Query-Efficient Correlation Clustering

David García–Soriano
d.garcia.soriano@isi.it
ISI Foundation
Turin, Italy

Konstantin Kutzkov
kutzkov@gmail.com
Amalfi Analytics
Barcelona, Spain

Francesco Bonchi
francesco.bonchi@isi.it
ISI Foundation, Turin, Italy
Eurecat, Barcelona, Spain

Charalampos Tsourakakis
ctsourak@bu.edu
Boston University
USA

ABSTRACT

Correlation clustering is arguably the most natural formulation of clustering. Given n objects and a pairwise similarity measure, the goal is to cluster the objects so that, to the best possible extent, similar objects are put in the same cluster and dissimilar objects are put in different clusters.

A main drawback of correlation clustering is that it requires as input the $\Theta(n^2)$ pairwise similarities. This is often infeasible to compute or even just to store. In this paper we study *query-efficient* algorithms for correlation clustering. Specifically, we devise a correlation clustering algorithm that, given a budget of Q queries, attains a solution whose expected number of disagreements is at most $3 \cdot \mathrm{OPT} + O(\frac{n^3}{Q})$, where OPT is the optimal cost for the instance. Its running time is $O(Q)$, and can be easily made non-adaptive (meaning it can specify all its queries at the outset and make them in parallel) with the same guarantees. Up to constant factors, our algorithm yields a provably optimal trade-off between the number of queries Q and the worst-case error attained, even for adaptive algorithms.

Finally, we perform an experimental study of our proposed method on both synthetic and real data, showing the scalability and the accuracy of our algorithm.

CCS CONCEPTS

• **Theory of computation → Graph algorithms analysis**; **Facility location and clustering**; **Active learning**.

KEYWORDS

correlation clustering, active learning, query complexity, algorithm design

ACM Reference Format:
David García–Soriano, Konstantin Kutzkov, Francesco Bonchi, and Charalampos Tsourakakis. 2020. Query-Efficient Correlation Clustering. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380220

1 INTRODUCTION

Correlation clustering [3] (or *cluster editing*) is a prominent clustering framework where we are given a set $V = [n]$ and a symmetric pairwise similarity function $\mathrm{sim} : \binom{V}{2} \to \{0, 1\}$, where $\binom{V}{2}$ is the set of unordered pairs of elements of V. The goal is to cluster the items in such a way that, to the best possible extent, similar objects are put in the same cluster and dissimilar objects are put in different clusters. Assuming that cluster identifiers are represented by natural numbers, a clustering ℓ is a function $\ell : V \longrightarrow \mathbb{N}$, and each cluster is a maximal set of vertices sharing the same label. Correlation clustering aims at minimizing the following cost:

$$\mathrm{cost}(\ell) = \sum_{\substack{(x,y)\in\binom{V}{2},\\ \ell(x)=\ell(y)}} (1 - \mathrm{sim}(x, y)) + \sum_{\substack{(x,y)\in\binom{V}{2},\\ \ell(x)\neq\ell(y)}} \mathrm{sim}(x, y). \qquad (1)$$

The intuition underlying the above problem definition is that if two objects x and y are dissimilar and are assigned to the same cluster we should pay a cost of 1, i.e., the amount of their dissimilarity. Similarly, if x, y are similar and they are assigned to different clusters we should pay also cost 1, i.e., the amount of their similarity $\mathrm{sim}(x, y)$. The correlation clustering framework naturally extends to non-binary, symmetric function, i.e., $\mathrm{sim} : \binom{V}{2} \to [0, 1]$. In this paper we focus on the binary case; the general non-binary case can be efficiently reduced to this case at a loss of only a constant factor in the approximation [3, Thm. 23]. The binary setting can be viewed very conveniently through graph-theoretic lenses: the n items correspond to the vertices of a *similarity graph* G, which is a complete undirected graph with edges labeled "+" or "-". An edge e causes a *disagreement* (of *cost* 1) between the similarity graph and a clustering when it is a "+" edge connecting vertices in different clusters, or a "–" edge connecting vertices within the same cluster. If we were given a *cluster graph* [22], i.e., a graph whose set of positive edges is the union of vertex-disjoint cliques, we would be able to produce a perfect (i.e., cost 0) clustering simply by computing the connected components of the positive graph. However, similarities will generally be inconsistent with one another, so incurring a certain cost is unavoidable. Correlation clustering aims at minimizing such cost. The problem may be viewed as the task of finding the equivalence relation that most closely resembles a given symmetric relation. The correlation clustering problem is NP-hard [3, 22].

Correlation clustering is particularly appealing for the task of clustering structured objects, where the similarity function is domain-specific. A typical application is clustering web-pages based on similarity scores: for each pair of pages we have a score between 0 and 1, and we would like to cluster the pages so that pages with a high similarity score are in the same cluster, and pages with a low similarity score are in different clusters. The technique is applicable to a multitude of problems in different domains, including duplicate detection and similarity joins [13, 17], spam detection [6, 20], co-reference resolution [19], biology [4, 8], image segmentation [18], social networks [7], and clustering aggregation [16]. A key feature of correlation clustering is that it does not require the number of clusters as part of the input; instead it automatically finds the optimal number, performing model selection.

Despite its appeal, the main practical drawback of correlation clustering is the fact that, given n items to be clustered, $\Theta(n^2)$ similarity computations are needed to prepare the similarity graph that serves as input for the algorithm. In addition to the obvious algorithmic cost involved with $\Theta(n^2)$ queries, in certain applications there is an additional type of cost that may render correlation clustering algorithms impractical. Consider the following motivating real-world scenarios. In biological sciences, in order to produce a network of interactions between a set of biological entities (e.g., proteins), a highly trained professional has to devote time and costly resources (e.g., equipment) to perform tests between all $\binom{n}{2}$ pairs of entities. In entity resolution, a task central to data integration and data cleaning [23], a crowdsourcing-based approach performs queries to workers of the form "does the record x represent the same entity as the record y?". Such queries to workers involve a monetary cost, so it is desirable to reduce their number. In both scenarios developing clustering tools that use fewer than $\binom{n}{2}$ queries is of major interest. This is the main motivation behind our work. At a high level we answer the following question:

Problem 1. *How to design a correlation clustering algorithm that outputs a good approximation in a* query-efficient *manner: i.e., given a budget of Q queries, the algorithm is allowed to learn the specific value of* $\mathrm{sim}(i, j) \in \{0, 1\}$ *for up to Q queries (i, j) of the algorithm's choice.*

Contributions. The main contributions of this work are summarized as follows:

- We design a computationally efficient randomized algorithm QECC that, given a budget of Q queries, attains a solution whose expected number of disagreements is at most $3 \cdot \mathrm{OPT} + O(\frac{n^3}{Q})$, where OPT is the optimal cost of the correlation clustering instance (Theorem 3.1). We can achieve this via a *non-adaptive* algorithm (Theorem 3.4).
- We show (Theorem 4.1) that up to constant factors, our algorithm is optimal, even for adaptive algorithms: any algorithm making Q queries must make at least $\Omega(\mathrm{OPT} + \frac{n^3}{Q})$ errors.
- We give a simple, intuitive heuristic modification of our algorithm QECC-heur which helps reduce the error of the algorithm in practice (specifically the recall of positive edges), thus partially bridging the constant-factor gap between our lower and upper bounds.
- We present an experimental study of our two algorithms, compare their performance with a baseline based on affinity propagation, and study their sensitivity to parameters such as graph size, number of clusters, imbalance, and noise.

2 RELATED WORK

We review briefly the related work that lies closest to our paper.

