Crowdsourced Clustering via Active Querying: Practical Algorithm with Theoretical Guarantees

Yi Chen [1] **Ramya Korlakai Vinayak** [1] **Babak Hassibi** [2]

Abstract

We propose a novel, practical, simple, and computationally efficient active querying algorithm for crowdsourced clustering that does not require knowledge of unknown problem parameters. We show that our algorithm succeeds in recovering the clusters when the crowdworkers provide answers with an error probability less than $1/2$ and provide sample complexity bounds on the number of queries made by our algorithm to guarantee successful clustering. While the bounds depend on the error probabilities, the algorithm itself does not require this knowledge. In addition to the theoretical guarantees, we implement and deploy the proposed algorithm on a real crowdsourcing platform to characterize its performance in real-world settings.

1. Introduction

Crowdsourcing, which refers to using a crowd of potentially non-expert humans to obtain information useful for downstream tasks, has become one of the most popular ways of collecting labeled datasets for supervised learning tasks (Sorokin & Forsyth, 2008; Raykar et al., 2010). There is an abundant amount of data, e.g., billions of images and texts, that can be readily scraped from the internet. However, most of these datasets are unlabeled, and it is unclear what structures might exist in them. Crowdsourcing can be a very useful resource for exploring structure in such data (Welinder et al., 2010).

We consider the problem of *crowdsourced clustering* – finding clusters in a dataset with unlabeled items by querying pairs of items for similarity: "Are items i and j from the same cluster?" Viewing the items to be clustered as nodes in a graph whose edges have not been observed, we have a problem of clustering a graph with an access to a noisy oracle that can answer pairwise similarity queries.

A passive strategy for clustering in this scenario is to query all the pairs (i.e., edges), a random subset of pairs (Vinayak & Hassibi, 2016), or a specifically constructed subset of pairs (Gomes et al., 2011; Ibrahim & Fu, 2021) and then perform graph clustering. A major hick-up of such passive strategies is that they can only recover relatively large clusters. This is because the existing polynomial time graph clustering algorithms can only recover clusters at least $\Omega(\sqrt{n})$ in size. This is related to the well-known hidden clique problem, where the goal is to find a hidden clique of a certain size in a random graph of size n. It is currently an open conjecture that there is no polynomial time passive algorithm that can recover a hidden clique of size smaller than $\sqrt{n}$. Active clustering, on the other hand, can potentially transcend this barrier. In this paper, we study how to cluster a set of items using these crowdsourced pairwise comparison queries in an active manner that overcomes the issue of recovering small clusters.

Our Contributions: We propose an *active crowdsourced clustering algorithm* that *does not rely on any unknown problem parameters*. It is computationally efficient, simple to implement, and capable of recovering clusters regardless of their sizes. We also provide an analysis of the proposed algorithm and sample complexity bound that guarantees the algorithm's success in recovering all the clusters with high probability (with failure probability decaying as $1/\text{poly}(n)$). A key observation is that when the crowdworkers are better than random guessers (i.e., the error probability is less than $1/2$), the problem of deciding whether two items, i and j, belong to the same cluster can be recast as a problem of inferring if the true parameter of a Bernoulli random variable is above or below $1/2$. Our algorithm is inspired by the finite law of iterated logarithms (LIL) for multi-arm bandits (Jamieson et al., 2014; Heckel et al., 2019). We implement and deploy the proposed algorithm on a real crowdsourcing platform and evaluate its performance in the real-world settings. Based on both the theoretical and the empirical results, we observe that the total number of

[1]Department of Electrical Computer Engineering, University of Wisconsin-Madison, Madison, WI, USA [2]Department of Electrical Engineering, California Institute of Technology, Pasadena, CA, USA. Correspondence to: Yi Chen <yi.chen@wisc.edu>, Ramya Korlakai Vinayak <ramya@ece.wisc.edu>.

AI & HCI Workshop at the 40^{th} International Conference on Machine Learning (ICML), Honolulu, Hawaii, USA. 2023.

queries made by active clustering algorithm is order-wise better than random querying. However, the advantage of our algorithm is most conspicuous when the datasets have small clusters, which is a hard scenario for passive clustering algorithms. For datasets with large clusters, which are easier settings, passive querying strategy of randomly querying a subset of edges followed by graph clustering can often be query efficient in practice. To the best of our knowledge, this is the first demonstration of active clustering algorithm working in practice (beyond simulations). We make our dataset publicly available and also the codebase to enable further development and deployment of such systems.

Related Literature: Many prior works that consider the problem of crowdsourced clustering using pairwise similarity queries employ passive strategy with either a deterministic pattern fixed a priori (Gomes et al., 2011; Ibrahim & Fu, 2021) or randomly chosen queries (Vinayak et al., 2014; Vinayak & Hassibi, 2016). Another related line of work is entity resolution in databases where the goal is to find data records that represent the same real-world entities. There is a rich line of work in this area (see (Wang et al., 2012; Vesdapunt et al., 2014; Verroios & Garcia-Molina, 2015) and the references there in) that use heuristics-based crowdsourcing algorithms to resolve entities. Most of these works assume that there is a machine generated similarity matrix between different data records and use this information to decide which data records to query. (Mazumdar & Saha, 2017a;c) provide analysis for some of the popular heuristics and algorithms when side information is present.

