Correlation Clustering with Same-Cluster Queries Bounded by Optimal Cost

Barna Saha
University of California, Berkeley, USA
http://www.barnasaha.net
barnas@berkeley.edu

Sanjay Subramanian
Allen Institute for Artificial Intelligence, Seattle, WA, USA[1]
http://www.sanjayss34.github.io
sanjays@allenai.org

Abstract

Several clustering frameworks with interactive (semi-supervised) queries have been studied in the past. Recently, clustering with same-cluster queries has become popular. An algorithm in this setting has access to an oracle with full knowledge of an optimal clustering, and the algorithm can ask the oracle queries of the form, “Does the optimal clustering put vertices u and v in the same cluster?” Due to its simplicity, this querying model can easily be implemented in real crowd-sourcing platforms and has attracted a lot of recent work.

In this paper, we study the popular correlation clustering problem (Bansal et al., 2002) under the same-cluster querying framework. Given a complete graph $G = (V, E)$ with positive and negative edge labels, correlation clustering objective aims to compute a graph clustering that minimizes the total number of disagreements, that is the negative intra-cluster edges and positive inter-cluster edges. In a recent work, Ailon et al. (2018b) provided an approximation algorithm for correlation clustering that approximates the correlation clustering objective within $(1 + \epsilon)$ with $O(\frac{k^{14} \log n \log k}{\epsilon^6})$ queries when the number of clusters, k, is fixed. For many applications, k is not fixed and can grow with $|V|$. Moreover, the dependency of $k^1 4$ on query complexity renders the algorithm impractical even for datasets with small values of k.

In this paper, we take a different approach. Let C_{OPT} be the number of disagreements made by the optimal clustering. We present algorithms for correlation clustering whose error and query bounds are parameterized by C_{OPT} rather than by the number of clusters. Indeed, a good clustering must have small C_{OPT}. Specifically, we present an efficient algorithm that recovers an exact optimal clustering using at most $2C_{OPT}$ queries and an efficient algorithm that outputs a 2-approximation using at most C_{OPT} queries. In addition, we show under a plausible complexity assumption, there does not exist any polynomial time algorithm that has an approximation ratio better than $1 + \alpha$ for an absolute constant $\alpha > 0$ with $o(C_{OPT})$ queries. Therefore, our first algorithm achieves the optimal query bound within a factor of 2.

We extensively evaluate our methods on several synthetic and real-world datasets using real crowd-sourced oracles. Moreover, we compare our approach against known correlation clustering algorithms that do not perform querying. In all cases, our algorithms exhibit superior performance.

2012 ACM Subject Classification Theory of computation → Unsupervised learning and clustering; Theory of computation → Approximation algorithms analysis

Keywords and phrases Clustering, Approximation Algorithm, Crowdsourcing, Randomized Algorithm

Digital Object Identifier 10.4230/LIPIcs.ESA.2019.81

Related Version A full version of the paper is available at https://arxiv.org/abs/1908.04976.

Supplement Material Code and Data:
https://www.github.com/sanjayss34/corr-clust-query-esa2019

[1] Most of this work was completed when the second author was at the University of Pennsylvania and the University of Massachusetts at Amherst.

27th Annual European Symposium on Algorithms (ESA 2019).
Editors: Michael A. Bender, Ola Svensson, and Grzegorz Herman; Article No. 81; pp. 81:1–81:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Funding *Barna Saha*: B. Saha is partially supported by an NSF CAREER Award CCF 1652303, a Google Faculty Award and an Alfred P. Sloan fellowship.
Sanjay Subramanian: This work was supported in part by the National Science Foundation (NSF) Research Experiences for Undergraduates (REU) program.

Acknowledgements The second author would like to thank Dan Roth for letting him use his machines for running experiments, Sainyam Galhotra for help with datasets, and Rajiv Gandhi for useful discussions.

1 Introduction

In correlation clustering, the algorithm is given potentially inconsistent information about similarities and dissimilarities between pairs of vertices in a graph, and the task is to cluster the vertices so as to minimize disagreements with the given information [7, 10]. The correlation clustering problem was first proposed by Bansal, Blum and Chawla [7] and since then it has found numerous applications in document clustering, image segmentation, grouping gene expressions etc. [7, 10].

In correlation clustering, we are given a complete graph $G = (V, E)$, $|V| = n$, where each edge is labelled either $+$ or $-$. An optimal clustering partitions the vertices such that the number of intra-cluster negative edges and inter-cluster positive edges is minimized. The problem is known to be NP-Hard. The seminal work of Bansal et al. [7] gave a constant factor approximation for correlation clustering. Following a long series of works [7, 9, 4, 16, 12], the best known approximation bounds till date are a 3-approximation combinatorial algorithm [1] and a 2.06-approximation based on linear programming rounding [10]. The proposed linear programming relaxation for correlation clustering [9, 1, 10] is known to have an integrality gap of 2, but there does not exist yet a matching algorithm that has an approximation ratio 2 or lower.

Correlation clustering problem can be extended to weighted graphs for an $O(\log n)$-approximation bound and is known to be optimal [12]. Moreover, when one is interested in maximizing agreements, a polynomial time approximation scheme was provided by Bansal et al. [7].

Over the last two decades, crowdsourcing has become a widely used way to generate labeled data for supervised learning. The same platforms that are used for this purpose can also be used for unsupervised problems, thus converting the problems to a semi-supervised active learning setting. This can often lead to significant improvements in accuracy. However, using crowdsourcing introduces another dimension to the optimization problems, namely minimizing the amount of crowdsourcing that is used. The setting of active querying has been studied previously in the context of various clustering problems. Balcan and Blum [6] study a clustering problem in which the only information given to the algorithm is provided through an oracle that tells the algorithm either to “merge” two clusters or to “split” a cluster. More recently, Ashtiani, Kushgra and Ben-David [5] considered a framework of same-cluster queries for clustering; in this framework, the algorithm can access an oracle that has full knowledge of an optimal clustering and can issue queries to the oracle of the form “Does the optimal clustering put vertices u and v in the same cluster?” Because of its simplicity, such queries are highly suited for crowdsourcing and has been studied extensively both in theory community [3, 19, 2, 15] and in applied domains [23, 14, 17, 22]. Correlation clustering has also been considered in this context. Ailon, Bhattacharya and Jaiswal [2] study correlation clustering in this framework under the assumption that the number k of clusters is fixed. They gave an $(1 + \epsilon)$ approximation algorithm for correlation clustering that runs in polynomial time and issues $O(k^{14} \log n \log k/\epsilon^6)$ queries. However, for most relevant applications, the number of clusters k is not fixed. Even for fixed k, the dependence of k^{14} is huge (consider $k = 2$ and $2^{14} = 16384$ with additional constants terms hidden under $O()$ notation).

In this paper, we give near-optimal algorithms for correlation clustering with same-cluster queries that are highly suitable for practical implementation and whose performance is parameterized by the optimum number of disagreements. Along with providing theoretical guarantees, we perform extensive experiments on multiple synthetic and real datasets. Let C_{OPT} be the number of disagreements made by the optimal clustering. Our contributions are as follows.

