becomes greater than $\hat{q}_u(t)$ and item 3 is moved to $\mathcal{C}_1$. The problem, however, is that $\epsilon$ and $\omega$ may be arbitrarily small, and hence we may need arbitrarily many queries. Algorithm 3 addresses this issue by querying additional items to continuously improve $\hat{p}_l(t)$ and $\hat{q}_u(t)$ so that $\omega$ cannot be arbitrarily small. To do this, it adds the first items added to $\mathcal{C}_1 \backslash \{a\}$ and $\mathcal{C}_2$ to a set $\mathcal{W}$ in Lines 15 to 17. It then samples item pairs of the form $\{a, i\}$ for all $i \in \mathcal{U}(t) \cup \mathcal{W}$. This ensures that at least one item from each cluster is continuously sampled, which makes $\hat{p}_l(t)$ and $\hat{q}_u(t)$ converge to $p$ and $q$, respectively, preventing $\omega$ from becoming arbitrarily small. Note again that the items in the set $\mathcal{W}$ at time $t$ are completely determined by the past observations.

# D ADDITIONAL DETAILS REGARDING THE ANALYSIS

In this section, we prove the correctness of `SplitItemsH` and `QBClusterH` and analyze their sample complexity. We also show that these arguments, with a bit more work, apply to the more efficient implementation of `SplitItemsH` from Appendix C.3 as well.

Let $\tilde{l}_{ai}(t) = \max_{s \le t} l_{ai}(s)$ and $\tilde{u}_{ai}(t) = \min_{s \le t} u_{ai}(s)$. It is easy to see that if $l_{ai}(s)$ and $u_{ai}(s)$ are valid for all $s \le t$, then $\alpha_{ai} \in [\tilde{l}_{ai}(t), \tilde{u}_{ai}(t)]$. We can, therefore, equivalently use $\tilde{l}_{ai}(t)$ and $\tilde{u}_{ai}(t)$ to define the confidence interval for $\alpha_{ai}$. These confidence intervals will be valid for all $t \ge 1$ and all items $i \in \mathcal{N} \backslash \{a\}$ with probability at least $1 - \delta$ by Lemma 4.1 (or Lemma C.2). We can similarly use $\tilde{p}_l(t) = \max_{s \le t} \hat{p}_l(s)$ and $\tilde{q}_u(t) = \min_{s \le t} \hat{q}_u(s)$ instead of $\hat{p}_l(t)$ and $\hat{q}_u(t)$. Besides being valid with high probability, these estimates have the desirable property of being monotonic. That is,

$$\tilde{l}_{ai}(t+1) \ge \tilde{l}_{ai}(t), \quad \tilde{u}_{ai}(t+1) \le \tilde{u}_{ai}(t), \quad \tilde{p}_l(t+1) \ge \tilde{p}_l(t), \quad \text{and} \quad \tilde{q}_u(t+1) \le \tilde{q}_u(t).$$

Thus, the confidence intervals can only shrink, the lower bound on $p$ can only go up, and the upper bound on $q$ can only come down as more queries are made. We use these modified estimates both in our implementation and in our analysis, but for simplicity we don't use the new notation.

Let $a$ be an anchor item. In Algorithm 1, for any item $i \in \mathcal{N} \backslash \{a\}$, the value of the lower end $l_{ai}(t)$ of the confidence interval $\mathcal{I}_{ai}(t) \coloneqq [l_{ai}(t), u_{ai}(t)]$ continuously increases with $t$, as we query all item pairs $\{a, i\}$ for $i \in \mathcal{N} \backslash \{a\}$. Therefore, the estimate $\hat{p}_l(t) = \max_{i \in \mathcal{N} \backslash \{a\}} l_{ai}(t)$ also improves with time. One can analogously argue that $\hat{q}_u(t)$ also continuously improves as more queries are made. The following lemma quantifies this improvement and shows that it also happens for the variant of `SplitItemsH` in Appendix C.3, even though this variant does not sample all item pairs.