A closely related work is (Yun & Proutiere, 2014), which focuses on the setting with fixed number of clusters of large sizes, i.e., $\Theta(n)$, which is an easier setting for clustering. They also assume that the number of clusters are known a priori. The authors propose spectral clustering-based algorithms and theoretically analyzed both passive and adaptive querying strategies.

Another closely related work is (Mazumdar & Saha, 2017b), which also considers active clustering by crowdsourcing. The key differences from our setting are that they forbid repeated querying of a pair of items, and they assume that the algorithm is aware of the error probability p. Two algorithms are proposed in (Mazumdar & Saha, 2017b), one that achieves a near-optimal query complexity but is computationally hard, while the other is computationally efficient but with sub-optimal query complexity. In particular, the query complexity of the computationally efficient algorithm grows quadratically in the number of clusters K, which is very costly when there are many small clusters. Both the algorithms require the cluster sizes to be at least $\Omega(\log n)$. Furthermore, both the algorithms in (Mazumdar & Saha, 2017b) require knowledge of the error probability p, which makes it difficult to deploy in practical crowdsourcing setups. Under similar assumptions of forbidden repeated queries and assuming the knowledge of error probability another recent work (Mukherjee et al., 2022) provides efficient algorithms to recover clusters of size at $\Omega(k \log n)$ for a fixed error probability.

In contrast, we consider the setting where the error probabilities are unknown and repeated querying of the same pair of items to different crowdworkers is allowed. Our algorithm is simple to implement, computationally efficient, capable of recovering clusters regardless of their sizes, and agnostic of the number of clusters and error probabilities, while achieving near optimal (up to logarithmic factors) query complexity. *One of the key contributions of our work is addressing how to deal with the unknown parameters, which is essential for making the algorithm practical in real-world settings.* We demonstrate that repeating queries is in fact practical and effective in deciding cluster memberships by deploying our algorithm on a popular crowdsourcing platform and running crowdsourced clustering tasks with real crowdworkers. The goal of repeated querying in our setting is not to drive the empirical error to 0 but instead to guarantee that either the lower confidence bound on the unknown true parameter is above or the upper confidence is below $1/2$.

2. Problem Setup

In this section we describe the problem setup, the model and the assumptions. Consider n items that belong to K disjoint clusters. Consider a pool of crowdworkers who provide noisy answers to pairwise queries of the type: "Are items i and j from the same cluster?" Let Query(i, j) denote such a pairwise query. Let $X_{ij}(s)$ denote the answer provided by crowdworker s to Query(i, j). In particular, $X_{ij}(s) = 1$ if the answer to Query(i, j) by worker s is "yes" and $X_{ij}(s) = 0$ if the answer is "no". For any pair of items i and j, and any positive integer m, let $\bar{X}_{ij}(m)$ denote the average of m independent answers to the Query(i, j), i.e.,

$$\bar{X}_{ij}(m) := \frac{1}{m} \sum_{s=1}^{m} X_{ij}(s). \tag{1}$$

For any item j, we use the notation cluster(j) to denote the cluster that contains item j.

Suppose the workers were perfect, then with $\Theta(nK)$ queries, we could assign all the items to the correct clusters. However, the workers on crowdsourcing platforms are not experts and hence maker errors.

Model: We consider the following two-coin model for worker errors. When two items i and j are from the same cluster, for all workers s,

$$X_{ij}(s) = \begin{cases} 1 & \text{with probability } p, \\ 0 & \text{with probability } 1-p. \end{cases}$$

When i and j are not from the same cluster, for all workers s,

$$X_{ij}(s) = \begin{cases} 1 & \text{with probability } q, \\ 0 & \text{with probability } 1-q. \end{cases}$$

We note that this is similar to the Stochastic Block Model (SBM) used in analyzing graph clustering or community detection problems (Holland et al., 1983; Condon & Karp, 2001), where $X_{ij} = 1$ denotes an edge and $X_{ij} = 0$ denotes no edge between two *nodes* i and j.

Assumptions: We assume that the answers given by different workers are independent, i.e., $X_{ij}(s)$ and $X_{ij}(s')$ are independent when $s \neq s'$. We also assume that while the workers make errors, they are better than random guessers, i.e., $1 \geq p > \frac{1}{2} > q \geq 0$.

3. Active Clustering Algorithm

In this section, we present the active querying algorithm for crowdsourced clustering. The algorithm proceeds by building clusters from scratch. Initially, we have a set of items to be clustered. A randomly chosen item is set as its own cluster at the beginning. We start by picking an item i that is yet to be clustered and query it with existing clusters to decide its membership. To decide the membership of item i with cluster(j), an item is picked at random from cluster(j) and the Query(i, j) is repeated with different crowdworkers until membership of i can be established with confidence. If it does not belong to any of the existing clusters, then it starts a new cluster. This process continues until all the items are clustered. *The key challenge is to decide the cluster memberships with guaranteed confidence when the error probabilities are unknown.* We propose using the finite law of iterated logarithms (Jamieson et al., 2014) to obtain time varying confidence bounds, which are monotonically decreasing in time t and are valid for all t. The detailed pseudocode for the algorithm is given in Algorithm 1.