1. A deterministic algorithm that outputs an *optimal* clustering using at most $2C_{OPT}$ queries (Section 3).
2. An expected 2-approximation algorithm that uses at most C_{OPT} queries in expectation (Section 4).
3. A new lower bound that shows it is not possible to get an $(1 + \alpha)$ approximation for some constant $\alpha > 0$ with any polynomial time algorithm that issues $o(C_{OPT})$ queries assuming GAP-ETH (see definition in Section 5).
4. An extensive experimental comparison that not only compares the effectiveness of our algorithms, but also compares the state-of-the art correlation clustering algorithms that do not require any querying (Section 6).

Assumption of an optimum oracle [5, 2] is quite strong in practice. However, our experiments reveal that such an assumption is not required. In correlation clustering, often the ± edges are generated by fitting an automated classifier, where each vertex corresponds to some object and is associated with a feature vector. In our experiments with real-world data, instead of an optimum oracle, we use crowdsourcing. By making only a few pair-wise queries to a crowd oracle, we show it is possible to obtain an optimum or close to optimum clustering. After our work, it came to our notice that it may be possible to use Bocker et al.'s [8] results on fixed-parameter tracktability of cluster editing to adapt to our setting, and get better constants on the query complexity. This is an ongoing work. Our algorithms and techniques are vastly different from [8] and are also considerably simpler.

2 Related Work

Asthiani et al. [5] considered the k-means objective with same-cluster queries and showed that it is possible to recover the optimal clustering under k-means objective with high probability by issuing $O(k^2 \log k + k \log n)$ queries if a certain margin condition holds for each cluster. Gamlath, Huang and Svensson extended the above result when approximation is allowed [15]. Ailon et al. [2] studied correlation clustering with same-cluster queries and showed that there exists an $(1 + \epsilon)$ approximation for correlation clustering where the number of queries is a (large) polynomial in k. Our algorithms are different from those in [2] in that our guarantees are parameterized by C_{OPT} rather than by k. Kushagra et al. [18] study a restricted version of correlation clustering where the valid clusterings are provided by a set of hierarchical trees and provide an algorithm using same-cluster queries for a related setting, giving guarantees in terms of the size of the input instance (or the VC dimension of the input instance) rather than C_{OPT}. [19] studied, among other clustering problems, a random instance of correlation clustering under same-cluster queries. Our algorithms are based on the basic 3-approximation algorithm of Ailon et al. [1] that selects a pivot vertex randomly and forms a cluster from that vertex and all of its +-neighbors. They further honed this approach by choosing to keep each vertex in the pivot's cluster with a probability that is a function of the linear programming solution. Chawla et al. [10] used a more sophisticated function of the linear programming solution to design the current state-of-the-art algorithm, which gives a 2.06 approximation for correlation clustering.

3 Finding an Optimal Clustering

We are given a query access to an oracle that given any two vertices u and v returns whether or not u and v are together in a cluster in an optimal solution. Let OPT denote the optimal solution which is used by the oracle. Given a positive $(+)$ edge (u, v), if OPT puts u and v in different clusters, then we say OPT makes a mistake on that edge. Similarly for a negative $(-)$ edge (u, v), if OPT puts them together in a cluster then again OPT makes a mistake on it. Similarly, our algorithm can decide to make mistakes on certain edges and our goal is to minimize the overall number of mistakes. It is easy to see that an optimal solution for a given input graph makes mistakes only on edges that are part of a $(+, +, -)$ triangle. Moreover, any optimal solution must make at least one mistake in such a triangle.

The pseudocode for our algorithm, QueryPivot, is given in Algorithm 1. The algorithm is as follows (in the following description, we give in brackets the corresponding line number for each step). We pick a pivot u arbitrarily from the set of vertices that are not clustered yet [line 5]. For each $(+, +, -)$ triangle (u, v, w) [line 10], if we have not yet determined via queries that OPT makes a mistake on $\{u, v\}$ or that OPT makes a mistake on $\{u, w\}$ [lines 11-14], then (1) we query $\{u, v\}$ [line 17] and if OPT makes a mistake on this edge, we too decide to make a mistake on this edge and proceed to the next $(+, +, -)$ triangle involving u and (2) if OPT does not make a mistake on $\{u, v\}$, then we query $\{u, w\}$ [line 23] and make a mistake on it if OPT makes a mistake on it. Note that if we have already queries one of $\{u, v\}$ or $\{u, w\}$ and found a mistake, we do not query the other edge [line 11]. Once we have gone through all $(+, +, -)$ triangles involving u then for every $v \neq u$, if we have not already decided to make a mistake on $\{u, v\}$, then if $\{u, v\}$ is a $+$ edge we keep v in u's cluster and if $\{u, v\}$ is a $-$ edge we do not put v in u's cluster. On the other hand, if we have decided to make a mistake on $\{u, v\}$, then if $\{u, v\}$ is a $-$ edge we keep v in u's cluster and if $\{u, v\}$ is a $+$ edge we do not put v in u's cluster. Finally, we remove all vertices in u's cluster from the set of remaining vertices and recursively call the function on the set of remaining vertices.

In the pseudocode, $Queried[v] = 1$ means the algorithm has already issued a query $(pivot, v)$ to the oracle, $Mistake[v] = 1$ means it has decided to make a mistake on the edge $(pivot, v)$ based on the oracle answer, and $Oracle(pivot, v)$ returns 1 iff OPT makes a mistake on the edge $\{pivot, v\}$. We prove the following theorem that shows that QueryPivot is able to recover the optimal clustering known to the oracle with a number of queries bounded in terms of C_{OPT}.

▶ **Theorem 3.1.** *Let C_{OPT} be the number of mistakes made by an optimal clustering. The* QueryPivot *algorithm makes C_{OPT} mistakes and makes at most $2C_{OPT}$ queries to the oracle.*

Algorithm 1 QueryPivot.

```
Input: vertex set $V$, adjacency matrix $A$, oracle $Oracle$
if $V == \emptyset$ then
  return $\emptyset$
end if
$pivot \leftarrow$ Arbitrary vertex in $V$
$T \leftarrow$ all $(+,+,-)$ triangles that include $pivot$
$C \leftarrow V$
$Queried \leftarrow$ length-n array of zeros
$Mistakes \leftarrow$ length-n array of zeros
for $(pivot, v, w) \in T$ do
  if $Mistake[v] == 1$ or $Mistake[w] == 1$ then
    continue
  else if $Queried[v] == 1$ and $Queried[w] == 1$ then
    continue
  else if $Queried[v] == 0$ then
    $Queried[v] \leftarrow 1$
    if $Oracle(pivot, v) == 1$ then
      $Mistake[v] \leftarrow 1$
    end if
  end if
  if $Queried[w] == 0$ and $Mistake[v] == 0$ then
    $Queried[w] \leftarrow 1$
    if $Oracle(pivot, w) == 1$ then
      $Mistake[w] \leftarrow 1$
    end if
  end if
end for
for $v \in V \setminus \{pivot\}$ do
  if $(v \in N^-(pivot)$ and $Mistake[v] == 0)$ or $(v \in N^+(pivot)$ and $Mistake[v] == 1)$ then
    $C = C \setminus \{v\}$
  end if
end for
return $\{C\} \cup$ QueryPivot$(V \setminus C, A, Oracle)$
```

For a given cluster C and a vertex $w \in C$, we denote by $N_C^+(w)$ the set of vertices in C that have $+$ edges with w. Similarly, we denote by $N_C^-(w)$ the set of vertices in C that have $-$ edges.