Correlation clustering. The correlation clustering problem is NP-hard [3, 22] and, in its minimizing disagreements formulation used above, is also APX-hard [11], so we do not expect a polynomial-time approximation scheme. Nevertheless, there are constant-factor approximation algorithms [1, 3, 11]. Ailon et al. [1] present QwickCluster, a simple, elegant 3-approximation algorithm. They improve the approximation ratio to 2.5 by utilizing an LP relaxation of the problem; the best approximation factor known to date is 2.06, due to Chawla et al. [12]. The interested reader may refer to the extensive survey due to Bonchi, García-Soriano and Liberty [6].

Query-efficient algorithms for correlation clustering. Query-efficient correlation clustering has received less attention. There exist two categories of algorithms, non-adaptive and adaptive. The former choose their queries before-hand, while the latter can select the next query based on the response to previous queries.

In an earlier preprint [5] we initiated the study of query-efficient correlation clustering. Our work there focused on a stronger local model which requires answering cluster-id queries quickly, i.e., outputting a cluster label for each given vertex by querying at most q edges per vertex. Such a q-query local algorithm allows a global clustering of the graph with qn queries; hence upper bounds for local clustering imply upper bounds for global clustering, which is the model we consider in this paper. The algorithm from [5] is non-adaptive, and the upper bounds we present here (Theorems 3.1 and 3.4) may be recovered by setting $\epsilon = \frac{n}{Q}$ in [5, Thm. 3.3]. A matching lower bound was also proved in [5, Thm. 6.1], but the proof therein applied only to non-adaptive algorithms. In this paper we present a self-contained analysis of the algorithm from [5] in the global setting (Theorems 3.1 and 3.4) and strengthen the lower bound so that it applies also to adaptive algorithms (Theorem 4.1). Additionally, we perform an experimental study of the algorithm.

Some of the results from [5] have been rediscovered several years later (in a weaker form) by Bressan, Cesa-Bianchi, Paudice, and Vitale [9]. They study the problem of query-efficient correlation clustering (Problem 1) in the adaptive setting, and provide a query-efficient algorithm, named ACC. The performance guarantee they obtain in [9, Thm. 1] is asymptotically the same that had already been proven in [5], but it has worse constant factors and is attained via an adaptive algorithm. They also modify the lower bound proof from [5] to make it adaptive [9, Thm. 9], and present some new results concerning the cluster-recovery properties of the algorithm.

In terms of techniques, the only difference between our algorithm QECC and the ACC algorithm from [9] is that the latter adds a check that discards pivots when no neighbor is found after inspecting a random sample of size $f(n-1) = Q/(n-1)$. This additional check is unnecessary from a theoretical viewpoint (see Theorem 3.1) and it has the disadvantage that it necessarily results in an adaptive algorithm. Moreover, the analysis of [9] is significantly more complex than ours, because they need to adapt the proof of the approximation guarantees of the QwickCluster algorithm from [1] to take into account the additional check, whereas we simply take the approximation guarantee as given and argue that stopping QwickCluster after k pivots have been selected only incurs an expected additional cost of $n^2/(2k)$ (Lemma 3.3).

3 ALGORITHM AND ANALYSIS

Before presenting our algorithm, we describe in greater detail the elegant algorithm due to Ailon et al. [1] for correlation clustering, as it lies close to our proposed method.

QwickCluster algorithm. The QwickCluster algorithm selects a random pivot v, creates a cluster with v and its positive neighborhood, removes the cluster, and iterates on the induced remaining subgraph. Essentially it finds a maximal independent set in the positive graph in random order. The elements in this set serve as cluster centers (pivots) in the order in which they were found. In the pseudocode below, $\Gamma_G^+(v)$ denotes the set of vertices to which there is a positive edge in G from v.

Algorithm 1 QwickCluster

```
Input: $G = (V, E)$, a complete graph with "+,-" edge labels
  $R \leftarrow V$                                ▷ Unclustered vertices so far
  while $R \neq \emptyset$ do
      Pick a pivot $v$ from $R$ uniformly at random.
      Output cluster $C = \{v\} \cup \Gamma_G^+(v) \cap R$.
      $R \leftarrow V \setminus C$
```

When the graph is clusterable, QwickCluster makes no mistakes. In [1], the authors show that the expected cost of the clustering found by QwickCluster is at most 3 OPT, where OPT denotes the optimal cost.

QECC. Our algorithm QECC (Query-Efficient Correlation Clustering) runs QwickCluster until the query budget Q is complete, and then outputs singleton clusters for the remaining unclustered vertices. The following subsection is devoted to the proof of our main result, stated next.

Algorithm 2 QECC

```
Input: $G = (V, E)$; query budget $Q$
  $R \leftarrow V$                                ▷ Unclustered vertices so far
  while $R \neq \emptyset \wedge Q \geq |R| - 1$ do
      Pick a pivot $v$ from $R$ uniformly at random.
      Query all pairs $(v, w)$ for $w \in R \setminus \{v\}$ to determine $\Gamma_G^+(v) \cap R$.
      $Q \leftarrow Q - |R| + 1$
      Output cluster $C = \{v\} \cup \Gamma_G^+(v) \cap R$.
      $R \leftarrow V \setminus C$
  Output a separate singleton cluster for each remaining $v \in R$.
```

Theorem 3.1. *Let G be a graph with n vertices. For any $Q > 0$, Algorithm QECC finds a clustering of G with expected cost at most $3 \cdot \mathrm{OPT} + \frac{n^3}{2Q}$ making at most Q edge queries. It runs in time $O(Q)$ assuming unit-cost queries.*

3.1 Analysis of QECC

For simplicity, in the rest of this section we will identify a complete "+,-" labeled graph G with its graph of *positive* edges (V, E^+), so that queries correspond to querying a pair of vertices for the existence of an edge. The set of (positive) neighbors of v in a graph $G = (V, E)$ will be denoted $\Gamma(v)$; a similar notation is used for the set $\Gamma(S)$ of positive neighbors of a set $S \subseteq V$. The cost of the optimum clustering for G is denoted OPT. When ℓ is a clustering, $\mathrm{cost}(\ell)$ denotes the cost (number of disagreements) of this clustering, defined by (1) with $\mathrm{sim}(x, y) = 1$ iff $\{x, y\} \in E$.

In order to analyze QECC, we need to understand how early stopping of QwickCluster affects the accuracy of the clustering found. For any non-empty graph G and pivot $v \in V(G)$, let $N_v(G)$ denote the subgraph of G resulting from removing all edges incident to $\Gamma(v)$ (keeping all vertices). Define a random sequence $G_0, G_1, \ldots$ of graphs by $G_0 = G$ and $G_{i+1} = N_{v_{i+1}}(G_i)$, where $v_1, v_2, \ldots$ are chosen independently and uniformly at random from $V(G_0)$. Note that $G_{i+1} = G_i$ if at step i a vertex is chosen for a second time.