Algorithm 1 Active Crowdclustering by Crowdsourcing; withou the knowledge of p and q

1: **Input:** set of items to be clustered V, $\zeta \in (0,1)$, $\delta \in (0, \log(1+\zeta)/e)$
2: Pick $i \in V$ randomly
3: Initialize $\mathcal{C} = \{\mathcal{C}_1 := \{i\}\}$
4: $V \leftarrow V \setminus \{i\}$
5: **while** $V \neq \emptyset$ **do**
6:   Pick $v \in V$ randomly
7:   **for** $k \in [|\mathcal{C}|]$ **do**
8:     Pick $u \in \mathcal{C}_k$ randomly
9:     $\bar{X}_{vu}(0) \leftarrow 0$
10:     **for** each time step t **do**
11:       $X_{vu}(t) \leftarrow Query(v, u)$ {Query to a distinct crowdworker}
12:       $\bar{X}_{vu}(t) \leftarrow \frac{t-1}{t}\bar{X}_{vu}(t-1) + \frac{1}{t}\bar{X}_{vu}(t)$ {Cumulative empirical average of the answers}
13:       $\psi(t) \leftarrow (1+\sqrt{\zeta})\sqrt{\frac{1+\zeta}{2t}\log\left(\frac{(1+\zeta)t}{\delta}\right)}$ {Confidence interval}
14:       **if** $\bar{X}_{uv}(t) - \psi(t) > \frac{1}{2}$ **then**
15:         $\mathcal{C}_k \leftarrow \mathcal{C}_k \cup \{v\}$ {Assign v to C_k}
16:         $V \leftarrow V \setminus \{v\}$
17:         **goto** Line 5
18:       **end if**
19:       **if** $\bar{X}_{uv}(t) + \psi(t) < \frac{1}{2}$ **then**
20:         **goto** Line 7 with k increments by 1 {Move to next cluster}
21:       **end if**
22:     **end for**
23:   **end for**
24:   **if** v is not assigned to $C_k, \forall k$ **then**
25:     $C \leftarrow C \cup \{v\}$ {Start a new cluster with v}
26:     $V \leftarrow V \setminus \{v\}$
27:   **end if**
28: **end while**

4. Performance Guarantees

We analyze Algorithm 1 under the error model inspired by the SBM and the assumptions described in Section 2. Let $\Delta = \frac{1}{2}\min\{p - \frac{1}{2}, \frac{1}{2} - q\}$. We can guarantee the following performance for Algorithm 1 under the assumptions on our model:

Theorem 4.1 (Main Theorem)**.** *Algorithm 1 succeeds in recovering all the clusters exactly with at most* $\mathcal{O}\left(\frac{nK}{\Delta^2}\log n \log \frac{1}{\Delta}\right)$ *queries overall, with high probability.*

Note that high probability here refers to an upper bound on the probability of failure that decays as $1/\text{poly}(n)$. While the bound on the query complexity is a function of the problem parameters, K, p and q, *the algorithm itself does not need to know these parameters*.

If $p < 1$ and $q > 0$, $\Omega\left(\frac{1}{\Delta^2}\right)$ repetitions per query are needed and hence $\Omega\left(\frac{nK}{\Delta^2}\right)$ queries are necessary for Algorithm 1 to succeed with probability at least $3/4$. Hence the upper bound on the number of queries is optimal up to log factors. Furthermore, the extra price that Algorithm 1 pays for not knowing Δ (as compared to the sample complexity if we knew p and q) is a $\log\left(\log\frac{b_2}{\Delta}\right)$ term. Comparing our bounds with the asymptotic lower bounds in (Yun & Proutiere, 2014) and the lower bounds in (Mazumdar & Saha, 2017b), we note that our bounds are within log factors of optimal query complexity. In particular, for $p = 1 - q$, the lower bound is of the order $\Omega(\frac{nK}{\Delta^2})$.

The following corollary provides the general version of the main theorem,

Corollary 4.2. *For any* $\zeta \in (0,1)$, $c \geq 3$, $\delta = \frac{\delta'}{n^c} \in$

$(0, \log(1+\zeta)/e)$, *then with probability at least* $1 - 1/n$, *Algorithm 1 succeeds in recovering all the clusters exactly and the total number of queries made is upper bounded by* $\mathcal{O}\left(nK\frac{b_1}{\Delta^2}\log\left(\frac{n^c}{b_3\delta'}\log\frac{b_2}{\Delta}\right)\right)$, *where* $b_1 = 3,\ b_2 = (1+\zeta)^2,\ b_3 = \frac{1}{(2(1+\sqrt{\zeta}))^3}$.

Note that c, δ and ζ can be chosen such that the failure probability decays as $1/\text{poly}(n)$ and this does not require the knowledge of error probabilities p and q. We further note that the choice of δ and ζ also affects the size of confidence interval $\psi(t)$ and hence the number of queries made by Algorithm 1. The bound presented in Theorem 4.1 is obtained by choosing $c = 4$ and $\zeta = 0.1151$.

4.1. Discussion

In this section we reflect on Algorithm 1, discuss extensions, and compare with passive querying.