**Lemma D.1.** *Assume that each ground truth cluster has at least two items. With $\beta(t, \delta)$ defined in Lemma 4.1, the estimates $\hat{p}_l(t)$ and $\hat{q}_u(t)$ in Algorithms 1 and 3 satisfy the following with probability at least $1 - \delta$.*

$$\hat{p}_l(t) \ge p - 2\sqrt{\beta(t, \delta)/2t} \qquad \text{and} \qquad \hat{q}_u(t) \le q + 2\sqrt{\beta(t, \delta)/2t} \qquad \forall t \ge 1.$$

*Proof.* Let $\mathcal{E} = \{\forall t \ge 1, \;\; p \ge \hat{p}_l(t) \;\wedge\; q \le \hat{q}_u(t) \;\wedge\; \forall i \in \mathcal{N} \backslash \{a\}, \; \alpha_{ai} \in [l_{ai}(t), u_{ai}(t)]\}$ be the event that the estimates $\hat{\rho}_l(t)$, $\hat{q}_u(t)$, and $\mathcal{I}_{ai}(t)$ are valid at all time $t$ and for all items $i \in \mathcal{N} \backslash \{a\}$ and a randomly chosen anchor $a$.

The proof is fairly simple for Algorithm 1. Consider an item $i$ such that $\alpha_{ai} = p$. Such an $i$ always exists as each cluster has at least two items. Under the event $\mathcal{E}$, by Lemma 4.1,

$$l_{ai}(t) \ge p - (u_{ai}(t) - l_{ai}(t)) = p - 2\sqrt{\beta(t, \delta)/2t}.$$

[3] In the worst case, we will have $p = u_{a3}(t)$ making $\mathcal{I}_{a3}(t)$ valid but very lopsided. Note that the confidence intervals need not be centered around the (unknown) true mean.

As $\hat{p}_l(t) = \max_{j \in \mathcal{N}\setminus\{a\}} l_{aj}(t)$, we therefore get $\hat{p}_l(t) \geq l_{ai}(t) \geq p - 2\sqrt{\beta(t,\delta)/2t}$. One can analogously show that $\hat{q}_u(t) \leq q + 2\sqrt{\beta(t,\delta)/2t}$.

Now lets move to Algorithm 3. We first argue that at least one item besides $a$ is eventually added to both $\mathcal{C}_1$ and $\mathcal{C}_2$. Suppose for the sake of contradiction that this doesn't happen. In this case, $\mathcal{U}(t) = \mathcal{N}\setminus\{a\}$ for all $t$. Take two items $i, j \in \mathcal{N}\setminus\{a\}$ such that $\alpha_{ai} = p$ and $\alpha_{aj} = q$. As more samples are collected, $l_{ai}(t)$ approaches $p$ and $u_{aj}(t)$ approaches $q$ under the event $\mathcal{E}$ as $\lim_{t\to\infty} \sqrt{\beta(t,\delta)/2t} = 0$. Therefore, we must have $l_{ai}(t) > u_{aj}(t)$ for large enough (but finite) $t$ as $p > q$. This triggers the movement of item $i$ to cluster $\mathcal{C}_1$ and item $j$ to cluster $\mathcal{C}_2$ (see Lines 11 and 12), leading to a contradiction.

Let $t_1$ be the first time step where this happens and let $i_1$ and $j_1$ be in the set of items moved to $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively, at this step. Without loss of generality, we assume that $\mathcal{W} = \{i_1, j_1\}$ in Lines 15 to 17. This means that pairs $\{a, i_1\}$ and $\{a, j_1\}$ will be queried at all $t \geq t_1$. These pairs were also queried for all $t < t_1$ as $i_1, j_1 \in \mathcal{U}(t)$ for $t < t_1$. Therefore, at any time $t$, the confidence intervals associated with these items is given by

$$l_{ai_1}(t) = \hat{\mu}_{ai_1}(t) - \sqrt{\beta(t,\delta)/2t} \qquad u_{ai_1}(t) = \hat{\mu}_{ai_1}(t) + \sqrt{\beta(t,\delta)/2t},$$
$$l_{aj_1}(t) = \hat{\mu}_{aj_1}(t) - \sqrt{\beta(t,\delta)/2t} \qquad u_{aj_1}(t) = \hat{\mu}_{aj_1}(t) + \sqrt{\beta(t,\delta)/2t}.$$