The following lemma is key:

Lemma 3.2. *Let G_i have average degree $\tilde{d}$. When going from G_i to G_{i+1}, the number of edges decreases in expectation by at least $\binom{\tilde{d}+1}{2}$.*

Proof. Let $V = V(G_0)$, $E = E(G_i)$ and let $d_u = |\Gamma(u)|$ denote the degree of $u \in V$ in G_i. Consider an edge $\{u, v\} \in E$. It is deleted if the chosen pivot v_i is an element of $\Gamma(u) \cup \Gamma(v)$ (which contains u and v). Let X_{uv} be the 0-1 random variable associated with this event, which occurs with probability

$$\mathbb{E}[X_{uv}] = \frac{|\Gamma(u) \cup \Gamma(v)|}{n} \geq \frac{1 + \max(d_u, d_v)}{n} \geq \frac{1}{n} + \frac{d_u + d_v}{2n}.$$

Let $D = \sum_{u<v \mid \{u,v\} \in E} X_{uv}$ be the number of edges deleted (we assume an ordering of V to avoid double-counting edges). By linearity of expectation,

$$\begin{aligned}
\mathbb{E}[D] &= \sum_{\substack{u<v \\ \{u,v\} \in E}} \mathbb{E}[X_{uv}] = \frac{1}{2} \sum_{\substack{u,v \in V \\ \{u,v\} \in E}} \mathbb{E}[X_{uv}] \\
&\geq \frac{1}{2} \sum_{\substack{u,v \\ \{u,v\} \in E}} \left(\frac{1}{n} + \frac{d_u + d_v}{2n} \right) \\
&= \frac{\tilde{d}}{2} + \frac{1}{4n} \sum_{\substack{u,v \\ \{u,v\} \in E}} (d_u + d_v).
\end{aligned}$$

Now we compute

$$\begin{aligned}
\frac{1}{4n} \sum_{\substack{u,v \\ \{u,v\} \in E}} (d_u + d_v) &= \frac{1}{2n} \sum_{\substack{u,v \\ \{u,v\} \in E}} d_u = \frac{1}{2n} \sum_u d_u^2 \\
&= \frac{1}{2} \mathop{\mathbb{E}}_{u \sim V}[d_u^2] \geq \frac{1}{2} \left(\mathop{\mathbb{E}}_{u \sim V}[d_u] \right)^2 = \frac{1}{2} \tilde{d}^2,
\end{aligned}$$

where in the last line, $\sim$ denotes uniform sampling and we used the Cauchy-Schwarz inequality. Hence $\mathbb{E}[D] \geq \frac{\tilde{d}}{2} + \frac{\tilde{d}^2}{2} = \binom{\tilde{d}+1}{2}$. □

Lemma 3.3. *Let G be a graph with n vertices and let $P = \{v_1, \ldots, v_r\}$ be the first r pivots chosen by running QwickCluster on G. Then the* *expected number of positive edges of G not incident with an element of $P \cup \Gamma(P)$ is less than $\frac{n^2}{2(r+1)}$.*

Proof. Recall that at each iteration QwickCluster picks a random pivot from R. This selection is equivalent to picking a random pivot v from the original set of vertices V and discarding it if $v \notin R$, repeating until some $v \in R$ is found, in which case a new pivot is added. Consider the following modification of QwickCluster, denoted SluggishCluster, which picks a pivot v at random from V but always increases the counter r of pivots found, even if $v \in R$ (ignoring the cluster creation step if $v \notin R$). We can couple both algorithms into a common probability space where each point ω contains a sequence of randomly selected vertices and each algorithm picks the next one in sequence. For any ω, whenever the first r pivots of SluggishCluster are $S = (v_1, \dots, v_r)$, then the first r' pivots of QwickCluster are the sequence S' obtained from S by removing previously appearing elements, where $r' = |S'|$. Hence $|V \setminus (S \cup \Gamma(S))| = |V \setminus (S' \cup \Gamma(S'))|$ and $r' \le r$. Thus the number of edges not incident with the first r pivots and their neighbors in SluggishCluster stochastically dominates the number of edges not incident with the first r pivots and their neighbors in SluggishCluster, since both numbers are decreasing with r.

Therefore it is enough to prove the claim for SluggishCluster. Let $n = |V(G_0)|$ and define $\alpha_i \in [0,1]$ by $\alpha_i = \frac{2\cdot|E(G_i)|}{n^2}$. We claim that for all $i \ge 1$ the following inequalities hold:

$$\mathbb{E}[\alpha_i \mid G_0, \dots, G_{i-1}] \le \alpha_{i-1}(1-\alpha_{i-1}), \tag{2}$$

$$\mathbb{E}[\alpha_i] \le \mathbb{E}[\alpha_{i-1}](1-\mathbb{E}[\alpha_{i-1}]), \tag{3}$$

$$\mathbb{E}[\alpha_i] < \frac{1}{i+1}. \tag{4}$$

Indeed, G_i is a random function of G_{i-1} only, and the average degree of G_{i-1} is $\widetilde{d}_{i-1} = \alpha_{i-1}n$ so, by Lemma 3.2,

$$\mathbb{E}[2\cdot|E(G_i)| \mid G_{i-1}] \le \alpha_{i-1}n^2 - 2\cdot\frac{1}{2}\widetilde{d}_{i-1}^2 = n^2\alpha_{i-1}(1-\alpha_{i-1}),$$

proving (2). Now (3) now follows from Jensen's inequality: since

$$\mathbb{E}[\alpha_i] = \mathbb{E}\left[\mathbb{E}[\alpha_i \mid G_0, \dots, G_{i-1}]\right] \le \mathbb{E}[\alpha_{i-1}(1-\alpha_{i-1})]$$

and the function $g(x) = x(1-x)$ is concave in $[0,1]$, we have

$$\mathbb{E}[\alpha_i] \le \mathbb{E}[g(\alpha_{i-1})] \le g(\mathbb{E}[\alpha_{i-1}]) = \mathbb{E}[\alpha_{i-1}](1-\mathbb{E}[\alpha_{i-1}]).$$

Finally we prove $\mathbb{E}[\alpha_i] < 1/(i+1)\ \forall i \ge 1$. For $i = 1$, we have:

$$\mathbb{E}[\alpha_1] \le g(\alpha_0) \le \max_{x\in[0,1]} g(x) = g\left(\frac{1}{2}\right) = \frac{1}{4} < \frac{1}{2}.$$

For $i > 1$, observe that g is increasing on $[0, 1/2]$ and

$$g\left(\frac{1}{i}\right) = \frac{1}{i} - \frac{1}{i^2} \le \frac{1}{i} - \frac{1}{i(i+1)} = \frac{1}{i+1},$$

so (4) follows from (3) by induction on i:

$$\mathbb{E}[\alpha_{i-1}] < \frac{1}{i} \implies \mathbb{E}[\alpha_i] \le g\left(\frac{1}{i}\right) \le \frac{1}{i+1}.$$

Therefore $\mathbb{E}[|E(G_r)|] = \frac{1}{2}\mathbb{E}[\alpha_r]n^2 \le \frac{n^2}{2(r+1)}$, as we wished to show. □

We are now ready to prove Theorem 3.1:

Proof of Theorem 3.1. Let OPT denote the cost of the optimal clustering of G and let C_r be a random variable denoting the clustering obtained by stopping QwickCluster after r pivots are found (or running it to completion if it finds r pivots or less), and putting all unclustered vertices into singleton clusters. Note that whenever C_i makes a mistake on a negative edge, so does C_j for $j \ge i$; on the other hand, every mistake on a positive edge by C_i is either a mistake by C_j ($j \ge i$) or the edge is not incident to any of the vertices clustered in the first i rounds. By Lemma 3.3, there are at most $\frac{n^2}{2(i+1)}$ of the latter in expectation. Hence $\mathbb{E}[\text{cost}(C_i)] - \mathbb{E}[\text{cost}(C_n)] \le \frac{n^2}{2(i+1)}$.

Algorithm QECC runs for k rounds, where $k \ge \lfloor \frac{Q}{n-1} \rfloor > \frac{Q}{n} - 1$ because each pivot uses $|R| - 1 \le n-1$ queries. Then

$$\mathbb{E}[\text{cost}(C_k)] - \mathbb{E}[\text{cost}(C_n)] < \frac{n^2}{2(k+1)} < \frac{n^3}{2Q}.$$

On the other hand, we have $\mathbb{E}[\text{cost}(C_n)] \le 3\cdot\text{OPT}$ because of the expected 3-approximation guarantee of QwickCluster from [1]. Thus $\mathbb{E}[\text{cost}(C_k)] \le 3\,\text{OPT} + \frac{n^3}{2Q}$, proving our approximation guarantee.