4.1.1. General bound with confusion matrix

While the main result is presented with a simple two-coin error model where the probabilities p and q capture the intra- and inter-cluster error probabilities respectively, the analysis can be extended to more general setting. Let $P \in [0,1]^{n\times n}$ be the *confusion matrix* associated with the n items being clustered, where each entry P_{ij} is the probability of the answer to Query(i,j) is 1. The assumption that the workers are better than random guessers in this general case implies that $P_{ij} > 1/2$ when i and j are from the same cluster and $P_{ij} < 1/2$ otherwise. Define $\Delta_{ij} := |P_{ij} - 1/2|$. The proof of Theorem 4.1 can be modified to obtain the following upper bound on the total number of queries made by Algorithm 1 in the general case for successfully recovering clusters with high probability,

$$\sum_{i,j:\{i,j\}\in\Omega} \frac{b_1}{\Delta_{ij}^2}\log\left(\frac{n^c}{b_3\delta}\log\frac{b_2}{\Delta_{ij}}\right),$$

where Ω is the set of queries made and $|\Omega| \le nK$.

4.1.2. Modifications in querying

In Algorithm 1, to decide if item i belongs to cluster(j), a random item j is picked as a representative from that cluster and Query(i, j) is repeated until a decision can be made about the membership of i. Instead of repeating Query(i, j) with the same representative item, we could pick a random element from cluster(j) for each repetition. For the assumed model, there is no statistical change from our assumptions and hence the guarantee provided by Theorem 4.1 still holds. For the general confusion matrix case described above, careful book keeping is needed as instead of $\mathbb{E}(\bar{X}_{ij}(t)) = P_{ij}$, we will have $\mathbb{E}(\bar{X}_{ij}(t)) = \frac{1}{t}\sum_{s=1}^{t} P_{ij}(s)$. In practice, switching to different representative elements from a cluster could help avoid being stuck with a bad representative picked by chance in the beginning from cluster(j).

4.1.3. Active vs. Passive Queries

Here we discuss the pros and cons of active querying for crowdsourced clustering when compared to using passive queries. Crowdsourced clustering using passive queries has been previously approached with a two step process (Gomes et al., 2011; Vinayak & Hassibi, 2016; Ibrahim & Fu, 2021). In the first step, a random or a carefully designed predetermined subset of the $\binom{n}{2}$ pairs of items, say $\lceil r\ \binom{n}{2}\rceil$ with $r \in (0,1]$ are queried to partially fill a noisy adjacency matrix. In the second step, a graph clustering algorithm runs on it. We focus on computationally efficient (polynomial time) clustering algorithms for this discussion.

- **Active querying succeeds regardless of cluster sizes:** Computationally efficient graph clustering algorithms, e.g., spectral clustering (McSherry, 2001; Rohe et al., 2011), convex clustering algorithms (Chen et al., 2014; Vinayak et al., 2014; Jalali et al., 2015), have a bottleneck in terms of the size of the smallest cluster that can be recovered. In particular, the smallest cluster has to be sufficiently large, i.e., at least $\Omega(\sqrt{n})$, to be recovered. This bottleneck of on the minimum cluster size is conjectured to also be necessary for any polynomial time graph clustering algorithm (related to the *hidden clique conjecture*). Therefore, using any known computationally efficient graph clustering algorithms with passive querying can only recover clusters of size at least $\Omega(\sqrt{n})$. On the contrary, the sufficient condition for exact recovery of the clusters (Theorem 4.1) using Algorithm 1) which is computationally efficient, holds regardless of the cluster sizes. This is also illustrated by our experiments on real crowdsourcing platform (see Section 6, Table 3).

- **Active querying algorithm is free of model parameters:** For passive querying, the knowledge of $p - q$ and $n_{\min}$, or other side information, is needed to a priori set the number of queries to be made that can guarantee the exact recovery of clusters (unless we make all $\binom{n}{2}$ queries). On the other hand, our active querying algorithm does not require the knowledge of p, q, K or the cluster sizes ahead of time to guarantee exact recovery. The only assumption is that the workers are better than random guessers ($p > 1/2 > q$).

- **Active vs. passive querying sample complexity:** The sufficient number of queries to guarantee exact recovery of clusters for our active querying algorithm is $\mathcal{O}\left(\frac{nK}{\Delta^2}\log n \log\frac{1}{\Delta}\right)$, where $\Delta = \min\{p - \frac{1}{2}, \frac{1}{2} - q\}$. Let us compare this to the state of the art sufficient conditions for exact recovery of clusters via graph clustering for SBM (see (Chen et al., 2014; Vinayak et al., 2014; Jalali et al., 2015) & references there in).

- When the smallest cluster is $\Theta(\sqrt{n})$: Passive graph clustering can guarantee exact recovery using at most $\mathcal{O}\left(n^2/(p-q)^2\right)$ random queries. In the case of $\sqrt{n}$ clusters of size $\Theta(\sqrt{n})$, passive graph clustering takes at most $\mathcal{O}\left(n^2/(p-q)^2\right)$ random queries to guarantee success while our algorithm takes at most $\mathcal{O}\left(\frac{n^{1.5}}{\Delta^2}\log n \log \frac{1}{\Delta}\right)$. So, there is room for $\sqrt{n}$ gain in sample complexity for active clustering.
- When the smallest cluster is very large, i.e, when all the clusters are of size $\Theta(n)$: In this case, since K is a constant, our algorithm takes at most $\mathcal{O}\left(\frac{n}{\Delta^2}\log n \log \frac{1}{\Delta}\right)$. Passive graph clustering algorithms can obtain correct clustering by using at most $\mathcal{O}\left(n(\log n)^2/(p-q)^2\right)$ random queries. So, the relative advantage of active clustering might be limited here and might not kick in until the dataset sizes are very large, depending on the hidden constants in these bounds. From our experiments on real crowdsourcing platform (see Section 6, Table 2), we observe that passive querying followed by graph clustering can provide very good clustering outcomes with much fewer queries compared to active clustering when the cluster sizes are large. However, the active clustering seems to pick up more granular differences within each cluster. See Section 6.1 for more details.