As $\alpha_{ai_1} = p$ and $\alpha_{aj_1} = q$, under the event $\mathcal{E}$,

$$l_{ai_1}(t) \geq p - (u_{ai_1} - l_{ai_1}) = p - 2\sqrt{\beta(t,\delta)/2t}, \text{ and}$$
$$u_{aj_1}(t) \leq q + (u_{aj_1} - l_{aj_1}) = q + 2\sqrt{\beta(t,\delta)/2t}.$$

As before, $\hat{p}_l(t) \geq l_{ai_1}(t)$ and $\hat{q}_u(t) \leq u_{aj_1}(t)$.

Noting that the event $\mathcal{E}$ happens with probability at least $1 - \delta$ for both Algorithm 1 and Algorithm 3 by Lemma 4.1 and Lemma C.2, respectively, finishes the proof. □

The proof above, in part, shows that $\hat{p}_l(t) \to p$ and $\hat{q}_u(t) \to q$ as $t$ becomes large. The difference between $\hat{p}_l(t)$ and $\hat{q}_u(t)$ can therefore not remain arbitrarily close to zero as more samples are collected. However, this is possible for Algorithm 3 only because it continuously samples items in the set $\mathcal{W}$, justifying the use of this set in the algorithm.

### D.1 Proof of Theorem 5.1

*Proof.* We divide the proof into two parts, the first showing the correctness of `SplitItemsH` and the second dedicated to `QBClusterH`.

**Correctness of `SplitItemsH`:** Let $\mathcal{E} = \{\forall t \geq 1,\ \ p \geq \hat{p}_l(t) \ \wedge\ q \leq \hat{q}_u(t) \ \wedge\ \forall i \in \mathcal{N}\setminus\{a\},\ \alpha_{ai} \in [l_{ai}(t), u_{ai}(t)]\}$ be the event that the estimates $\hat{\rho}_l(t)$, $\hat{q}_u(t)$, and $\mathcal{I}_{ai}(t)$ are valid at all time $t$ and for all items $i \in \mathcal{N}\setminus\{a\}$ and a randomly chosen anchor $a$. We show that all cluster assignments are correct and the algorithm terminates in finite time under event $\mathcal{E}$, which happens with probability at least $1-\delta$ by Lemma 4.1. Keep in mind that $p > q$ by Definition 2.1.

1. **Correctness of the assignments:** An item $i$ is moved to cluster $\mathcal{C}_1$ if $l_{ai}(t) > \hat{q}_u(t)$ in Line 11. Under the event $\mathcal{E}$,
$$\alpha_{ai} \geq l_{ai}(t) > \hat{q}_u(t) \geq q.$$
As $\alpha_{ai}$ takes only two values $p$ and $q$, this implies that $\alpha_{ai} = p$, and hence items $a$ and $i$ belong to the same cluster. The decision to move item $i$ to cluster $\mathcal{C}_1$ is therefore correct. One can similarly show that, under event $\mathcal{E}$, an item $i$ is not in the same cluster as $a$ if $u_{ai}(t) < \hat{p}_l(t)$, making $\mathcal{C}_2$ the correct choice for $i$ if there are only two clusters.