Finally, the time spent inside each iteration of the main loop is dominated by the time spent making queries to vertices in R, since this number also bounds the size of the cluster found. Therefore the running time of QECC is $O(Q)$. □

3.2 A non-adaptive algorithm.

Our algorithm QECC is adaptive in the way we have chosen to present it: the queries made when picking a second pivot depend on the result of the queries made for the first pivot. However, this is not necessary: we can instead query for the neighborhood of a random sample S of size $\frac{Q}{n-1}$. If we use the elements of S to find pivots, the same analysis shows that the output of this variant meets the exact same error bound of $3\,\text{OPT} + n^3/(2Q)$. For completeness, we include pseudocode for the adaptive variant of QECC below (Algorithm 3).

In practice the adaptive variant we have presented in Algorithm 2 will run closer to the query budget, choosing more pivots and reducing the error somewhat below the theoretical bound, because it does not "waste" queries between a newly found pivot and the neighbors of previous pivots. Nevertheless, in settings where the similarity computations can be performed in parallel, it may become advantageous to use Non-adaptive QECC. Another benefit of the non-adaptive variant is that it gives a one-pass streaming algorithm for correlation clustering that uses only $O(Q)$ space and processes edges in arbitrary order.

Theorem 3.4. *For any $Q > 0$, Algorithm Non-adaptive QECC finds a clustering of G with expected cost at most $3\cdot\text{OPT} + \frac{n^3}{2Q}$ making at most Q non-adaptive edge queries. It runs in time $O(Q)$ assuming unit-cost queries.*

Proof. The number of queries it makes is $S = (n-1) + (n-2) + \dots (n-k) = \frac{2n-1-k}{2}k \le Q$. Note that $\frac{n-1}{2}k \le S \le Q \le (n-1)k$. The proof of the error bound proceeds exactly as in the proof of Theorem 3.1 (because $k \ge \frac{Q}{n-1}$). The running time of the querying phase of Non-adaptive QECC is $O(Q)$ and, assuming a hash table is used to store query answers, the expected running time of the second phase is bounded by $O(nk) = O(Q)$, because $k \leq \frac{2Q}{n-1}$. □

Another interesting consequence of this result (coupled with our lower bound, Theorem 4.1), is that adaptivity does not help for correlation clustering (beyond possibly a constant factor), in stark contrast to other problems where an exponential separation is known between the query complexity of adaptive and non-adaptive algorithms (e.g., [10, 14]).

Algorithm 3 Non-adaptive QECC

```
Input: $G = (V, E)$; query budget $Q$
  $k \leftarrow \max\{t \leq n \mid (2n-1-t)t \leq 2Q\}$.
  Let $S = (s_1, \dots, s_k)$ be a uniform random sample from $V$
   (with or without replacement)

  ▷ Querying phase: find $\Gamma^+_G(v)$ for each $v \in S$
  for each $v \in S, w \in V, v < w$ do
      Query $(v, w)$

  ▷ Clustering phase
  $R \leftarrow V$
  $i \leftarrow 1$
  while $R \neq \emptyset \wedge i \leq k$ do
      if $s_i \in R$ then
          Output cluster $C = \{s_i\} \cup \Gamma^+_G(s_i) \cap R$.
          $R \leftarrow V \setminus C$
      $i \leftarrow i + 1$
  Output a separate singleton cluster for each remaining $v \in R$.
```

4 LOWER BOUND

In this section we show that QECC is essentially optimal: for any given budget of queries, no algorithm (adaptive or not) can find a solution better than that of QECC by more than a constant factor.

Theorem 4.1. *For any $c \geq 1$ and T such that $8n < T \leq \frac{n^2}{2048c^2}$, any algorithm finding a clustering with expected cost at most $c \cdot \mathrm{OPT} + T$ must make at least $\Omega(\frac{n^3}{Tc^2})$ adaptive edge similarity queries.*

Note that this also implies that any purely multiplicative approximation guarantee needs $\Omega(n^2)$ queries (e.g. by taking $T = 10n$).

Proof. Let $\epsilon = \frac{T}{n^2}$; then $\frac{1}{n} < \epsilon \leq \frac{1}{2048c^2}$. By Yao's minimax principle [25], it suffices to produce a distribution $\mathcal{G}$ over graphs with the following properties:

- the expected cost of the optimal clustering of $G \sim \mathcal{G}$ is $\mathbb{E}[\mathrm{OPT}(G)] \leq \frac{\varepsilon n^2}{c}$;
- for any deterministic algorithm making fewer than $L/2 = \frac{n}{2048\varepsilon c^2}$ queries, the expected cost (over $\mathcal{G}$) of the clustering produced exceeds $2\varepsilon n^2 \geq c \cdot \mathbb{E}[\mathrm{OPT}(G)] + T$.

Let $\alpha = \frac{1}{4c}$ and $k = \frac{1}{32c\varepsilon}$. We can assume that c, k and $\alpha n/k$ are integral (here we use the fact that $\varepsilon > 1/n$). Let $A = \{1, \dots, (1-\alpha)n\}$ and $B = \{(1-\alpha)n + 1, \dots, n\}$.

Consider the following distribution $\mathcal{G}$ of graphs: partition the vertices of A into exactly k equal-sized clusters $C_1, \dots, C_k$. The set of positive edges will be the union of the cliques defined by $C_1, \dots, C_k$, plus edges joining each vertex $v \in B$ to all the elements of C_{r_v} for a randomly chosen $r_v \in [k]$; r_v is chosen independently of r_w for all $w \neq v$.

Define the *natural clustering* of a graph $G \in \mathcal{G}$ by the classes $C'_i = C_i \cup \{v \in B \mid r_v = i\}$ $(i \in [k])$. We view N also as a graph formed by a disjoint union of the k cliques determined by $\{C'_i\}_{i \in [k]}$. This clustering will have a few disagreements because of the negative edges between different vertices $v, w \in B$ with $r_v = r_w$. For any pair of distinct elements $v, w \in B$, this happens with probability $1/k$. The cost of the optimal clustering of G is bounded by that of the natural clustering N, hence

$$\mathbb{E}[\mathrm{OPT}] \leq \mathbb{E}[\mathrm{cost}(N)] = \frac{\binom{\alpha n}{2}}{k} \leq \frac{\alpha^2 n^2}{2k} = \frac{\varepsilon}{c} n^2.$$

We have to show that any algorithm making $< L/2$ queries to graphs drawn from $\mathcal{G}$ produces a clustering with expected cost larger than $2\varepsilon n^2$. Since all graphs in $\mathcal{G}$ induce the same subgraphs on A and B separately, we can assume without loss of generality that the algorithm queries only edges between A and B. Note that the neighborhoods in G of every pair of vertices from the same C_i are the same: $\Gamma^+_G(u) = \Gamma^+_G(v)$ and $\Gamma^-_G(u) = \Gamma^-_G(v)$ for all $u, v \in C_i$, $i \in [k]$; moreover, u and v are joined by a positive edge. Therefore, if $u, v \in C_i$ but the algorithm assigns u and v to different clusters, either moving u to v's cluster or v to u's cluster will not decrease the cost. All in all, we can assume that the algorithm outputs k clusters $C'_1, \dots, C'_k$ with $C_i \subseteq C'_i$ for all i, plus (possibly) some clusters $C'_{k+1}, \dots, C'_{k'}$ $(k' \geq k)$ involving only elements of B.