In summary, our active clustering algorithm 1 has advantages in terms of being agnostic to model parameters and that its success does not depend on a cluster size bottleneck. Active clustering is competitive or better than random queries order-wise, but the advantages can be realized in practice in the regime when the cluster sizes are small which is a hard scenario for a passive algorithms.

We would like to emphasize that the challenge we address in this paper concerns with how to decide a cluster membership without knowing p and q. Our routine that uses time varying confidence intervals can make the second phase of the algorithms in (Mazumdar & Saha, 2017b) practical. However, the initial phase in their algorithms would still need the knowledge of error probability.

5. Simulations: Passive vs. Active Querying

Here we compare the performance of active and passive clustering algorithms under easy and difficult settings on simulated data. With $n = 900$ items to be clustered, we consider two scenarios varying the number of clusters $K^\star$ with each cluster of equal size: (1) *Easy case* with $K^\star = 3$ with *large cluster sizes*, of the order of, $\Theta(n)$, (2) *difficult case* with $K^\star = 30$ with *small cluster sizes* around the threshold of $\Theta(\sqrt{n})$. To simulate crowdworkers' answers, we construct a confusion matrix $P \in [0,1]^{n\times n}$ for each of the scenarios. Each entry P_{ij} denotes the probability of observing an edge between item i and j. We draw $P_{ij} \sim$ Uniform$[0.6, 0.85]$ if item i and j belong to the same cluster, otherwise $P_{ij} \sim$ Uniform$[0.1, 0.35]$. We use variation of information (VI) (Meila, 2007) to measure the difference between the output clustering and ground truth clustering. Note that VI ≥ 0 and the smaller the VI the better with VI $= 0$ indicating perfect match.

Table 1. VI for the clustering outcome and the total number of queries, denoted as TQ in the column header, made after running Algorithm 1 and passive clustering on simulated datasets.

Method	VI↓ ($K^\star = 3$)	TQ	VI↓ ($K^\star = 30$)	TQ
Active 1 (this paper)	**0.09 ± 0.06** (K = 5)	41,014	**0.52 ± 0.07** (K = 40)	277,370
(Yun & Proutiere, 2014) Adaptive	2.74 ± 0.37 (K = 14)	52,225	3.63 ± 0.33 (K = 24)	294,530
(Yun & Proutiere, 2014) Passive	3.41 ± 0.15 (K = 14)	52,225	3.84 ± 0.32 (K = 13)	294,530
K-means, Passive	0.31 ± 0.35 (K = 3)	41,014	3.4 ± 0 (K = 1)	277,370
(Vinayak et al., 2014) +Kmeans, Passive	**0 ± 0 (K = 3)**	41,014	3.4 ± 0 (K = 1)	277,370
Spectral, Passive	**0.04 ± 0 (K = 3)**	41,014	3.4 ± 0 (K = 1)	277,370
(Vinayak et al., 2014) +Spectral, Passive	**0.07 ± 0.22 (K = 3)**	41,014	3.4 ± 0 (K = 1)	277,370

We run our active clustering algorithm 1 and adaptive algorithm in (Yun & Proutiere, 2014). We also tried running the active clustering algorithm in (Mazumdar & Saha, 2017b), but it failed to run as the initial stage did not yield any clusters even after search over hyper-parameters. For passive algorithms, we ran the random query algorithm from (Yun & Proutiere, 2014), k-means, spectral clustering (McSherry, 2001), and convex algorithms (Vinayak & Hassibi, 2016). Each algorithm is run 10 times and the results are shown in Table 1, and discussed below:

- In the *easy setting* with $K = 3$ clusters (large clusters),

Table 2. The percentage of node pairs in error, variation of information (VI) for the clustering outcome, the average number of repetitions per query, and the total number of queries, denoted as TQ, made after running Algorithm 1 on Dogs3 dataset run with the help of real crowdworkers on AMT. The 2nd row shows the best clustering result from (Vinayak & Hassibi, 2016) (in Table 4) for the same dataset for passive clustering.

METHOD	PAIR ERR.%	VI ↓	MEAN T	TQ
ACTIVE (FROM SCRATCH)	12.5%	1.85	21.98	$43,572$
PASSIVE (VINAYAK & HASSIBI, 2016)	20%	$\mathbf{0.23}$	N/A	$\mathbf{17,626}$
ACTIVE, INITIALIZED (NON-RANDOM REPEAT)	14.27%	1.42	23.20	$29,189$
ACTIVE, INITIALIZED (RANDOM REPEAT)	14.14%	1.12	22.14	$28,824$

the passive clustering is on par with our active clustering algorithm 1. In particular, the spectral clustering and convex clustering algorithms obtain nearly perfect clustering.