2. **Termination:** The algorithm terminates when $\mathcal{U}(t)$ is empty. Assume for the sake of contradiction that this never happens and let item $i \in \mathcal{U}(t)$ for all $t \geq 1$. The item pair $\{a, i\}$ is sampled at all time steps, and hence
$$l_{ai}(t) = \hat{\mu}_{ai}(t) - \sqrt{\beta(t,\delta)/2t} \qquad \text{and} \qquad u_{ai}(t) = \hat{\mu}_{ai}(t) + \sqrt{\beta(t,\delta)/2t}.$$

Assuming $\alpha_{ai} = p$, we have $l_{ai}(t) \geq p - (u_{ai}(t) - l_{ai}(t)) = p - 2\sqrt{\beta(t,\delta)/2t}$. Let $t'$ be the smallest value of $t$ such that

$$\sqrt{\frac{\beta(t,\delta)}{2t}} < \frac{p-q}{4}. \tag{16}$$

Note that such a $t'$ is finite as $p > q$ and $\lim_{t\to\infty} \sqrt{\frac{\beta(t,\delta)}{2t}} = 0$. At this time step $t'$, by Lemma D.1,

$$l_{ai}(t') - \hat{q}_u(t') \geq p - q - 4\sqrt{\beta(t',\delta)/2t'} > 0,$$

where the last inequality follows from eq. (16). Thus, at a finite time $t'$, we have $l_{ai}(t') > \hat{q}_u(t')$, and item $i$ will be moved from $\mathcal{U}(t')$ to $\mathcal{C}_1$ by Line 11. As $i \neq \mathcal{U}(t'+1)$, this contradicts our assumption that $i \in \mathcal{U}(t)$ for all $t \geq 1$.

One can similarly show that if $\alpha_{ai} = q$ then eventually item $i$ is moved to $\mathcal{C}_2$ after a finite number of steps. Therefore, there can be no item that stays forever in the set of unclustered items $\mathcal{U}(t)$, and hence the algorithm eventually terminates.

As $\mathcal{E}$ happens with probability at least $1-\delta$ and the algorithm terminates in finite time with correct cluster assignment for all items under $\mathcal{E}$, we conclude that Algorithm 1 is $\delta$-PAC.

**Correctness of `QBClusterH`:** Algorithm 2 makes one call to `SplitItemsH` in each phase until all items are clustered. We say that the $j^{th}$ phase has succeeded if the returned clusters satisfy $\mathcal{C}_1^j = \{i \in \mathcal{N}^j : \alpha_{a^j i} = p\}$ and $\mathcal{C}_2^j = \mathcal{N}^j \backslash \mathcal{C}_1^j$, where $a^j$ is the anchor chosen in the call to `SplitItemsH` in this phase. As `SplitItemsH` is $\delta$-PAC[4], the probability of the $j^{th}$ call succeeding in finite time is at least $1-\delta^j$, where `QBClusterH` uses $\delta^j = \frac{\delta}{j(j+1)}$. Suppose there are $k$ ground truth clusters and assume that the first $k$ phases succeed. This happens with probability at least $1 - \sum_{j=1}^{k} \delta^j \geq 1 - \delta$. In each call, the cluster containing the anchor for that call is correctly isolated from the set of unclustered items $\mathcal{N}^{j+1}$ in finite time. This gives the correct $k$ ground truth clusters in the end in finite time, making `QBClusterH` $\delta$-PAC.

□

One can follow the same steps as above to show that Algorithm 3 is $\delta$-PAC by using Lemma C.2 instead of Lemma 4.1. We now show a high probability upper bound on the sample complexity of our algorithms. Note that the proof below applies as is to Algorithm 3 as well.

### D.2 Proof of Theorem 5.2

*Proof.* Let $\mathcal{E} = \{\forall t \geq 1,\ \ p \geq \hat{p}_l(t) \ \wedge\ q \leq \hat{q}_u(t) \ \wedge\ \forall i \in \mathcal{N}\backslash\{a\},\ \alpha_{ai} \in [l_{ai}(t), u_{ai}(t)]\}$ be the event that the estimates $\hat{\rho}_l(t)$, $\hat{q}_u(t)$, and $\mathcal{I}_{ai}(t)$ are valid at all time $t$ and for all items $i \in \mathcal{N}\backslash\{a\}$ and a randomly chosen anchor $a$. In what follows, assume that the event $\mathcal{E}$ has occurred.