For $v \in B$, let $s_v \in [k']$ denote the cluster that the algorithm assigns v to. For every $v \in B$, let G_v denote the event that the algorithm queries (u, v) for some $u \in C_{r_v}$ and, whenever G_v does not hold, let us add a "fictitious" query to the algorithm between v and some arbitrary element of C_{s_v}. This ensures that whenever $r_v = s_v$, the last query of the algorithm verifies its guess and returns 1 if the correct cluster has been found. This adds at most $|B| \leq n \leq \frac{L}{2}$ queries in total. Let $Q_1, Q_2, \dots, Q_z$ be the (random) sequence of queries issued by the algorithm and let $i^v_1, i^v_2, \dots, i^v_{T_v}$ be the indices of those queries involving a fixed vertex $v \in B$. Note that r_v is independent of the response to all queries not involving v and, conditioned on the result of all queries up to time $t < i_{T_v}$, r_v is uniformly distributed among the set $\{i \in [k] \mid (Q_j \notin C_i \forall j < t)\}$, whose size is upper-bounded by $k - t + 1$. Therefore

$$\Pr[Q_{i^v_t} \in C_{r_v} \mid Q_1, \dots, Q_{i^v_{t-1}}] \leq \frac{1}{k - t + 1},$$

which becomes an equality if the algorithm does not query the same cluster twice. It follows by induction that

$$\Pr[\{Q_{i^v_1}, \dots, Q_{i^v_t}\} \cap C_{r_v} \neq \emptyset] \leq \frac{t}{k}. \tag{5}$$

Let M_v be the event that the algorithm makes more than $k/2$ queries involving v. The event $r_v = s_v$ is equivalent to G_v, i.e., the event $\{Q_{i^v_1}, \dots, Q_{i^v_{T_v}}\} \cap C_{r_v} \neq \emptyset$, because of our addition of one fictitious query for v. We have

$$\Pr[r_v = s_v] = \Pr[G_v] \leq \Pr[M_v] + \Pr[G_v \wedge \overline{M}_v].$$

In other words, either the algorithm makes many queries for v, or it hits the correct cluster with few queries. (Without fictitious queries, we would have to add a third term for the probability that the algorithm picks by chance the correct s_v.) We will use the first term $\Pr[M_v]$ to control the expected query complexity. The second term, $\Pr[G_v \wedge \overline{M}_v]$, is bounded by $\frac{1}{2}$ by (5) because $T_v \leq k/2$ whenever $\overline{M}_v$ holds. Hence

$$\Pr[r_v \neq s_v] \geq \frac{1}{2} - \Pr[M_v],$$

so

$$\mathbb{E}[|\{v \in B \mid r_v \neq s_v\}|] = \sum_{v \in B} \Pr[r_v \neq s_v] \geq \frac{\alpha n}{2} - \left(\sum_{v \in B} \Pr[M_v]\right).$$

Each vertex $v \in B$ with $s_v \neq r_v$, causes disagreements with all of $C_{r_v} \subseteq C'_{r_v}$ and $C_{s_v} \subseteq C'_{r_v}$, introducing at least $2|A|/k \geq n/k$ new disagreements.

If we denote by X the cost of the clustering found and by Z the number of queries made, we have

$$\begin{aligned}\mathbb{E}[X] &\geq \frac{n}{k}\,\mathbb{E}[|\{v \in B \mid r_v \neq s_v\}|] \\ &\geq \frac{\alpha n^2}{2k} - \frac{n}{k}\left(\sum_{v \in B} \Pr[M_v]\right) \\ &= 4\epsilon n^2 - \frac{n}{k}\left(\sum_{v \in B} \Pr[M_v]\right).\end{aligned}$$

In particular, if $\mathbb{E}[X] \leq 2\epsilon n^2$, then we must have

$$\sum_{v \in B} \Pr[M_v] \geq \frac{2\epsilon n^2}{n/k} = 2\epsilon n k = \frac{n}{16c}.$$

But then we can lower bound the expected number of queries by

$$\mathbb{E}[Z] \geq \frac{k}{2} \sum_{v \in B} \Pr[M_v] \geq \frac{nk}{32c} = \frac{n}{1024c^2\epsilon} = L = \frac{n^3}{1024c^2 T},$$

of which at most $L/2$ are the fictitious queries we added. This completes the proof.

□

5 A PRACTICAL IMPROVEMENT

As we will see in Section 6, algorithm QECC, while provably optimal up to constant factors, sometimes returns solutions with poor recall of positive edges when the query budget is low. Intuitively, the reason is that, while picking a random pivot works in expectation, sometimes a low-degree pivot is chosen and all $|R| - 1$ queries are spent querying its neighbors, which may not be worth the effort for a small cluster when the query budget is tight. To entice the algorithm to choose higher-degree vertices (which would also improve the recall), we propose to bias it so that pivots are chosen with probability proportional to their positive degree in the subgraph induced by R. The conclusion of Lemma 3.3 remains unaltered in this case, but whether this change preserves the approximation guarantees from [1] on which we rely is unclear. In practice, this heuristic modification consistently improves the recall on all the tests we performed, as well as the total number of disagreements in most cases.

We cannot afford to compute the degree of each vertex with a small number of queries, but the following scheme is easily seen to choose each vertex $u \in R$ with probability $d_u/(2E)$, where d_u is the degree of u in the subgraph $G[R]$ induced by R, and $E > 0$ is the total number of edges in $G[R]$:

(1) Pick random pairs of vertices to query $(u, v) \in R \times R$ until an edge $(u, v) \in E$ is found;
(2) Select the first endpoint u of this edge as a pivot.

When $E = 0$, this procedure will simply run out of queries to make.

Pseudocode for QECC-HEUR is shown below.

Algorithm 4 QECC-HEUR

```
Input: $G = (V, E)$; query budget $Q$
  $R \leftarrow V$                                ▷ Unclustered vertices so far
  while $|R| > 1 \wedge Q \geq |R| - 1$ do
      Pick a pair $(u, v)$ from $R \times R$ uniformly at random.

      if $u \neq v$ then
          Query $(u, v)$
          $Q \leftarrow Q - 1$

          if $(u, v) \in E$ then
              Query all pairs $(v, w)$ for $w \in R \setminus \{u, v\}$
                to determine $\Gamma^+_G(v) \cap R$.
              $Q \leftarrow Q - |R| + 2$.
              Output cluster $C = \{v\} \cup \Gamma^+_G(v) \cap R$.
              $R \leftarrow V \setminus C$
  Output a separate singleton cluster for each remaining $v \in R$.
```

6 EXPERIMENTS

In this section we present the results of our experimental evaluations of QECC and QECC-HEUR, on both synthetic and real-world graphs. We view an input graph as defining the set of positive edges; missing edges are interpreted as negative edges.

6.1 Experimental setup

Clustering quality measures. We evaluate the clustering S produced by QECC and QECC-HEUR in terms of total *cost* (number of disagreements), *precision* of positive edges (ratio between the number of positive edges between pairs of nodes clustered together in S and the total number of pairs of vertices clustered together in S), and *recall* of positive edges (ratio between the number of positive edges between pairs of nodes clustered together in S and the total number of positive edges in G). Although our algorithms have been designed to minimize total cost, we deem it important to consider precision and recall values to detect extreme situations in which, for example, a graph is clustered into n singletons clusters which, if the graph is very sparse, may have small cost, but very low recall. All but one of the graphs G we use are accompanied with a ground-truth clustering (by design in the case of synthetic graphs), which we compare against.

Baseline. As QECC is the first query-efficient algorithm for correlation clustering, any baseline must be based on another clustering method. We turn to affinity propagation methods, in which a matrix of similarities (affinities) are given as input, and then messages about the "availability" and "responsibility" of vertices as possible cluster centers are transmitted along the edges of a graph, until a

Table 1: Dataset characteristics: name, type, size and ground truth error measures.

Dataset	Type	|V|	|E|	# clusters	GT cost	GT precision	GT recall
S(2000,20,0.15,2)	synthetic	2,000	104,985	20	30,483	0.859	0.852
Cora	real	1,879	64,955	191	23,516	0.829	0.803
Citeseer	real	3,327	4,552	-	-	-	-
Mushrooms	real	8,123	18,143,868	2	11,791,251	0.534	0.683

high-quality set of cluster pivots is found; see [15]. We design the following query-efficient procedure as a baseline:

(1) Pick k random vertices without replacement and query their complete neighborhood. Here k is chosen as high as possible within the query budget Q, i.e.,

$$k = \arg\max\{t \mid (2n - t - 1)t/2 \leq Q\}.$$

(2) Set the affinity of any pair of vertices queried to 1 if there exists an edge.
(3) Set all remaining affinities to zero.
(4) Run the affinity propagation algorithm from [15] on the resulting adjacency matrix.