- In the *difficult setting* (small clusters), our active clustering algorithm 1 outperforms all the other algorithms. This setting is around the threshold where the passive algorithms struggle in recovering the clusters. We also note that the adaptive algorithm in (Yun & Proutiere, 2014) is designed to work for the easy setting where the cluster sizes are at least $\Theta(n)$. So, it is not surprising that it does not perform well in the difficult setting. However, both the active and passive version of the algorithms in (Yun & Proutiere, 2014) do not perform well in the easy setting either. This is because they rely on estimating a homogeneous error parameter for each block to decide a priori the number of repeated queries to make with each cluster which affects the accuracy of decisions of cluster memberships. This highlights the problem of relying on problem parameters for deciding cluster memberships.

6. Experiments Using Real Data

In this section we present experimental results using real datasets and both synthetic and real crowdworkers.

6.1. Experiments on a real crowdsourcing platform.

For experiments with real crowdworkers, we use Amazon Mechanical Turk ((AMT)) platform where crowdworkers answered pairwise queries (Figure 1). The instructions we provided is shown in Figure 3. We note that we did not enforce the gold standard questions. We used all the data we obtained and paid all the workers who participated regardless of accuracy. Histograms of worker error rates for all the experiments with real crowdworkers is shown in Figure 2.

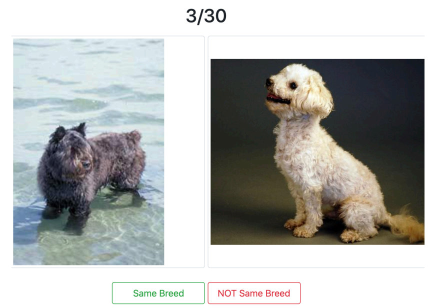

Figure 1. Sample of the pairwise queries displayed to the corwdworkers on Amazon Mechanical Turk (AMT).

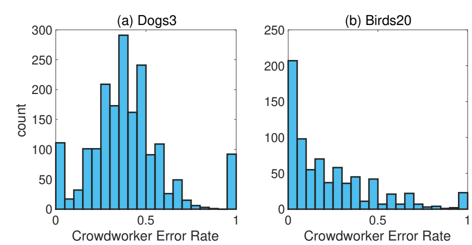

Figure 2. Histograms of corwdworker error rate on AMT for Dogs3 and Birds20 datasets.

The backend and frontend for this active querying system is implemented using Node.js, embedded Javascript, CSS and Bootstrap. We implemented a batched version of Algorithm 1 for efficiency in terms of time to run the experiments on AMT. Instead of querying one item at a time and waiting for its cluster membership to be decided, we maintain an active querying batch with 30 images to be queried (until the end where only a few items are remaining to be clustered). We also maintain a set of yet to be queried set and a clustered set. When a decision is arrived at for an image in the active querying batch as to which cluster it belongs to or to form a new cluster, it is moved out of the batch to clustered set and a randomly chosen image from yet to be queried set is added to the batch. Each crowdworker is shown 30 pairs of images to cluster (Figure 1 shows an example of a query).

In order to avoid excessive cost by repetition of *difficult* to cluster images, we set the maximum number of repetitions to 80. If an image took more than 80 queries to decide whether it belongs to a cluster and it happens for all the clusters, then it is considered a *difficult* to cluster image and put in a separate bucket of such hard images. We set $\zeta = 0.0001, \delta = 0.3$ for all the experiments unless specified

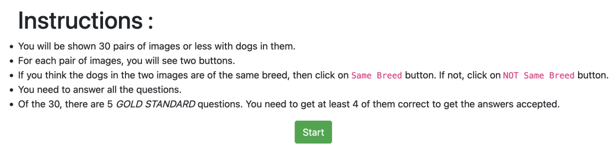

Figure 3. Sample of instructions shown for pair queries. Note that we did not enforce the gold standard questions. We used all the data we obtained and paid all the workers who participated in the tasks.

otherwise.

6.1.1. Easy setting: Large cluster sizes

Dogs3 dataset (Khosla et al., 2011; Vinayak & Hassibi, 2016) has 473 dogs of 3 different breeds (see Figure 4): Norfolk Terrier (172 images), Toy Poodle (151 images) and Bouvier des Flanders (150 images). This dataset has larger cluster sizes.

Figure 4. Sample images from the three clusters in the Dogs3 Dataset

We ran the following three experiments on AMT for the Dogs3 dataset:

1. Starting from no images being clustered (referred to as ***from scratch***).
2. Starting from initialized clusters where we start with the three clusters initialized with 50 images randomly chosen from respective breeds. When querying for an image i's membership with a cluster(j),
   (a) Choosing a random image from cluster(j) as a representative and repeating the same query to different crowdworkers (referred to as ***initialized, non-random repeat***).
   (b) Picking a randomly chosen image form cluster(j) for each repetition to different crowdworkers (referred to as ***initialized, random repeat***).

   The initialization and the order in which the images were picked to be added to the active query batch were the same for both the *non-random* and *random* repeat versions.

The results are summarized in Table 2. For comparison with passive clustering, we will refer to the results in (Vinayak & Hassibi, 2016) for the same dataset. A total of 134 images for *from scratch*, 104 images for *initialized non-random repeat* and 38 images for *initialized random repeat* experiments respectively remained as *difficult* to cluster in these experiments.