The algorithm time complexity is dominated by the time taken to check $(+,+,-)$ triangles involved with the pivots. Let E^+ denote the set of positive edges in G. Then all the $(+,+,-)$ triangles that include a pivot can be checked in time $O(|E^+| * n)$.

▶ **Lemma 3.2.** *The* QueryPivot *algorithm outputs a valid partition of the vertices.*

Proof. Note that the pivot is never removed from C. Hence, between each pair of consecutive recursive calls, at least one vertex is removed from V. The algorithm must then terminate after at most n recursive calls. Moreover, in each recursive call, the set of vertices passed to the next recursive call is disjoint from the cluster created in that recursive call. Thus, inductively, the sets returned by the algorithm must be disjoint. ◀

▶ **Lemma 3.3.** *Consider a clustering $\mathcal{C}$ in which some cluster C contains vertices u, v s.t. $\{u, v\}$ is a $-$ edge and s.t. u and v do not form a $(+,+,-)$ triangle with any other vertex in C. $\mathcal{C}$ is suboptimal.*

Proof. If we were to remove v from C and put it in a singleton cluster, we would make $|N^-_C(v)| - |N^+_C(v)|$ fewer mistakes than $\mathcal{C}$. If $|N^-_C(v)| - |N^+_C(v)| > 0$, then $\mathcal{C}$ is suboptimal. Therefore, assume $|N^-_C(v)| \leq |N^+_C(v)|$. Now, note that $\forall w \in N^+_C(v)$, $w \in N^-_C(u)$ because otherwise, u, v, and w form a $(+,+,-)$ triangle. Thus $|N^+_C(v)| \leq |N^-_C(u)| - 1$ because $v \in N^-_C(u)$. Moreover, $|N^+_C(u)| \leq |N^-_C(v)| - 1$ because $u \in N^-_C(v)$.

Hence, if we were to remove u from C and put it in a singleton cluster, we would make $|N^-_C(u)| - |N^+_C(u)| \geq |N^+_C(v)| - |N^-_C(v)| + 2$ fewer mistakes than $\mathcal{C}$. Since $|N^-_C(v)| - |N^+_C(v)| \leq 0$, $|N^+_C(v)| - |N^-_C(v)| + 2 > 0$, so $\mathcal{C}$ is suboptimal. ◀

▶ **Lemma 3.4.** *Consider a clustering $\mathcal{C}$ in which a cluster C_1 contains a vertex u, a different cluster C_2 contains a vertex v, $\{u,v\}$ is a $+$ edge, and in every $(+,+,-)$ triangle that includes $\{u,v\}$, the clustering makes at least 2 edge mistakes. $\mathcal{C}$ is suboptimal.*

Proof. If we were to remove u from C_1 and put it in C_2, we would make $|N^+_{C_2}(u)| + |N^-_{C_1}(u)| - |N^-_{C_2}(u)| - |N^+_{C_1}(u)| = 2|N^+_{C_2}(u)| + |C_1| - |C_2| - 2|N^+_{C_1}(u)|$ fewer mistakes than $\mathcal{C}$. If $2|N^+_{C_2}(u)| + |C_1| - |C_2| - 2|N^+_{C_1}(u)| > 0$, then $\mathcal{C}$ is suboptimal. Otherwise, note that $\forall w \in N^+_{C_1}(u)$, $w \in N^+_{C_1}(v)$ because if not, u, v, w would form a $(+,+,-)$ triangle in which the algorithm makes fewer than 2 edge mistakes. By a similar argument, $\forall w \in N^+_{C_2}(v)$, $w \in N^+_{C_2}(u)$. Thus, since in addition, $\{u,v\}$ is a $+$ edge, we have that $|N^+_{C_1}(v)| \geq |N^+_{C_1}(u)| + 1$ and $|N^+_{C_2}(u)| \geq |N^+_{C_2}(v)| + 1$. Now if we were to remove v from C_2 and put it in C_1, the number of mistakes will reduce by $|N^+_{C_1}(v)| + |N^-_{C_2}(v)| - |N^-_{C_1}(v)| - |N^+_{C_2}(v)| = 2|N^+_{C_1}(v)| + |C_2| - |C_1| - 2|N^+_{C_2}(v)|$. Since $|N^+_{C_1}(v)| \geq |N^+_{C_1}(u)| + 1$ and $|N^+_{C_2}(v)| \leq |N^+_{C_2}(u)| - 1$, we have that $2|N^+_{C_1}(v)| + |C_2| - |C_1| - 2|N^+_{C_2}(v)| \geq 2(|N^+_{C_1}(u)| + 1) + |C_2| - |C_1| - 2(|N^+_{C_2}(u) - 1)$. Since $2|N^+_{C_2}(u)| + |C_1| - |C_2| - 2|N^+_{C_1}(u)| \leq 0$, $2(|N^+_{C_1}(u)| + 1) + |C_2| - |C_1| - 2(|N^+_{C_2}(u) - 1) > 0$, so $\mathcal{C}$ is suboptimal. ◀

▶ **Lemma 3.5.** *When given an oracle corresponding to an optimal clustering OPT, the clustering returned by the* QueryPivot *algorithm is identical to OPT. It follows that the algorithm's clustering makes at most as many mistakes as OPT.*

Proof. We will prove inductively that in each recursive call, the cluster C returned by the algorithm is a cluster in OPT. Note that at the beginning of the first recursive call, the claim that all clusters formed so far are clusters in OPT is vacuously true because there are no clusters yet formed. Now consider an arbitrary but particular recursive call, and let u be the pivot in this recursive call. Suppose for contradiction that C is not a cluster in OPT.
Case 1: There is a vertex v such that $v \notin C$, but in OPT, v is in the same cluster as u. Let H be the cluster in OPT that contains u and v. First, observe that H must be a subset of the remaining vertices in this recursive call; otherwise, one of the clusters formed in a previous call contains some vertex in H but does not include u, contradicting the induction hypothesis because this previously formed cluster is not a cluster in OPT. Next, note that for any mistake that the algorithm makes on an edge incident on a pivot, the algorithm queries the OPT oracle and makes the mistake iff OPT makes the mistake. Then if $\{u,v\}$ is a $+$ edge, then the algorithm must have queried the oracle for $\{u,v\}$ and found that OPT makes a mistake on it because the algorithm decided to make a mistake on that edge. This implies that OPT puts u and v in different clusters, which is a contradiction. Now suppose instead that $\{u,v\}$ is a $-$ edge. Again if the algorithm queried the oracle for $\{u,v\}$, then OPT must have put u and v in different clusters, so it must be the case that the algorithm did not query the oracle for $\{u,v\}$. It follows that for any $(+,+,-)$ triangle (u,v,w) that includes $\{u,v\}$, our algorithm has queried $\{u,w\}$ and found OPT makes a mistake on the $+$ edge $\{u,w\}$. Then for any such triangle, $w \notin H$. It follows that u and v do not form a