Most of the pieces for proving Theorem 5.2 are already in place. Pick an item $i$ with $\alpha_{ai} = p$, and let $t'$ be the smallest value of $t$ such that eq. (16) holds. If $i \in \mathcal{U}(t')$, then, as we show in the proof of Theorem 5.1, at time $t'$,

$$l_{ai}(t') > \hat{q}_u(t').$$

Item $i$ will therefore be moved out of $\mathcal{U}(t)$ at time $t'$ by Line 11. If item $i \notin \mathcal{U}(t')$, then it was removed from $\mathcal{U}(t)$ at some time $t < t'$. In either case, `SplitItemsH` queries the pair $\{a, i\}$ at most $t'$ times. One can analogously

[4]Note that when the first $k-1$ phases succeed, the $k^{th}$ call to `SplitItemsH` only receives items from one cluster. This violates Theorem 5.1's assumption of having two clusters with at least two items each. However, one can still follow the arguments from the proof of Theorem 5.1, and use the estimates from eq. (15), to show that the last call will still succeed with high probability.

show that at most $t'$ queries are made even when $\alpha_{ai} = q$. Now, $t'$ is the smallest natural number that satisfies

$$\begin{aligned}\sqrt{\frac{\beta(t',\delta)}{2t'}} < \frac{p-q}{4} \quad &\Rightarrow \quad \frac{\beta(t',\delta)}{2t'} < \frac{(p-q)^2}{16} \\ &\overset{\text{(a)}}{\Rightarrow} \quad \frac{1}{2t'} \ln \frac{2(t'+1)t'^2 n^2}{\delta} < \frac{(p-q)^2}{16} \\ &\overset{\text{(b)}}{\Rightarrow} \quad t' > \frac{40}{(p-q)^2} \ln \frac{32n^2}{\delta(p-q)^2}\end{aligned}$$

In the calculations above, $n$ denotes the number of items and (a) follows from the definition of $\beta(t,\delta)$ in Lemma 4.1. Implication $(b)$ uses the following fact that can be easily verified numerically (also see Appendix D.3).

**Fact 1.** Let $\alpha \in (0, 1/4)$ and $\beta \geq 4$. Then, $\frac{1}{2x}\ln(\beta x^2(x+1)) < \alpha$ for all $x > \frac{2.5}{\alpha}\ln\left(\frac{\beta}{\alpha}\right)$.

More precisely, implication (b) follows from the right hand side of implication (a) by setting $\alpha = \frac{(p-q)^2}{16}$ and $\beta = \frac{2n^2}{\delta}$ in Fact 1. Fact 1 can be used because $\frac{(p-q)^2}{16} < 1/4$ and $\frac{2n^2}{\delta} > 4$ when $n \geq 2$. Using the smallest integer value of $t'$ that satisfies the right hand side of (b), we conclude that the maximum number of times item pair $\{a, i\}$ is queried is bounded by

$$t' \leq \frac{40}{(p-q)^2} \ln\left(\frac{32n^2}{\delta(p-q)^2}\right) + 1.$$

We can now bound the total number of queries $\tau$ made by the algorithm by adding the maximum number of queries for all items in $\mathcal{N}\backslash\{a\}$. As this calculation is valid under event $\mathcal{E}$, we get with probability at least $1-\delta$ (by Lemma 4.1),

$$\tau \leq (n-1)\left(\frac{40}{(p-q)^2}\ln\left(\frac{32n^2}{\delta(p-q)^2}\right) + 1\right) = O\left(\frac{n}{(p-q)^2}\ln\frac{n^2}{\delta(p-q)^2}\right).$$