We also compare the quality measures for QECC and QECC-heur for a range of query budgets Q with those from the expected 3-approximation algorithm QwickCluster from [1]. While better approximation factors are possible (2.5 from [1], 2.06 from [12]), these algorithms require writing a linear program with $\Omega(n^3)$ constraints and all $\Omega(n^2)$ pairs of vertices need to be queried. By contrast, QwickCluster typically performs much fewer queries, making it more suitable for comparison.

Synthetic graphs. We construct a family of synthetic graphs $\mathcal{S} = \{S(n, k, \alpha, \beta)\}$, parameterized by the number of vertices n, number of clusters in the ground truth k, imbalance factor α, and noise rate β. The ground truth $T(n, k, \alpha)$ for $S(n, k, \alpha, \beta)$ consists of one clique of size $\alpha n/k$ and $k - 1$ cliques of size $(1 - \alpha)n/k$, all disjoint. To construct the input graph $S(n, k, \alpha, \beta)$, we flip the sign of every edge between same-cluster nodes in $T(n, k, \alpha)$ with probability β, and we flip the sign of every edge between distinct-cluster nodes with probability $\beta/(k - 1)$. (This ensures that the number total number of positive and negative edges flipped is roughly the same.)

Real-world graphs. For our experiments with real-world graphs, we choose three with very different characteristics:

- The cora dataset[1], where each node is a scientific publication represented by a string determined by its title, authors, venue, and date. Following [21], nodes are joined by a positive edge when the Jaro string similarity between them exceeds or equals 0.5.
- The Citeseer dataset[2], a record of publication citations for Alchemy. We put an edge between two publications if one of them cites the other [24].
- The Mushrooms dataset[3], including descriptions of mushrooms classified as either edible or poisonous, corresponding to the two ground-truth clusters. Each mushroom is described by a set of features. To construct the graph, we remove the edible/poisonous feature and place an edge between two mushrooms if they differ on at most half the remaining features. This construction has been inspired by [16], who show that high-quality clusterings can often be obtained by aggregating clusterings based on single features.

[1]https://github.com/sanjayss34/corr-clust-query-esa2019
[2]https://github.com/kimiyoung/planetoid
[3]https://archive.ics.uci.edu/ml/datasets/mushroom

Methodology. All the algorithms we test are randomized, hence we run each of them 50 times and compute the empirical average and standard deviations of the total cost, precision and recall values. We compute the average number A of queries made by QwickCluster and then run our algorithm with an allowance of queries ranging from $2n$ to A at regular intervals.

We use synthetic graphs to study how cost and recall vary in terms of (1) number of nodes n; (2) number of clusters k; (3) imbalance parameter α; (4) noise parameter β. For each plot, we fix all remaining parameters and vary one of them.

As the runtime for QECC scales linearly with the number of queries Q, which is an input parameter, we chose not to report detailed runtimes. We note that a simple Python implementation of our methods runs in under two seconds in all cases on an Intel i7 CPU at 3.7Ghz, and runs faster than the affinity propagation baseline we used (as implemented in Scikit-learn).

6.2 Experimental results

Table 1 summarizes the datasets we tested. Figure 1 shows the measured clustering cost against the number of queries Q performed by QECC and QECC-heur in the synthetic graph $S(2000, 20, 0.15, 2)$ and the real-world Cora, Citeseer and Mushrooms datasets.

Comparison with the baseline. It is clearly visible that both QECC-heur and QECC perform noticeably better than the baseline for all query budgets Q. As expected, all accuracy measures are improved with higher query budgets. The number of non-singleton clusters found by QECC-heur and QECC increases with higher values of Q, but decreases when using the affinity-propagation-based baseline. We do not show this value for the baseline on Mushrooms because it is of the order of hundreds; in this case the ground truth number of clusters is just two, and QwickCluster, QECC and QECC-heur need very few queries (compared to n) to find the clusters quickly.

QwickCluster vs QECC and QECC-heur. At the limit, where Q equals the average number A of queries made by QwickCluster, both QECC and QECC-heur perform as well as QwickCluster. In our synthetic dataset, the empirical average cost of QwickCluster is roughly 2.3 times the cost of the ground truth, suggesting that it is nearly a worst-case instance for our algorithm since QwickCluster has an expected 3-approximation guarantee. Remarkably, in the real-world dataset cora, QECC-heur can find a solution just as

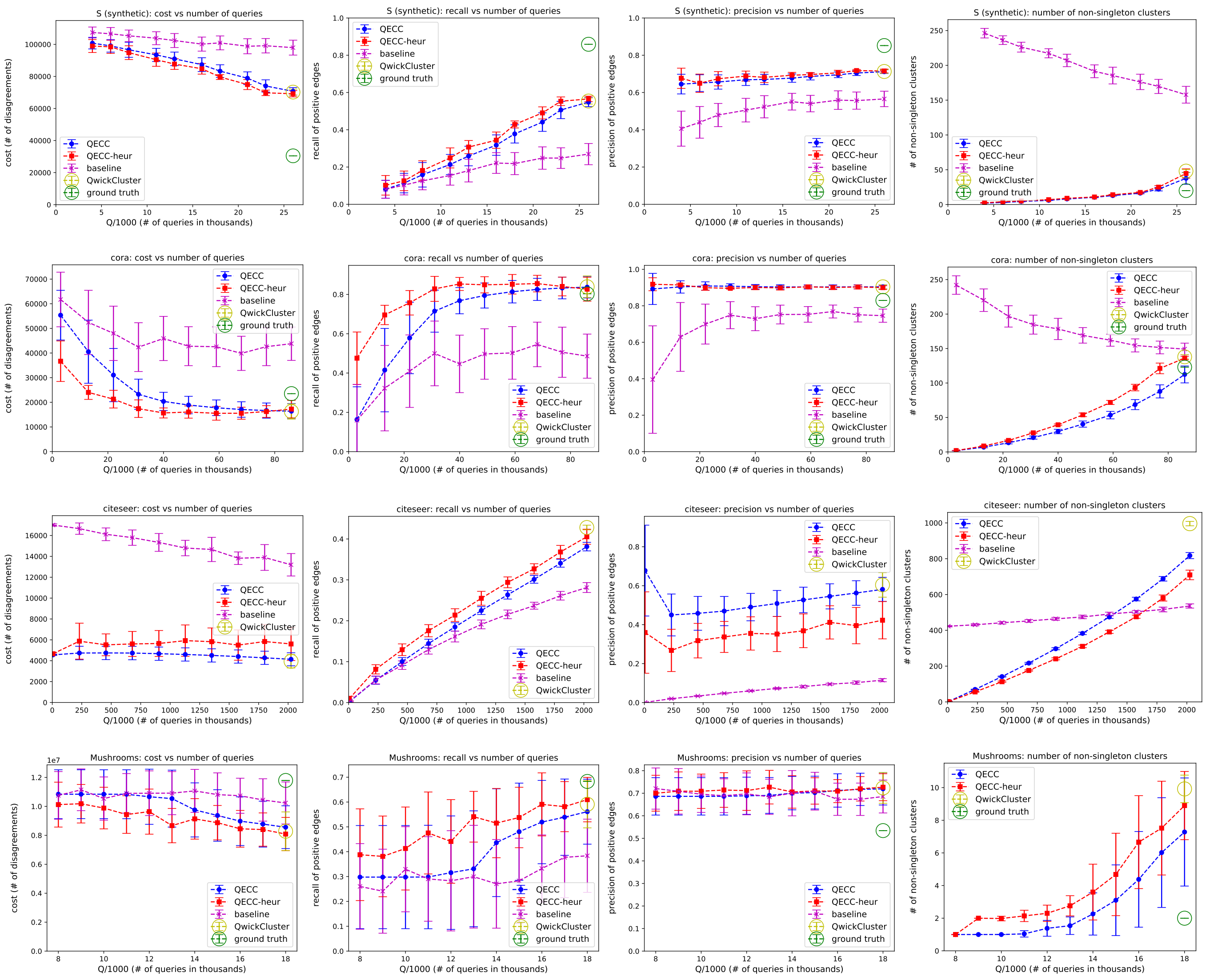

Figure 1: Accuracy measures of our two algorithms, the baseline, and ground truth on the datasets of Table 1.

good as the ground truth with just 40,000 queries. Notice that this is half what QwickCluster needs and much smaller than the $\binom{n}{2} \approx$ 1.7 million queries that full-information methods for correlation clustering such as [12] require.