$(+,+,-)$ triangle with any vertex in H. Since u and v are in the same cluster H in OPT, $\{u,v\}$ is a $-$ edge, and u and v do not form a $(+,+,-)$ triangle with any other vertex in H, the conditions for Lemma 3.3 are satisfied. Therefore, OPT is a suboptimal clustering, which is a contradiction.
Case 2: There is a vertex v such that $v \in C$, but in OPT, v is not in u's cluster. As in the first case, if $\{u,v\}$ were a $-$ edge, the algorithm must make a mistake on $\{u,v\}$ and so must have queried OPT and found that OPT made a mistake on $\{u,v\}$, a contradiction. Now suppose instead that $\{u,v\}$ is a $+$ edge. If there is some vertex w that was clustered prior to this recursive call s.t. $\{u,v,w\}$ is a $(+,+,-)$ triangle in which OPT makes exactly one mistake (on $\{u,v\}$), then note that either u or v should be in the same cluster as w because one of $\{u,w\}$ and $\{v,w\}$ must be a $+$ edge; in this case, we have reached a contradiction with the inductive hypothesis because the previously formed cluster that included w did not include u or v. Then in order to show that the conditions for Lemma 3.4 are satisfied, we must show that for every vertex w in the set of remaining vertices when u is the pivot, if (u,v,w) is a $(+,+,-)$ triangle, then OPT must make at least two mistakes in the triangle. Since OPT makes a mistake on $\{u,v\}$ but the algorithm does not do so, it must be the case that the algorithm did not query $\{u,v\}$. Since the algorithm did not query $\{u,v\}$, for every $(+,+,-)$ triangle (u,v,w) that includes $\{u,v\}$ and such that w is in the set of remaining vertices when u is the pivot, OPT must make a mistake on $\{u,w\}$. Then since OPT makes a mistake on $\{u,v\}$ and on $\{u,w\}$ in any $(+,+,-)$ triangle (u,v,w), we have by Lemma 3.4 that OPT is suboptimal clustering, which is a contradiction. ◀

▶ **Lemma 3.6.** *Let C_{OPT} be the number of mistakes made by an optimal clustering OPT. Then the* QueryPivot *algorithm makes at most $2C_{OPT}$ queries to the oracle.*

Proof. The algorithm queries the oracle only when considering $(+,+,-)$ triangles. Note that whenever considering a particular $(+,+,-)$ triangle, if the algorithm makes a query, it makes at most two queries when considering that triangle and makes at least one mistake that had not been made when considering previous triangles. Therefore, the algorithm makes at most twice as many queries as mistakes. Since the algorithm makes exactly C_{OPT} mistakes, the algorithm makes at most $2C_{OPT}$ queries. ◀

Theorem 3.1 follows directly from Lemmas 3.5 and 3.6.

4 A 2-Approximation Algorithm for Correlation Clustering

A natural question that arises from QueryPivot is how to use fewer queries and obtain an approximation guarantee that is better than the state-of-the-art outside the setting with same-cluster queries, which is a 2.06-approximation. In this section, we show that a randomized version of QueryPivot gives a 2-approximation in expectation using at most C_{OPT} queries in expectation.

The algorithm RandomQueryPivot(p) is as follows. We pick a pivot u uniformly at random from the vertices yet to be clustered. For each $(+,+,-)$ triangle (u,v,w), we have two cases. (1) If $\{u,v\}$ and $\{u,w\}$ are both $+$ edges, then with probability p (chosen appropriately), we query both $\{u,v\}$ and $\{u,w\}$ and for each of these two edges we make a mistake on the edge iff OPT makes a mistake on the edge. With probability $1-p$ we make no queries for this triangle and proceed to the next triangle. (2) If one of $\{u,v\}$ and $\{u,w\}$ is a $+$ edge and the other is a $-$ edge, then with probability p, we do the following. First, we query the $+$ edge and if OPT makes a mistake on it, then we make a mistake on it and proceed to the next triangle. If OPT does not make a mistake on the $+$ edge, then we query

Algorithm 2 RandomQueryPivot.

```
Input: vertex set $V$, adjacency matrix $A$, oracle $Oracle$, parameter $p$
if $V == \emptyset$ then
    return $\emptyset$
end if
$pivot \leftarrow$ Random vertex in $V$
$T \leftarrow$ all $(+, +, -)$ triangles that include $pivot$
$C \leftarrow V$
$Queried \leftarrow$ length-n array of zeros
$Mistakes \leftarrow$ length-n array of zeros
for $(pivot, v, w) \in T$ do
    // Without loss of generality, suppose that $\{pivot, v\}$ is a $+$ edge
    Sample $r$ from $Uniform(0, 1)$
    if $r > p$ then
        continue
    end if
    if $Oracle(pivot, v) == 1$ then
        $Mistake[v] \leftarrow 1$
    end if
    if $Mistake[v] == 0$ or $\{pivot, w\}$ is a $+$ edge then
        if $Oracle(pivot, w) == 1$ then
            $Mistake[v] \leftarrow 1$
        end if
    end if
end for
for $v \in V \setminus \{pivot\}$ do
    if $(v \in N^-(pivot)$ and $Mistake[v] == 0)$ or $(v \in N^+(pivot)$ and $Mistake[v] == 1)$ then
        $C = C \setminus \{v\}$
    end if
end for
return $\{C\} \cup$ RandomQueryPivot$(V \setminus C, A, Oracle)$
```

the $-$ edge and make a mistake on the $-$ edge iff OPT does so. Again, with probability $1 - p$ we make no queries for this triangle and proceed to the next triangle. Once we have gone through all triangles, if we have not already decided to make a mistake on $\{u, v\}$, then if $\{u, v\}$ is a $+$ edge we keep v in u's cluster and if $\{u, v\}$ is a $-$ edge we do not put v in u's cluster. On the other hand, if we have decided to make a mistake on $\{u, v\}$, then if $\{u, v\}$ is a $-$ edge we keep v in u's cluster and if $\{u, v\}$ is a $+$ edge we do not put v in u's cluster. Finally, we remove all vertices in u's cluster from the set of remaining vertices and recursively call the function on the set of remaining vertices. Note that given a pivot u and a $(+, +, -)$ triangle containing u, if the algorithm chooses not to query either of the edges incident on u, then the algorithm must make a mistake on the edge opposite to u in that triangle.