□

### D.3 More details about Fact 1

Define $f(x) = \frac{1}{2x}\ln(\beta(x+1)^3)$. Then, for any $\beta \geq 1$ and $x \geq 1$, we have

$$\frac{1}{2x}\ln(\beta x^2(1+x)) \leq f(x).$$

Therefore, to show Fact 1, it suffices to show that $f(x) < \alpha$ for all $x > \frac{2}{\alpha}\ln\frac{\beta}{\alpha}$. Note that,

$$f(x) = \frac{1}{2x}\ln(\beta(x+1)^3) = \frac{1}{2x}\ln(\beta) + \frac{3}{2x}\ln(x+1).$$

Let $f_1(x) = \frac{1}{2x}\ln(\beta)$ and $f_2(x) = \frac{3}{2x}\ln(x+1)$. It is easily seen that both $f_1(x)$ and $f_2(x)$ are monotonically decreasing functions. Moreover,

$$f_1(x) < \frac{\alpha}{4} \quad \Rightarrow \quad x > \frac{2}{\alpha}\ln(\beta) =: x_1.$$

Let $x_2 \coloneqq \frac{2.5}{\alpha}\ln(\frac{4}{\alpha})$. Then,

$$f_2(x_2) = \alpha\frac{3}{5\ln(4/\alpha)}\ln\left(\frac{2.5}{\alpha}\ln\left(\frac{4}{\alpha}\right) + 1\right).$$

One can numerically verify that $\frac{3}{5\ln(4/\alpha)}\ln\left(\frac{2.5}{\alpha}\ln\left(\frac{4}{\alpha}\right)+1\right) < \frac{3}{4}$ for all $\alpha \in (0, 1/4)$. Therefore, $f(x_2) < \frac{3}{4}\alpha$. Choosing $x_0 = \max\{x_1, x_2\}$, we get $f(x) < \alpha$ for all $x > x_0$. All that remains is to show that $x_0 \leq \frac{2.5}{\alpha}\ln\frac{\beta}{\alpha}$. To see this, note that

$$x_1 = \frac{2}{\alpha}\ln\beta < \frac{2.5}{\alpha}\ln\beta < \frac{2.5}{\alpha}\ln\frac{\beta}{\alpha} \quad \forall\beta > 1 \text{ and } \alpha \in (0, 1/4).$$

Similarly,

$$x_2 = \frac{2.5}{\alpha}\ln\frac{4}{\alpha} \leq \frac{2.5}{\alpha}\ln\frac{\beta}{\alpha} \quad \forall\beta \geq 4 \text{ and } \alpha \in (0, 1/4).$$

Therefore, for all $\beta \geq 4$ and $\alpha \in (0, 1/4)$, we have $\max\{x_1, x_2\} \leq \frac{2.5}{\alpha}\ln\frac{\beta}{\alpha}$, and hence $f(x) < \alpha$ for all $x > \frac{2.5}{\alpha}\ln\frac{\beta}{\alpha}$.

### D.4 Proof of Theorem 5.3

In this section, we prove a slightly more general variant of Theorem 5.3. Recall that `QBClusterH` arbitrarily orders items in $\mathcal{N}$ before the clustering begins. Then, in each phase, it discovers one *pure* cluster whose identity is determined by the anchor chosen in that step. This means that clusters are discovered in arbitrary order. However, the size of the cluster found in phase $j$ determines how many queries will be needed in subsequent phases. For example, if $|\mathcal{C}_1^j|$ is large, $|\mathcal{N}^{j+1}|$ will be smaller, reducing the number of items to be clustered in the next phase. Therefore, the worst case happens when clusters are discovered in the increasing order of their sizes. This is the worst-case expression that is derived in the result below.