Comparing the results for *active from scratch* and the best passive clustering result from (Vinayak & Hassibi, 2016), we make the following observations. The clustering outcome for passive querying followed by graph clustering seems to significantly outperform active querying with just 40% of the number of queries. Recall from discussions in Section 4.1.3 that the theoretical bounds on total query complexity for active algorithm in large cluster regime is $\mathcal{O}\left(\frac{n}{\Delta^2}\log n \log\frac{1}{\Delta}\right)$ which when compared to the bound of $\mathcal{O}\left(n(\log n)^2/(p-q)^2\right)$ for passive clustering is only orderwise better marginally. The data sizes we are working with here might be too small for such a slight advantage to get reflected depending on the hidden constants in these bounds. We further note that the clustering outcome from *active from scratch* has overall 3 large clusters (corresponding to the 3 breeds), 5 very small clusters that pick up two groups of poodles that look very different from the rest, two groups of terriers that are slightly darker and those with ears pointed when imaged, and a group of Bouvier des Flanders and 6 outliers (with only one image per cluster). So, while the clustering outcome of Algorithm 1 overall does not match the ground truth of three clusters very well compared to passive querying, it does seem to capture more granular nuances in the images.

By comparing between the results for the set up of *initialized non-random repeat* and *random repeat*, we note that there is not a large difference in the percentage of pairs that were in error and the average number of repetitions made per query. An issue that could arise when the representative of a cluster is fixed as is the case in the *non-random repeat* setting, is that if we are unlucky to pick a bad/atypical example from the cluster as the representative, it can lead to either error or exceeding the difficult query repeat limit. Whereas, in the *random repeat* set up, this is usually ameliorated as a random representative is chosen for each repetition. This is also reflected in the clustering outcome where the random repeat setting performs slightly better than that of the non-random repeat. We note that in both the cases, if the image being queried itself is a difficult image, then it is hard to avoid large number of repetitions.

6.1.2. Hard setting: Small cluster sizes

Birds20 is a dataset we created using a subset of Caltech-UCSD Birds dataset (Wah et al., 2011). It has 125 images of birds from 20 different species: American Goldfinch (6), Arctic Tern (5), Baltimore Oriole (7), Blue Jay (4), Cardinal (10), Eared Grebe (3), Eastern Towhee (5), Fish Crow (4), Green Jay (6), Groove Billed Ani (6), Horned Puffin

Table 3. The percentage of node pairs in error, VI for the clustering outcome, the average number of repetitions per query, and the total number of queries, denoted as TQ, made after running Algorithm 1 and passive clustering on Birds20 dataset run with the help of real crowdworkers on AMT.

METHOD	PAIR ERR.%	VI ↓	MEAN T	TQ
ACTIVE, FROM SCRATCH	1.69%	**0.88** (K = 20)	12.34	15,160
PASSIVE FULL, (7750 EDGES ×1)	15.6%	1.64 ± 0.11 (K = 6)	N/A	7,750
PASSIVE SUBSET REPEAT, (5054 EDGES ×3)	18.4%	1.64 ± 0.13 (K = 11)	N/A	15,162

(5), House Sparrow (10), Laysan Albatross (5), Least Tern (5), Mallard (10), Pileated Woodpecker (4), Red Winged Blackbird (10), Rufous Hummingbird (5), White Breasted Kingfisher (10), and White Pelican (5). The number images in each species cluster is shown in the bracket. This dataset has very small cluster sizes and allows us to investigate the performance of Algorithm 1 in small-cluster-regime in practice.

We ran the following three experiments on AMT for the Birds20 dataset:

1. Active clustering with the batched implementation of Algorithm 1 starting from no images being clustered (referred to as ***from scratch***).
2. Passive querying followed by graph clustering with
   (a) All $\binom{125}{2} = 7750$ edges queried once (referred to as ***passive full***).
   (b) Randomly chosen subset of edges with each edge queried thrice (referred to as ***passive subset repeat***). We chose 5054 edges randomly so that with 3 repetitions it matches the total queries made in *active from scratch* setting and use majority voting to get the adjacency matrix.

   We ran k-means, spectral clustering (McSherry, 2001) and improved convex algorithm (Vinayak et al., 2014) (followed by k-means and spectral clustering) for graph clusteirng on the passively queried adjacency matrices. The results for these experiments with Birds20 dataset are summarized in Table 3. For passive clustering, we present the best results with the number of clusters that are resolved by the respective adjacency matrices.

Comparing the outcome of Algorithm 1 (*active from scratch*) with passive clustering, we make the following observations. In this small cluster regime, active Algorithm 1 provides much better clustering outcomes than the passive clustering. We note that Algorithm 1 recovered 20 clusters overall. In contrast, the adjacency matrices filled by *passive full* and *passive repeat* could only resolve 6 and 11 clusters respectively. This is due to the limitations of efficient clustering algorithms with respect to recovering small clusters (see Section 4.1.3).

Worker error rates, time and cost. The results presented in this section is based on the participation of $4,369$ crowdworkers on AMT. Our task was open to all crowdworkers on AMT with at least 500 human intelligent tasks (HITs) approved and with a HIT approval rate of at least 95%. The crowdworkers took on average $5.54s$ and $4.6s$ in the Dogs3 and Birds20 experiments respectively for *from scratch* experiments. The histogram of error rates for these experiments are shown in Figure 2. The crowdworkers took on average $5.47s$ and $4.86s$ to answer each pairwise question for the Dogs3 experiments for initialized with non-random repetition and initialized with random repetition respectively. Time taken per query for the passive querying experiments on Birds20 dataset were $3.73s$ for the passive full and $4.07s$ for the passive repeat set ups. We paid $\$0.30$ per task which involved 30 pairs of questions which roughly translates to $\$7.20/\text{hr}$ (with around $5s$ time per pair query). We also note that the AMT adds 40% additional fees for using their platform.