▶ **Theorem 4.1.** RandomQueryPivot(p) *gives a* $\max\left(2, \frac{3}{1+2p}\right)$*-approximation in expectation and uses at most* $\max(4p, 1) * C_{OPT}$ *queries in expectation.*

▶ **Corollary 4.2.** *When* $p = 0.25$*,* RandomQueryPivot *gives a* 2*-approximation in expectation and uses at most* C_{OPT} *queries in expectation.*

▶ **Lemma 4.3.** *In an arbitrary but particular recursive call, the probability that* RandomQueryPivot *queries edge* $\{u, v\}$ *on which* OPT *makes a mistake given that* u *is the pivot is equal to the probability that* RandomQueryPivot *queries edge* $\{u, v\}$ *given that* v *is the pivot.*

Proof. For a $+$ edge $\{u,v\}$ on which OPT makes a mistake, the probability that the edge is queried given that one of the vertices is the pivot is a function only of the number of $(+,+,-)$ triangles that include the edge. In particular, if T is the number of $(+,+,-)$ triangles including the edge, the probability that the edge is queried is $1-(1-p)^T$. This number of triangles does not depend on the pivot vertex, so the claim holds if $\{u,v\}$ is a $+$ edge. If $\{u,v\}$ is a $-$ edge, then we claim that the probability that $\{u,v\}$ is queried given that either u or v is a function only of the number of $(+,+,-)$ triangles that include $\{u,v\}$ in which OPT makes a mistake only on this $-$ edge. This claim is true because (1) in any $(+,+,-)$ triangle in which OPT makes a mistake on the $-$ and a $+$ edge, OPT must make a mistake on all of the three edges in the triangle and (2) when considering a $(+,+,-)$ triangle such that the pivot is an endpoint of the $-$ edge, the algorithm queries the $-$ edge iff OPT does not make a mistake on the $+$ edge of which the pivot is an endpoint. It follows that for any $(+,+,-)$ triangle in which the algorithm queries the $-$ edge, OPT must make a mistake only on the $-$ edge. Since the number of $(+,+,-)$ triangles that include $\{u,v\}$ in which OPT makes a mistake only on $\{u,v\}$ does not depend on whether u or v is the pivot, the claim holds when $\{u,v\}$ is a $-$ edge. ◀

Let $s_{uv}=1$ if $\{u,v\}$ is a $-$ edge and 0 otherwise. Let c^*_{uv} equal 1 if OPT makes a mistake on $\{u,v\}$ and 0 otherwise.

Let OPT^t be the number of edges $\{u,v\}$ s.t. $c^*_{uv}=1$ and the algorithm makes a decision on $\{u,v\}$ in iteration t. Let ALG^t be the number of edges $\{u,v\}$ s.t. the algorithm makes a mistake on $\{u,v\}$ and the algorithm makes a decision on $\{u,v\}$ in iteration t.

Let V_t be the set of vertices remaining at the beginning of iteration t. Let D^t_{uv} be the event that the algorithm makes a decision on $\{u,v\}$ in iteration t.

▶ **Lemma 4.4.** *Let T be the number of iterations that the algorithm takes to cluster all vertices. If $E[ALG^t|V_t]\leq \alpha E[OPT^t|V_t]$, for each iteration t, then $E\left[\sum_{t=1}^T ALG^t\right]\leq \alpha E\left[\sum_{t=1}^T OPT^t\right]$.*

Proof. Define $X_0=0$ and for each $s>0$, define $X_s=\sum_{t=1}^s \alpha OPT^t - ALG^t$. If the condition in the lemma holds, then X_s is a submartingale because $E[X_{s+1}|X_s]\geq X_s$. Also, T is a stopping time that is almost surely bounded (since $T\leq n$ with probability 1). By Doob's optional stopping theorem [24, p. 100], if T is a stopping time that is almost surely bounded and X is a discrete-time submartingale, then $E[X_T]\geq E[X_0]$. Then we have that $E[X_T]=E\left[\sum_{t=1}^T \alpha OPT^t - ALG^t\right]\geq E[X_0]=0$. ◀

▶ **Lemma 4.5.** *The expected number of mistakes made by the algorithm's clustering is at most* $\max\left(2,\frac{3}{1+2p}\right)C_{OPT}$.

Proof Sketch. Here we give a sketch of the proof. By Lemma 4.4, if we show that $E[ALG^t-\alpha OPT^t]\leq 0$ for any t, where $\alpha\leq\max\left(2,\frac{3}{1+2p}\right)$, then the claim will follow. Let A^t_w be the event that $w\in V_t$ is the pivot in iteration t.

$$E[OPT^t|V_t]=\sum_{\{u,v\}\subseteq E\cap(V_t\times V_t)}\frac{c^*_{uv}}{|V_t|}\sum_{w\in V_t}\Pr[D^t_{uv}|A^t_w]$$

Now we will write $E[ALG^t|V_t]$ by charging the algorithm's mistakes to each of OPT's mistakes.

Let M_{uv}^t be the charge incurred to $\{u,v\}$ in iteration t. We will assign charges such that $M_{uv}^t = 0$ if $c_{uv}^* = 0$. Then

$$E[ALG^t|V_t] = \sum_{\{u,v\}\subseteq E\cap(V_t\times V_t)} c_{uv}^* E[M_{uv}^t] = \sum_{\{u,v\}\subseteq E\cap(V_t\times V_t)} c_{uv}^* \frac{1}{|V_t|} \sum_{w\in V_t} E[M_{uv}^t|A_w^t]$$

Our goal is to compute an upper bound on $E[M_{uv}^t|A_w^t]$. To do so, we define several events. For each edge $\{u,v\}$ s.t. $c_{uv}^* = 1$, define the following subsets of V_t: $\{u,v\}$, $\forall i \in \{1,2,3\}$, T_i^{uv} is the set of vertices w s.t. $\{u,v,w\}$ is a $(+,+,-)$ triangle in which OPT makes exactly i mistakes, S^{uv} is the set of vertices w s.t. $\{u,v,w\}$ is a $(+,-,-)$ or $(+,+,+)$ triangle in which OPT makes exactly 2 mistakes, $R^{uv} \equiv V_t \setminus T_1^{uv} \setminus T_2^{uv} \setminus S^{uv} \setminus \{u,v\}$. Furthermore, let T_{2u}^{uv} be the subset of T_2^{uv} s.t. $w \in T_{2u}^{uv}$ if the 2 mistakes in $\{u,v,w\}$ are both incident on u. Similarly, let S_u^{uv} be the subset of S^{uv} s.t. $w \in T_u^{uv}$ if the 2 mistakes in $\{u,v,w\}$ are both incident on u. $\forall w \in T_1^{uv}$, the probability that the algorithm makes a mistake on $\{u,v\}$ given that w is the pivot is $\Pr[D_{uv}^t|A_w^t] = 1$. Note that T_1^{uv}, T_2^{uv}, $\{u,v\}$, S^{uv}, and R^{uv} partition V_t.

We compute $E[M_{uv}^t|A_w^t]$ (or an upper bound thereof) when w is in each of the sets $\{u,v\}$, T_1^{uv}, T_{2u}^{uv}, T_{2v}^{uv}, S_u^{uv}, S_v^{uv}, and R^{uv}, which partition V_t. Similarly, we analyze $\Pr[D_{uv}^t|A_w^t]$, breaking up the calculation based on whether the pivot w is in T_1^{uv}, T_{2u}^{uv}, T_{2v}^{uv}, S_{2u}^{uv}, S_{2v}^{uv}, $\{u,v\}$, or R^{uv}.