**Theorem D.1.** *For any* `QBC` *instance* $\nu = (n, \rho, k, p, q)$ *with* $k \geq 2$ *clusters, let* $n_i$ *denote the number of items in the* $i^{th}$ *ground-truth cluster. Assume without loss of generality that the clusters are indexed such that* $n_1 \leq n_2 \leq \ldots \leq n_k$. *Further, assume that* $n_1 \geq 2$. *Then, with probability at least* $1 - \delta$, `QBClusterH` *makes at most* $\tau$ *queries such that,*

$$\tau = O\left(\sum_{j=1}^{k} \frac{n - \sum_{i=1}^{j-1} n_i}{(p-q)^2} \ln\left(\frac{(n - \sum_{i=1}^{j-1} n_i)^2 j(j+1)}{\delta(p-q)^2}\right)\right).$$

*Proof.* Let $\hat{n}_j = |\mathcal{C}_1^j|$, where recall that $\mathcal{C}_1^j$ is the pure cluster returned by `SplitItemsH` in the $j^{th}$ phase. `QBClusterH` then requests `SplitItemsH` to cluster $\tilde{n}_{j+1} = n - \sum_{i=1}^{j} \hat{n}_i$ items in the $(j+1)^{th}$ iteration. This takes at most $O\left(\frac{\tilde{n}_{j+1}}{(p-q)^2} \ln \frac{\tilde{n}_{j+1}^2 j(j+1)}{\delta(p-q)^2}\right)$ queries with probability at least $1 - \delta^j$ by Theorem 5.2 if all previous calls to `SplitItemsH` have succeeded.

We argued in the proof of Theorem 5.1 that, with probability at least $1 - \delta$, `QBClusterH` runs for exactly $k$ phases, successfully isolating one pure cluster from the rest in each phase. This recovers the correct $k$ ground-truth clusters at the end. In the worst case, clusters are discovered in increasing order of their sizes, making $\hat{n}_i = n_i$ for $i = 1, 2, \ldots, k$. Adding the sample complexity of each phase gives us the sample complexity of `QBClusterH`. □

Theorem 5.3 is a special case of this result. For instance, one can obtain it by setting $n_1 = n_2 = \ldots = n_{k-1} = 2$, $n_k = n - 2(k-1)$, and replacing the $j$ inside ln with its maximum value $k$.

## E MORE DETAILS ABOUT `SplitItemsS` AND `QBClusterS`

In this section, we provide more details about `SplitItemsS` and `QBClusterS`. Let us start with `SplitItemsS`. We borrow a more efficient way to solve the best-arm identification problem from Kaufmann et al. (2016). The solution uses a test based on the so-called *SGLRT stopping rule*[5].

The test goes as follows. Suppose $X \sim$ Bernoulli($\mu_1$) and $Y \sim$ Bernoulli($\mu_2$) are independent, and we have paired samples $(x_t, y_t)_{t \geq 1}$ from these random variables. Let $\hat{\mu}_1(t) = \frac{1}{t}\sum_{s \leq t} x_s$ and $\hat{\mu}_2(t) = \frac{1}{t}\sum_{s \leq t} y_s$ be the empirical means at time $t$. For a given $\epsilon \in (0, 1)$, the test terminates when

$$\frac{1}{2}\left[d\left(\hat{\mu}_1(t), \frac{\hat{\mu}_1(t) + \hat{\mu}_2(t)}{2}\right) + d\left(\hat{\mu}_1(t), \frac{\hat{\mu}_1(t) + \hat{\mu}_2(t)}{2}\right)\right] > \frac{2}{t} \ln\left(\frac{t(\ln(3t))^2}{\epsilon}\right), \tag{17}$$

where $d(x, y)$ is the binary relative entropy function defined in Section 3. Let $\xi$ be the random termination time. By checking if $\hat{\mu}_1(\xi) > \hat{\mu}_2(\xi)$, the test correctly accepts/rejects the hypothesis $\mu_1 > \mu_2$ with probability at least $1 - \epsilon$.