Effect of graph characteristics. As expected, total cost and recall improve with the number of queries on all datasets (Figure 1); precision, however, remains mainly constant throughout a wide range of query budgets. To evaluate the impact of the graph and noise parameters on the performance of our algorithms, we perform additional tests on synthetic datasets where we fix all parameters to those of $S(2000, 20, 0.15, 2)$ except for the one under study. Figure 2 shows the effect of graph size n (1st row), number of clusters k, (2nd row), imbalance parameter α (3rd row) and noise parameter β (4th row) on total cost and recall, in synthetic datasets. Here we used $Q = 15000$ for all three query-bounded methods: QECC, QECC-HEUR and the baseline. Naturally, QwickCluster gives the best results as it has no query limit. All other methods tested follow the same trends, most of which are intuitive:

- Cost increases with n, and recall decreases, indicating that more queries are necessary in larger graphs to achieve the same quality. Precision, however stays constant.
- Cost decreases with k (because the graph has fewer positive edges). Recall stays constant for the ground truth and the unbounded-query method QwickCluster as it is essentially determined by the noise level, but it decreases with k for the query-bounded methods. Again, precision remains constant except for the baseline, where it decreases with k.
- Recall increases with imbalance α because the largest cluster, which is the easiest to find, accounts for a larger fraction of the total number of edges m. Precision also increases. On

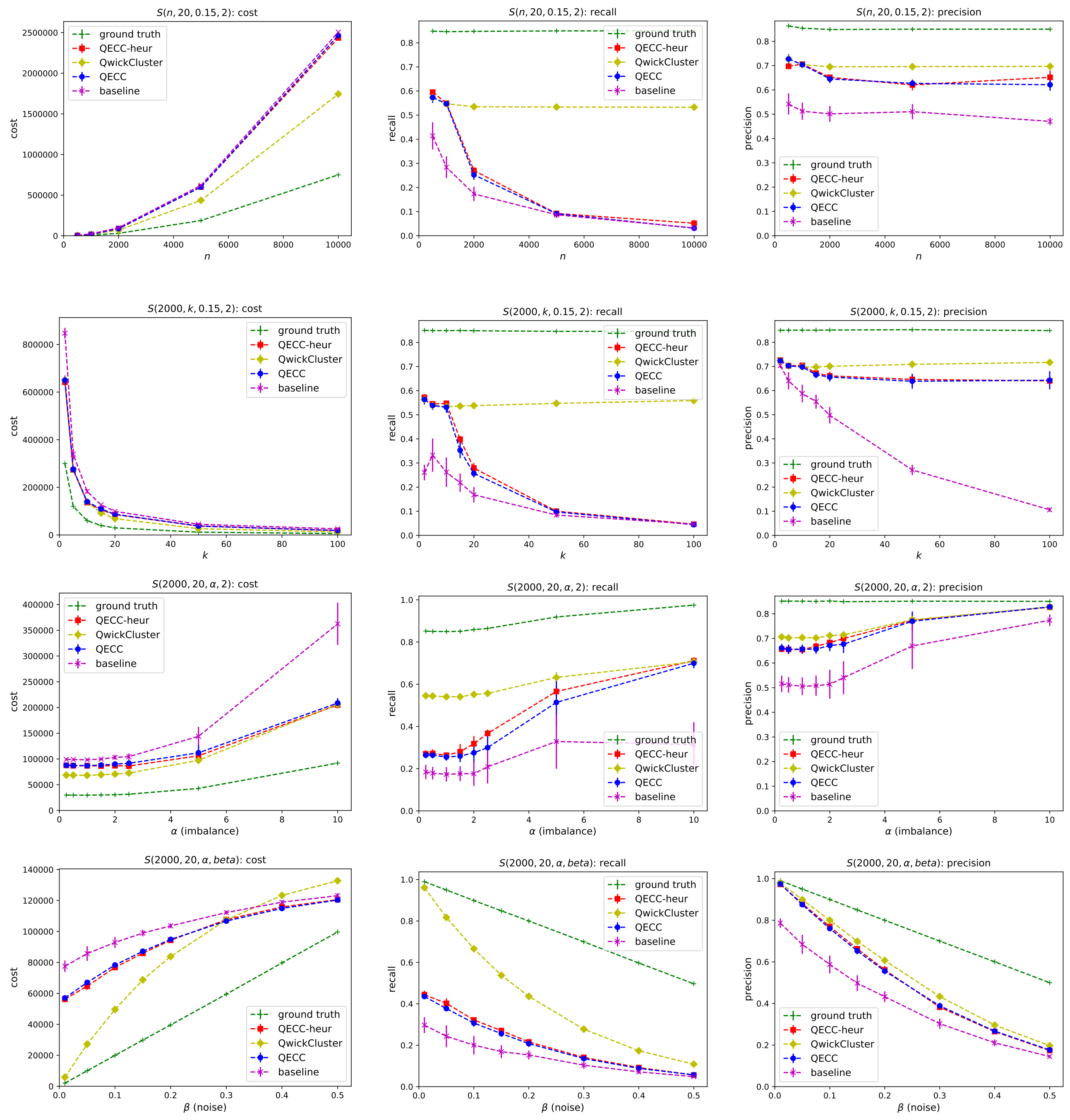

Figure 2: Effect of graph size n (1rst row), number of clusters k, (2nd row), imbalance parameter α (3rd row) and noise parameter β (4rth row) total cost and recall, for a fixed number $Q = 15000$ of queries, except for QwickCluster.

the other hand, m itself increases with imbalance, possibly explaining the increase in total cost.

- Finally, cost increases linearly with the level of noise β, while recall and precision decrease as β grows higher.

Effect of adaptivity. Finally, we compare the adaptive QECC with Non-adaptive QECC as described at the end of Section 3. Figure 3 compares the performance of both on the synthetic dataset and on Cora. While both have the same theoretical guarantees, it can be observed that the non-adaptive variant of QECC comes at moderate increase in cost and, decrease in recall and precision.