In order to prove the claim, we show that

$$\sum_{w\in V_t} E[M_{uv}^t|A_w^t] \le \max\left(2, \frac{3}{1+2p}\right) \sum_{w\in V_t} \Pr[D_{uv}^t|A_w^t]$$

Thus, we have shown that $E[ALG^t|V_t] \le \max\left(2, \frac{3}{1+2p}\right) E[OPT^t|V_t]$. By Lemma 4.4, the claim follows. ◀

▶ **Lemma 4.6.** *The expected number of queries made by* RandomQueryPivot *is at most* $\max(4p, 1)\, C_{OPT}$.

Proof. We follow an approach similar to that taken in the proof of Lemma 4.5. We will bound the number of queries made by the algorithm in each iteration t by charging queries to edges on which OPT makes a mistake and on which the algorithm makes a mistake in iteration t. Let U^t be the number of queries made by the algorithm in iteration t. We charge queries as follows to an edge $\{u,v\}$ on which OPT makes a mistake:

1. When u or v is the pivot, the algorithm makes at most 1 query on $\{u,v\}$ itself.
2. When u or v is the pivot (suppose WLOG u is the pivot), $\forall w \in T_1^{uv}$ (defined in the proof of Lemma 4.5), the algorithm makes a query on $\{u,w\}$ with probability p if $\{u,w\}$ is a $+$ edge.
3. When the pivot w is in T_1^{uv}, then with probability p at most 2 queries are made when the algorithm considers the triangle $\{u,v,w\}$.
4. Note that we need not worry about charging mistakes in $(+,+,-)$ triangles in which OPT makes 2 mistakes because when considering such a triangle the algorithm is guaranteed not to query the $-$ edge on which OPT does not make a mistake. We also need not worry about charging mistakes in $(+,+,-)$ triangles in which OPT makes 3 mistakes because each edge can be charged for any query made on that edge.

$$
\begin{aligned}
E[U^t|V_t] &\le \sum_{\{u,v\}\subseteq E\cap(V_t\times V_t)} \frac{c^*_{uv}}{|V_t|}\left[2(1+p|T_1^{uv}|)+\sum_{w\in T_1^{uv}} 2p\right]\\
&\le \sum_{\{u,v\}\subseteq E\cap(V_t\times V_t)} \frac{c^*_{uv}}{|V_t|}\left[2+4p|T_1^{uv}|\right]
\end{aligned}
$$

Recall from the proof of Lemma 4.5 that

$$
E[OPT^t|V_t] = \sum_{\{u,v\}\subseteq E\cap(V_t\times V_t)} \frac{c^*_{uv}}{|V_t|}\sum_{w\in V_t}\Pr[D^t_{uv}|A^t_w] \ge \sum_{\{u,v\}\subseteq E\cap(V_t\times V_t)} \frac{c^*_{uv}}{|V_t|}\left(2+|T_1^{uv}|\right)
$$

Here the second inequality follows from computing $\sum_{w\in V_t}\Pr[D^t_{uv}|A^t_w]$ (see Case 1 and 7). Clearly, $\frac{2+4p|T_1^{uv}|}{2+|T_1^{uv}|} \le \max(4p,1)$, so $E[U^t|V_t] \le \max(4p,1)\,E[OPT^t|V_t]$. Then by Lemma 4.4, the claim follows. ◀

Theorem 4.1 follows directly from Lemmas 4.5 and 4.6.

5 Lower bound on Query Complexity

The query complexities of the algorithms presented in this paper are linear in C_{OPT}, but it is not clear whether this number of queries is necessary for finding an (approximately) optimal solution. In this section, we show that a query complexity linear in C_{OPT} is necessary for approximation factors below a certain threshold assuming that the Gap-ETH, stated below, is true.

▶ **Hypothesis 5.1** (Gap-ETH). *There is some absolute constant $\gamma > 0$ s.t. any algorithm that can distinguish between the following two cases for any given 3-SAT instance with n variables and m clauses must take time at least $2^{\Omega(m)}$. (see e.g. [13])*

i. *The instance is satisfiable.*

ii. *Fewer than $(1-\gamma)m$ of the clauses are satisfiable.*

The proof of the following lemma is provided in the appendix, which appears in the extended version of the paper.

▶ **Lemma 5.2.** *Let C_{OPT} be the optimum number of mistakes for a given instance of correlation clustering. Assuming Hypothesis 1, there is no $\left(1+\frac{\gamma}{10}\right)$-approximation algorithm for correlation clustering on N vertices that runs in time $2^{o(C_{OPT})}poly(N)$ where γ is as defined in Hypothesis 1.*

As a corollary to the above lemma, we obtain the following.

▶ **Theorem 5.3.** *There is no polynomial-time $\left(1+\frac{\gamma}{10}\right)$-approximation algorithm for correlation clustering that uses $o(C_{OPT})$ queries.*

Proof. Suppose there exists an algorithm that approximates correlation clustering with an approximation factor of $\left(1+\frac{\gamma}{10}\right)$ and uses at most $o(C_{OPT})$ queries. We follow the algorithm but instead when the algorithm issues a query, we branch to two parallel solutions instances with the two possible query answers from the oracle. Since the number of queries is $o(C_{OPT})$, the number of branches/solutions that we obtain by this process is at most $2^{o(C_{OPT})}$. We return the one which gives the minimum number of mistakes. This gives a contradiction to Lemma 5.2. ◀

6 Experiments

In this section, we report detailed experimental results on multiple synthetic and real-world datasets. We compare the performance of the existing correlation clustering algorithms that do not issue any queries, alongside with our new algorithms. We compare three existing algorithms: the deterministic constant factor approximation algorithm of Bansal et al. [7] (BBC), the combinatorial 3-approximation algorithm of Ailon et al. [1] (ACN), and the state-of-the-art 2.06-approximation algorithm of Chawla et al. based on linear program (LP) rounding [10] (LP-Rounding). The code and data used in our experiments can be found at `https://github.com/sanjayss34/corr-clust-query-esa2019`.

6.1 Datasets

Our datasets range from small synthetic datasets to large real datasets and real crowd answers obtained using Amazon Mechanical Turk. Below we give a short description of them.

Synthetics Datasets: Small

We generate graphs with ≈ 100 nodes by varying the cluster size distribution as follows. [N] represents 10 cliques whose sizes are drawn i.i.d. from a $Normal(8, 2)$ distribution. This generates clusters of nearly equal size. [S] represents 5 clusters of size 5 each, 4 clusters of size 15 each and one cluster of size 30. This generates clusters with moderate skew. [D] represents 3 cliques whose total size is 100 and whose individual sizes are determined by a draw from a $Dirichlet((3, 1, 1))$ distribution. This generates clusters with extreme skewed distribution with one cluster accounting for more than 80% of edges.

Synthetics Datasets: Large

We generate two datasets *skew* and *sqrtn* each containing 900 nodes of fictitious hospital patients data, including name, phone number, birth date and address using the data set generator of the Febrl system [11]. *skew* contains few ($\approx \log n$) clusters of large size ($\approx \frac{n}{\log n}$), moderate number of clusters ($\approx \sqrt{n}$) of moderate size ($\approx \sqrt{n}$) and a large tail of small clusters. *sqrtn* contains $\sqrt{n}$ clusters of size $\sqrt{n}$.