Kaufmann et al. (2016) showed that this test is equivalent to the following procedure. Obtain paired samples $\{x_t, y_t\}_{t \geq 1}$. Let the confidence interval for $\mu_1$ and $\mu_2$ at time $t$ be given by $[l_1(t), u_1(t)]$ and $[l_2(t), u_2(t)]$,

[5] SGLRT stands for Sequential Generalized Likelihood Ratio Test

respectively. Then, for $j = 1, 2$, set

$$l_j(t) = \inf \left\{ z < \hat{\mu}_j(t) : td(\hat{\mu}_j(t), z) \leq \ln \frac{2t(\ln(6t))^2}{\epsilon} \right\},$$
$$u_j(t) = \sup \left\{ z > \hat{\mu}_j(t) : td(\hat{\mu}_j(t), z) \leq \ln \frac{2t(\ln(6t))^2}{\epsilon} \right\},$$

where $\hat{\mu}_1(t)$ and $\hat{\mu}(t)$ are the empirical estimates of $\mu_1$ and $\mu_2$ using the first $t$ samples. The test terminates when the confidence intervals separate out, or in other words, when $l_1(t) > u_2(t)$ or $l_2(t) > u_1(t)$. See the proof of Lemma 11 in Kaufmann et al. (2016) for more details.

This test is asymptotically optimal in the sense that it satisfies $\mathrm{P}\left(\limsup_{\epsilon \to 0} \frac{\xi}{\ln(1/\epsilon)} \leq \frac{2(1+\eta)}{d(\mu_1, \mu_2)}\right) = 1$ for all $\eta > 0$ (Kaufmann et al., 2016).

In our context, let $\mathcal{N}$ be the set of items to be grouped into two clusters. Define $\psi(t, \delta) = \ln \frac{t(\ln(3t))^2 |\mathcal{N}|^2}{\delta}$ and let

$$l_{ai}(t) = \inf\{z < \hat{\mu}_{ai}(t) : td(\hat{\mu}_{ai}(t), z) \leq \psi(2t, \delta)\},$$
$$u_{ai}(t) = \sup\{z > \hat{\mu}_{ai}(t) : td(\hat{\mu}_{ai}(t), z) \leq \psi(2t, \delta)\}.$$

For the $l_{ai}(t)$ and $u_{ai}(t)$ defined above, the following statement is a simple consequence of Lemma 4 in Kaufmann and Kalyanakrishnan (2013).

$$\mathrm{P}(\alpha_{ai} \in [l_{ai}(t), u_{ai}(t)]) \geq 1 - \frac{\delta}{t(\ln(6t))^2 |\mathcal{N}|^2}.$$

Taking a union bound over $t \geq 1$, we get

$$\mathrm{P}(\forall t \geq 1, \alpha_{ai} \in [l_{ai}(t), u_{ai}(t)]) \geq 1 - \frac{\delta}{|\mathcal{N}|^2}.$$

As this is true for arbitrary choices of anchor $a$ and item $i \in \mathcal{N} \backslash \{a\}$, we can again take a union bound and get

$$\mathrm{P}(\forall t \geq 1, \forall i \in \mathcal{N} \backslash \{a\}, \;\; \alpha_{ai} \in [l_{ai}(t), u_{ai}(t)]) \geq 1 - \delta,$$

where $a$ is a randomly chosen anchor item. Therefore, these confidence intervals are correct with high probability, as in Lemma 4.1.

`SplitItemsS` replicates `SplitItemsH` as is, except that it uses the confidence intervals described above for all calculations, thereby implementing the test based on the SGLRT stopping rule in Meta-Algorithm 1. Given the high-probability validity of these confidence intervals (argued above), the proof of correctness of `SplitItemsS` follows along the same lines as the proof of Theorem 5.1. The fact that these intervals are asymptotically optimal explains the lower sample complexity of the algorithm, especially when $p$ and $q$ change with $n$.

`QBClusterS` is defined exactly as Algorithm 2 except that it calls `SplitItemsS` instead of `SplitItemsH` at each iteration.