7. Conclusion

In this work, we considered the problem of clustering a set of items into disjoint clusters with the help of noisy crowdworkers who can answer pairwise comparison queries of type "Are items i and j in the same cluster?". We proposed a practical active clustering algorithm towards this goal and under mild assumptions, provided bounds on query complexity that guarantees the exact recovery of the clusters. The proposed active algorithm does not need the knowledge of any problem parameters, in particular the error probabilities, number of clusters or size of the clusters. We implemented this algorithm on a real crowdsourcing platform to demonstrate its efficacy and study its performance in the large cluster and small cluster regimes. While the theoretical bound on the query complexity is order-wise better for the active clustering algorithm, when the clusters are large, passive algorithms can, in fact, provide very good clustering outcomes with much fewer queries in practice. The main advantage of the active clustering algorithm seems to be in the case when there could be clusters of very small sizes that passive clustering algorithms will fail to recover. A hybrid approach that gets the best of both worlds would be useful to develop, and we leave it to future work.

References

AMT. Amazon mechanical turk. https://www.mturk.com/. Accessed: 2022-10-12.

Chen, Y., Jalali, A., Sanghavi, S., and Xu, H. Clustering partially observed graphs via convex optimization. *Journal of Machine Learning Research*, 15(1):2213–2238, 2014.

Condon, A. and Karp, R. M. Algorithms for graph partitioning on the planted partition model. *Random Struct. Algorithms*, 18(2):116–140, 2001.

Gomes, R. G., Welinder, P., Krause, A., and Perona, P. Crowdclustering. In *Advances in Neural Information Processing Systems 24*, pp. 558–566. 2011.

Heckel, R., Shah, N. B., Ramchandran, K., and Wainwright, M. J. Active ranking from pairwise comparisons and when parametric assumptions do not help. *The Annals of Statistics*, 47(6):3099–3126, 2019.

Holland, P. W., Laskey, K. B., and Leinhardt, S. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109 – 137, 1983. ISSN 0378-8733.

Ibrahim, S. and Fu, X. Mixed membership graph clustering via systematic edge query. *IEEE Transactions on Signal Processing*, 2021.

Jalali, A., Han, Q., Dumitriu, I., and Fazel, M. Relative Density and Exact Recovery in Heterogeneous Stochastic Block Models. *ArXiv e-prints*, December 2015.

Jamieson, K. G., Malloy, M., Nowak, R., and Bubeck, S. lil' ucb : An optimal exploration algorithm for multi-armed bandits. In *COLT*, pp. 423–439, 2014.

Khosla, A., Jayadevaprakash, N., Yao, B., and Fei-Fei, L. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, June 2011.

Mazumdar, A. and Saha, B. A theoretical analysis of first heuristics of crowdsourced entity resolution. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017a.

Mazumdar, A. and Saha, B. Clustering with noisy queries. *Advances in Neural Information Processing Systems*, 30, 2017b.

Mazumdar, A. and Saha, B. Query complexity of clustering with side information. *Advances in Neural Information Processing Systems*, 30, 2017c.

McSherry, F. Spectral partitioning of random graphs. In *FOCS*, pp. 529–537. IEEE Computer Society, 2001. ISBN 0-7695-1390-5.

Meila, M. Comparing clusterings—an information based distance. *J. Multivar. Anal.*, 98(5):873–895, May 2007. ISSN 0047-259X.

Mukherjee, C. S., Peng, P., and Zhang, J. Recovering unbalanced communities in the stochastic block model with application to clustering with a faulty oracle. *arXiv preprint arXiv:2202.08522*, 2022.

Raykar, V. C., Yu, S., Zhao, L. H., Valadez, G. H., Florin, C., Bogoni, L., and Moy, L. Learning from crowds. *J. Mach. Learn. Res.*, 11:1297–1322, August 2010. ISSN 1532-4435.

Rohe, K., Chatterjee, S., and Yu, B. Spectral clustering and the high-dimensional stochastic blockmodel. *Annals of Statistics*, 39(4):1878–1915, August 2011.

Sorokin, A. and Forsyth, D. Utility data annotation with Amazon Mechanical Turk. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pp. 1–8. IEEE, June 2008. ISBN 978-1-4244-2339-2.

Verroios, V. and Garcia-Molina, H. Entity resolution with crowd errors. In *ICDE*, pp. 219–230. IEEE Computer Society, 2015.

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

Vinayak, R. K. and Hassibi, B. Crowdsourced clustering: Querying edges vs triangles. In *Advances in Neural Information Processing Systems*, pp. 1316–1324, 2016.

Vinayak, R. K., Oymak, S., and Hassibi, B. Graph clustering with missing data: Convex algorithms and analysis. In *Neural Information Processing Systems Conference (NIPS)*, 2014.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

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

Welinder, P., Branson, S., Belongie, S., and Perona, P. The multidimensional wisdom of crowds. In *Neural Information Processing Systems Conference (NIPS)*, 2010.

Yun, S.-Y. and Proutiere, A. Community detection via random and adaptive sampling. In *Conference on learning theory*, pp. 138–175. PMLR, 2014.