Noise Models for Synthetic Datasets

Initially, all intra-cluster edges are labelled with $+$ sign and all inter-cluster edges are labelled with $-$ sign. Next, the signs of a subset of edges are flipped according to the following distributions. Denote by $C_1, C_2, ..., C_k$ the clusters that we generate. Let N denote the number of vertices in a graph. Let $\ell_1 = 0.01$, $\ell_2 = 0.1$, and L be an integer. For the small datasets, we set $L = 100$, and for the large *skew* and *sqrtn* datasets, we set $L = \lfloor \ell_2 \binom{N}{2} \rfloor$.

I. Flip sign of L edges uniformly at random.

II. Flip sign of $\min\{\lfloor L/k \rfloor, |C_i| - 1\}$ edges uniformly at random within each clique C_i. Do not flip sign of the inter-cluster edges.

III. Flip sign of edges as in II in addition to selecting uniformly at random $\lceil \ell_1 |C_i||C_j| \rceil$ edges between each pair of cliques C_i, C_j and flipping their sign.

Real-World Datasets

We use several real-world datasets.

- In the *cora* dataset [20], each node is a scientific paper represented by a string determined by its title, authors, venue, and date; edge weights between nodes are computed using Jaro string similarity [14, 25]. The *cora* dataset consists of $1.9K$ nodes, 191 clusters with the largest cluster-size being 236.
- In the *gym* dataset [22], each node corresponds to an image of a gymnast, and each edge weight reflects the similarity of the two images (i.e. whether the two images correspond to the same person). The *gym* dataset consists of 94 nodes with 12 clusters and maximum cluster size is 15.
- In the *landmarks* dataset [17], each node corresponds to an image of a landmark in Paris or Barcelona, and the edge weights reflect the similarity of the two images. The *landmarks* dataset consists of 266 nodes, 13 clusters and the maximum size of clusters is 43.
- In the *allsports* dataset [23], the nodes correspond to images of athletes in one of several sports, and the edge weights reflect the similarity of the two images. The pairs of images across sports are easy to distinguish but the images within the same category of sport are quite difficult to distinguish due to various angles of the body, face and uniform. The *allsports* dataset consists of 200 nodes with 64 clusters and with a maximum size of cluster being just 5.

Since the underlying graphs are weighted, we convert the edge weights to ± 1 labels by simply labeling an edge $+$ if its weight is at least $1/2$ and $-$ otherwise (the edge weights in all of the weighted graphs are in $[0, 1]$). We also perform experiments directly on the weighted graphs [10] to show how the above rounding affects the results.

Oracle

For small datasets, we use the Gurobi (`www.gurobi.org`) optimizer to solve the integer linear program (ILP) for correlation clustering [10] to obtain the optimum solution, which is then used as an oracle. For larger datasets like *skew*, *sqrtn* and *cora*, ILP takes prohibitively long time to run. For these large datasets,the ground-truth clustering is available and is used as the oracle.

For practical implementation of oracles, one can use the available crowd-sourcing platforms such as the Amazon Mechanical Turk. It is possible that such an oracle may not always give correct answer. We also use such crowd-sourced oracle for experiments on real datasets. Each question is asked 3 to 5 times to Amazon Mechanical Turk, and a majority vote is taken to resolve any conflict among the answers. We emphasize that the same-cluster query setting can be useful in practice because two different sources of information can produce the edge signs and the oracle – for instance, the edge signs can be produced by a cheap, automated computational method (e.g. classifiers), while the oracle answers can be provided by humans through the crowd-sourcing mechanism explained above.

6.2 Results

We compare the results of our QueryPivot and RandomQueryPivot algorithm as well as the prior algorithms BBC [7], ACN [1] and LP-Rounding [10]. For the algorithms that are randomized (ACN, LP-Rounding and RandomQueryPivot), we report the average of three runs. The algorithm of Bansal et al. [7] requires setting a parameter δ. We tried several values of δ on several of the datasets and chose the value that seemed to give the best performance overall.

Table 1 Results for Experiments on synthetic small datasets. BBC denotes the algorithm of [7], ACN denotes the 3-approximation algorithm of [1], LP Rounding denotes the algorithm of [10], QP denotes QueryPivot, and RQP denotes RandomQueryPivot(0.25). All numerical columns except those marked as "Queries" give the number of mistakes made by the algorithm.

Mode	ILP Oracle	BBC	ACN	LP Rounding	QP	QP Queries	RQP	RQP Queries
N+I	100	271	205.67	100	100	113	104.33	63.3
N+II	48	104	70.0	48	48	47	56.33	27.33
N+III	93	201	130	123	93	91	97.67	66.0
D+I	100	100	267.0	100	100	86	100	83.33
D+II	48	48	144.33	48	48	57	48	40.67
D+III	64	64	216.33	64	64	71	64	75.0
S+I	100	969	206.0	100	100	136	100.67	92.0
S+II	60	831	100.67	61.67	60	71	63.67	58.67
S+III	137	913	297	141.33	137	159	139.33	107.33

Table 2 Results for Large Synthetic Datasets where Mistakes are measured with respect to ground-truth clustering and the oracle is the ground-truth clustering.

Dataset/Mode	LP Rounding	BBC	ACN	QP	QP Queries	RQP	RQP Queries
Skew (I)		8175	31197.67	0	17108	71.33	10051.33
Skew (II)		700	1182.33	60	668	282	558.0
Skew (III)		8175	12260.67	56	8977	293.0	4475.67
Sqrtn (I)		13050	36251.33	0	13171	9.67	7851.0
Sqrtn (II)		0	1484.67	0	748	0.0	711.0
Sqrtn (III)		13050	12711.33	0	6449	0.0	2693.33

Synthetic Datasets

Table 1 summarizes the results of different algorithms on small synthetic datasets.

As we observe, our QueryPivot algorithm always obtains the optimum clustering. Moreover, RandomQueryPivot has a performance very close to QueryPivot but often requires much less queries. Interestingly, the LP-rounding algorithm performs very well except for $N + III$. ACN and BBC algorithms have worse performance than LP-Rounding, and in most cases ACN is preferred over BBC.

For the larger synthetic datasets *skew* and *sqrt*, as discussed the ground-truth clustering is used as an oracle. We also use the ground-truth clustering to count the number of mistakes. On these datasets, the LP-rounding algorithm caused an out-of-memory error on a machine with 256 GB main memory that we used. The linear programming formulation for correlation clustering has $O(n^3)$ triangle inequality constraints; this results in very high time and space complexity rendering the LP-rounding impractical for correlation clustering on large datasets. Table 2 summarizes the results.

As we observe, QueryPivot algorithm recovers the exact ground-truth clustering in several cases. RandomQueryPivot has a low error rate as well and uses significantly fewer queries.