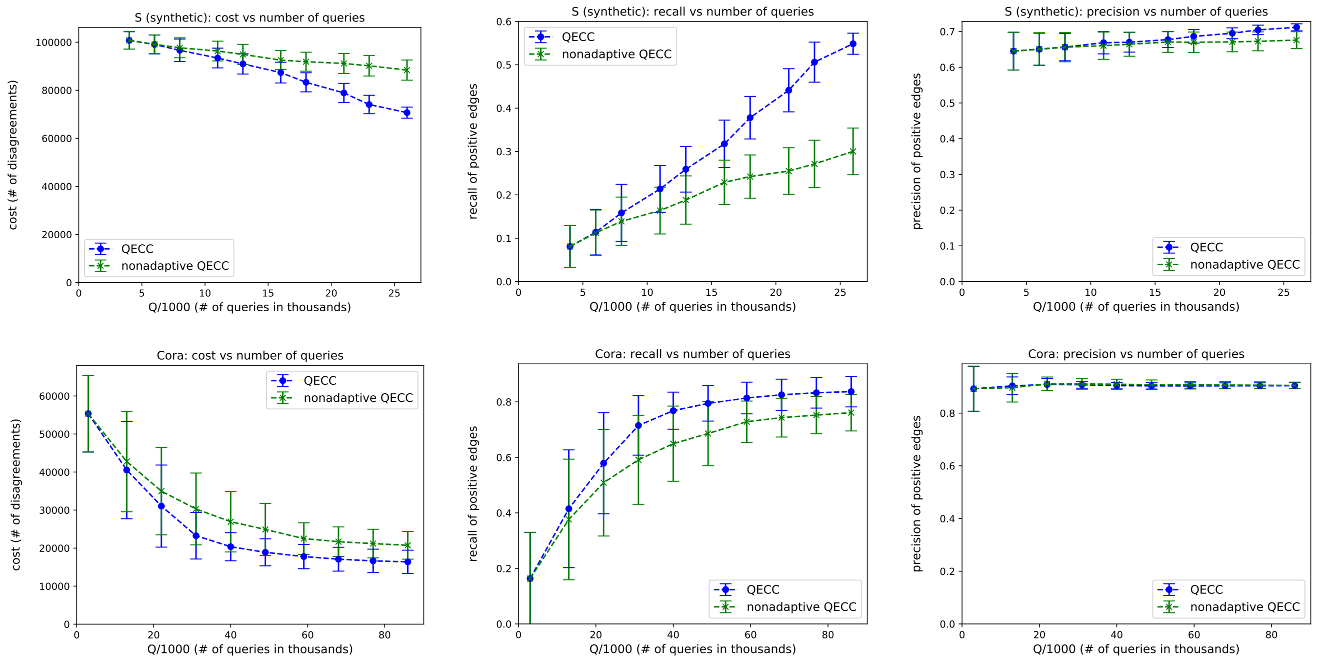

Figure 3: Comparison of QECC and its non-adaptive variant.

7 CONCLUSIONS

This paper presents the first query-efficient correlation clustering algorithm with provable guarantees. The trade-off between the running time of our algorithms and the quality of the solution found is nearly optimal. We also presented a more practical algorithm that consistently achieves higher recall values than our theoretical algorithm. Both of our algorithms are amenable to simple implementations.

A natural question for further research would be to obtain query-efficient algorithms based on the better LP-based approximation algorithms [12], improving the constant factors in our guarantee. Another intriguing question is whether one can devise other graph-querying models that allow for improved theoretical results while being reasonable from a practical viewpoint. The reason an additive term is needed in the error bounds is that, when the graph is very sparse, many queries are needed to distinguish it from an empty graph (i.e., finding a positive edge). We note that if we allow neighborhood oracles (i.e., given v, we can obtain a linked list of the *positive* neighbours of v in time linear in its length), then we can derive a constant-factor approximation algorithm with $O(n^{3/2})$ neighborhood queries, which can be significantly smaller than the number of edges. Indeed, Ailon and Liberty [2] argue that with a neighborhood oracle, QwickCluster runs in time $O(n + OPT)$; if $OPT \leq n^{3/2}$ this is $O(n^{3/2})$. On the other hand, if $OPT > n^{3/2}$ we can stop the algorithm after $r = \sqrt{n}$ rounds, and by Lemma 3.3, we incur an additional cost of only $O(n^{3/2}) = O(OPT)$. This shows that more powerful oracles allow for smaller query complexities. Our heuristic QECC-HEUR also suggests that granting the ability to query a random positive edge may help. These questions are particularly relevant to clustering graphs with many small clusters.

ACKNOWLEDGMENTS

Part of this work was done while CT was visiting ISI Foundation. DGS, FB, and CT acknowledge support from Intesa Sanpaolo Innovation Center. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

REFERENCES

[1] Nir Ailon, Moses Charikar, and Alantha Newman. 2008. Aggregating inconsistent information: Ranking and clustering. *J. ACM* 55, 5 (2008), 1–27.
[2] Nir Ailon and Edo Liberty. 2009. Correlation Clustering Revisited: The "True" Cost of Error Minimization Problems. In *Proc. of 36th ICALP*. 24–36.
[3] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation Clustering. *Machine Learning* 56, 1-3 (2004), 89–113.
[4] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. 1999. Clustering Gene Expression Patterns. *Journal of Computational Biology* 6, 3/4 (1999), 281–297.
[5] Francesco Bonchi, David García-Soriano, and Konstantin Kutzkov. 2013. *Local Correlation Clustering.* Technical Report. arXiv preprint arXiv:1312.5105.
[6] Francesco Bonchi, David García-Soriano, and Edo Liberty. 2014. Correlation clustering: from theory to practice. In *KDD*. 1972. http://videolectures.net/kdd2014_bonchi_garcia_soriano_liberty_clustering/
[7] Francesco Bonchi, Aristides Gionis, Francesco Gullo, Charalampos Tsourakakis, and Antti Ukkonen. 2015. Chromatic correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 9, 4 (2015), 34.
[8] Francesco Bonchi, Aristides Gionis, and Antti Ukkonen. 2013. Overlapping correlation clustering. *Knowl. Inf. Syst.* 35, 1 (2013), 1–32.
[9] Marco Bressan, Nicolò Cesa-Bianchi, Andrea Paudice, and Fabio Vitale. 2019. *Correlation Clustering with Adaptive Similarity Queries.* Technical Report. arXiv preprint arXiv:1905.11902.
[10] Joshua Brody, Kevin Matulef, and Chenggang Wu. 2011. Lower bounds for testing computability by small width OBDDs. In *International Conference on Theory and Applications of Models of Computation*. Springer, 320–331.

[11] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. 2005. Clustering with qualitative information. *J. Comput. System Sci.* 71, 3 (2005), 360–383.

[12] Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. 2015. Near optimal LP rounding algorithm for correlationclustering on complete and complete k-partite graphs. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. ACM, 219–228.

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

[14] Eldar Fischer. 2004. On the strength of comparisons in property testing. *Information and Computation* 189, 1 (2004), 107–116.

[15] Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *Science* 315, 5814 (2007), 972–976.

[16] Aristides Gionis, Heikki Mannila, and Panayiotis Tsaparas. 2007. Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data* 1, 1, Article 4 (March 2007).

[17] Oktie Hassanzadeh, Fei Chiang, Renée J. Miller, and Hyun Chul Lee. 2009. Framework for Evaluating Clustering Algorithms in Duplicate Detection. *PVLDB* 2, 1 (2009), 1282–1293.

[18] Sungwoong Kim, Sebastian Nowozin, Pushmeet Kohli, and Chang Dong Yoo. 2011. Higher-Order Correlation Clustering for Image Segmentation. In *NIPS*. 1530–1538.

[19] Andrew McCallum and Ben Wellner. 2005. Conditional models of identity uncertainty with application to noun coreference. In *Advances in neural information processing systems*. 905–912.

[20] Anirudh Ramachandran, Nick Feamster, and Santosh Vempala. 2007. Filtering spam with behavioral blacklisting. In *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 342–351.

[21] Barna Saha and Sanjay Subramanian. 2019. *Correlation Clustering with Same-Cluster Queries Bounded by Optimal Cost*. Technical Report. arXiv preprint arXiv:1908.04976.

[22] Ron Shamir, Roded Sharan, and Dekel Tsur. 2004. Cluster graph modification problems. *Discrete Applied Mathematics* 144, 1-2 (2004), 173–182.

[23] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. 2012. Crowder: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment* 5, 11 (2012), 1483–1494.

[24] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, Vol. 48. 40–48.

[25] Andrew Chi-Chin Yao. 1977. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (FOCS 1977)*. IEEE, 222–227.