Table 3 Results for Real-World Datasets where mistakes are measured with respect to ground-truth clustering and the oracle is the ground-truth clustering. LP Rounding (weighted) refers to the LP rounding of [10] applied to the weighted input graph.

Dataset/ Mode	LP Rounding	LP Rounding (weighted)	BBC	ACN	QP	QP Queries	RQP	RQP Queries
Cora			62891	26065.0	4526	2188	4664.67	1474.33
Gym	221.0	332.67	449	301.67	8	150	82.67	97.33
Landmarks	29648.0	25790.0	31507	28770	3426	953	1124.67	1467.0
Allsports	230.0	226.33	227	253.33	217	41	223.67	21.0

Table 4 Running times (in seconds) for the results in Table 3. For randomized algorithms, the time shown is the average over three trials.

Dataset/Mode	LP Rounding	LP Rounding (weighted)	BBC	ACN	QP	RQP
Cora			1.58	0.16	2170.33	515.59
Gym	4.96	5.09	0.004	0.0014	0.33	0.27
Landmarks	190.96	9571.32	0.048	0.00067	1.96	2.82
Allsports	41.54	42.08	0.018	0.025	13.28	12.76

Real-World Datasets

The results for the real-world datasets are reported in Table 3, 5 and 6. It is evident from Table 3 that our algorithms outperform the existing algorithms by a big margin in recovering the original clusters. Table 3 also includes results for the LP-rounding algorithm applied to the original weighted graph for the Gym, Landmarks, and Allsports datasets. We also report in Table 4 the running times for the experiments in Table 3. These numbers show that the BBC and ACN algorithms are substantially faster than the others, while our algorithms are substantially faster than the LP-rounding algorithm.

Table 5 reports the results using a faulty crowd oracle. Contrasting the results of Table 3 and 5, we observe minimal performance degradation; that is, our algorithms are robust to noise. The results in this table are important, as this setting is closest to the typical real-world application of same-cluster queries. Note that the source of information that gives the signs of the edges is different from that which is the crowd oracle. For the landmarks dataset, the original edge weights are determined by a gist detector [21], while the oracle used in Table 5 is given by high-quality crowd workers. For the gym and allsports datasets, the original edge weights are determined by (lower quality) human crowd workers, but the oracle used in Table 5 is based on high-quality crowd workers. Finally, in Table 6, we report

Table 5 Results for Real-World Datasets where mistakes are measured with respect to ground-truth clustering and the oracle is the crowd.

Dataset/Mode	QP	QP Queries	RQP	RQP Queries
Gym	135	175	160.0	104.67
Landmarks	4645	1997	2172.33	1548.33
Allsports	218	41	223.67	21.0

Table 6 Results for Real-World Datasets where mistakes are measured with respect to the graph and the oracle is the optimal ILP solution for the graph.

Dataset/Mode	LP Rounding	BBC	ACN	QP	QP Queries	RQP	RQP Queries
Gym	276.0	464	338.0	207	171	211.0	112.67
Landmarks	4092.0	4995	5240.67	4092	267	4092.0	265.33
Allsports	33.33	65	40.67	28	36	30.33	18.67

the results using the optimum ILP solution as the oracle. For the larger datasets, it is neither possible to run the ILP nor LP-Rounding due to their huge space and time requirements. Again our algorithms perform the best, followed by the LP-rounding algorithm.

References

1 N. Ailon, M. Charikar, and A. Newman. Aggregating Inconsistent Information: Ranking and Clustering. *Symposium on the Theory of Computing (STOC)*, 2005.

2 Nir Ailon, Anup Bhattacharya, and Ragesh Jaiswal. Approximate Correlation Clustering Using Same-Cluster Queries. In *Latin American Symposium on Theoretical Informatics*, pages 14–27. Springer, 2018.

3 Nir Ailon, Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. Approximate clustering with same-cluster queries. *arXiv preprint*, 2017. arXiv:1704.01862.

4 Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)*, 55(5):23, 2008.

5 H. Ashtiani, S. Kushagra, and S. Ben-David. Clustering with Same-Cluster Queries. *Advances in Neural Information Processing Systems (NIPS)*, 2016.

6 M. F. Balcan and A. Blum. Clustering with Interactive Feedback. *International Conference on Algorithmic Learning Theory (ALT)*, 2008.

7 N. Bansal, A. Blum, and S. Chawla. Correlation Clustering. *Symposium on Foundations of Computer Science (FOCS)*, 2002.

8 Sebastian Böcker, Sebastian Briesemeister, Quang Bao Anh Bui, and Anke Truß. Going weighted: Parameterized algorithms for cluster editing. *Theoretical Computer Science*, 410(52):5467–5480, 2009.

9 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.

10 S. Chawla, K. Makarychev, T. Schramm, and G. Yaroslavtsev. Near Optimal LP Rounding Algorithm for Correlation Clustering on Complete and Complete k-partite Graphs. *Symposium on the Theory of Computing (STOC)*, pages 219–228, 2015.

11 Peter Christen. Febrl: a freely available record linkage system with a graphical user interface. In *Proceedings of the second Australasian workshop on Health data and knowledge management-Volume 80*, pages 17–25. Australian Computer Society, Inc., 2008.

12 Erik D Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187, 2006.

13 I. Dinur. Mildly exponential reduction from Gap 3SAT to polynomial-gap label-cover. *Electronic Colloquium on Computational Complexity (ECCC)*, 2016.

14 Donatella Firmani, Sainyam Galhotra, Barna Saha, and Divesh Srivastava. Robust Entity Resolution Using a CrowdOracle. *IEEE Data Eng. Bull.*, 41(2):91–103, 2018.

15 Buddhima Gamlath, Sangxia Huang, and Ola Svensson. Semi-Supervised Algorithms for Approximately Optimal and Accurate Clustering. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 57:1–57:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.57.

16 I. Giotis and V. Guruswami. Correlation Clustering with a Fixed Number of Clusters. *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

17 Anja Gruenheid, Besmira Nushi, Tim Kraska, Wolfgang Gatterbauer, and Donald Kossmann. Fault-tolerant entity resolution with the crowd. *arXiv preprint*, 2015. `arXiv:1512.00537`.

18 Shrinu Kushagra, Shai Ben-David, and Ihab Ilyas. Semi-supervised clustering for de-duplication. *arXiv preprint*, 2018. `arXiv:1810.04361`.

19 A. Mazumdar and B. Saha. Clustering with Noisy Queries. *Advances in Neural Information Processing Systems (NIPS)*, 2017.

20 Andrew McCallum. Data. URL: `https://people.cs.umass.edu/~mccallum/data.html`.

21 Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

22 Vasilis Verroios and Hector Garcia-Molina. Entity resolution with crowd errors. In *2015 IEEE 31st International Conference on Data Engineering*, pages 219–230. IEEE, 2015.

23 Vasilis Verroios, Hector Garcia-Molina, and Yannis Papakonstantinou. Waldo: An adaptive human interface for crowd entity resolution. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1133–1148. ACM, 2017.

24 David Williams. *Probability with martingales*. Cambridge university press, 1991.

25 William E Winkler. Overview of record linkage and current research directions. In *Bureau of the Census*. Citeseer, 2006.