Local correlation clustering

Francesco Bonchi David García–Soriano
Yahoo Labs
Barcelona, Spain
{bonchi, davidgs}@yahoo-inc.com

Konstantin Kutzkov
IT University of Copenhagen
Copenhagen, Denmark
konk@itu.dk

Abstract

Correlation clustering is perhaps 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.

Despite its theoretical appeal, the practical relevance of correlation clustering still remains largely unexplored. This is mainly due to the fact that correlation clustering requires the $\Theta(n^2)$ pairwise similarities as input. In large datasets this is infeasible to compute or even only to store.

In this paper we initiate the investigation into *local* algorithms for correlation clustering, laying the theoretical foundations for clustering "big data". In *local correlation clustering* we are given the identifier of a single object and we want to return the cluster to which it belongs in some globally consistent near-optimal clustering, using a small number of similarity queries.

Local algorithms for correlation clustering open the door to *sublinear-time* algorithms, which are particularly useful when the similarity between items is costly to compute, as it is often the case in many practical application domains. They also imply (*i*) distributed and streaming clustering algorithms, (*ii*) constant-time estimators and testers for cluster edit distance, and (*iii*) property-preserving parallel reconstruction algorithms for clusterability.

Specifically, we devise a local clustering algorithm attaining a $(3,\varepsilon)$-approximation (a solution with cost at most $3\cdot\mathsf{OPT}+\varepsilon n^2$, where OPT is the optimal cost). Its running time is $O(1/\varepsilon^2)$ independently of the dataset size. If desired, an explicit approximate clustering for all n objects can be produced in time $O(n/\varepsilon)$ (which is provably optimal). We also provide a fully additive $(1,\varepsilon)$-approximation with local query complexity $\mathsf{poly}(1/\varepsilon)$ and time complexity $2^{\mathsf{poly}(1/\varepsilon)}$. The explicit clustering can be found in time $n\cdot\mathsf{poly}(1/\varepsilon)+2^{\mathsf{poly}(1/\varepsilon)}$. The latter yields the fastest polynomial-time approximation scheme for correlation clustering known to date.

1 Introduction

In *correlation clustering*[1] we are given a set V of n objects and a pairwise similarity function $\mathrm{sim} : V \times V \to [0,1]$, and 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 $\mathrm{c}\ell$ is a function $\mathrm{c}\ell : V \to \mathbb{N}$. Correlation clustering aims at minimizing the cost:

$$\sum_{\substack{(x,y)\in V\times V,\\ \mathrm{c}\ell(x)=\mathrm{c}\ell(y)}} (1-\mathrm{sim}(x,y)) \; + \sum_{\substack{(x,y)\in V\times V,\\ \mathrm{c}\ell(x)\neq\mathrm{c}\ell(y)}} \mathrm{sim}(x,y). \tag{1}$$

The intuition underlying the above problem definition is that if two objects x and y are assigned to the same cluster we should pay the amount of their dissimilarity $(1-\mathrm{sim}(x,y))$, while if they are assigned to different clusters we should pay the amount of their similarity $\mathrm{sim}(x,y)$.

In the most widely studied setting, the similarity function is binary, i.e., $\mathrm{sim} : V \times V \to \{0,1\}$. This setting can be viewed very conveniently trough graph-theoretic lenses: the n items correspond to the vertices of a *similarity graph* G, which is a complete undirected graph with edges labelled "+" 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* [37] (or *clusterable* graph), 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 can be viewed as an agnostic learning problem, where we try to approximate the adjacency function of G by the hypothesis class of cluster graphs; alternatively, it is the task of finding the equivalence relation that most closely resembles a given symmetric relation R.

Correlation clustering provides a general framework in which one only needs to define a suitable similarity function. This makes it particularly appealing for the task of clustering structured objects, where the similarity function is domain-specific and does not rely on an ad hoc specification of some suitable metric such as the Euclidean distance of vectors. Thanks to this generality, the technique is applicable to a multitude of problems in different domains, including duplicate detection and similarity joins [17, 27], biology [11], image segmentation [30] and social networks [12].

Another key feature of correlation clustering is that it does not require a prefixed number of clusters, instead it automatically finds the optimal number.

Despite its appeal, correlation clustering has been, so far, mainly of theoretical interest. This is due to its scaling behavior with the size of the input data: given n items to be clustered, building the complete similarity graph G requires $\Theta(n^2)$ similarity computations. For a large n, the the similarity graph G might unfeasible to construct, or even only to store. This is the main bottleneck of correlation clustering and the reason why its practical relevance still remains largely unexplored.

The high-level contribution of our work is to overcome the main drawback of correlation clustering, making it scalable. We achieve this by designing algorithms that can construct a clustering in a *local* and distributed manner.

The input of a local clustering algorithm is the identifier of one of the n objects to be clustered, along with a short random seed. After making a small number of oracle similarity queries (probes

[1] Sometimes called *clustering with qualitative information* or *cluster editing*.

into the pairwise similarity matrix), a local algorithm outputs the cluster to which the object belongs, in some globally consistent near-optimal clustering.

1.1 A model for local correlation clustering

In the following we focus on the binary case: we will discuss the non-binary case together with other extensions in Section 7. We work with the adjacency matrix model, which assumes oracle access to the input graph G. Namely, given $x, y \in V(G)$, we can ask whether $\{x, y\}$ is a positive edge of G; each query is charged with unit cost. By *explicitly finding* a clustering we mean storing $\mathrm{c}\ell(v)$ for every $v \in V(G)$. In this explicit model a running time of $\Omega(n)$ is necessary as it requires to specify all values. An algorithm with complexity $O(n)$ for (approximate) correlation clustering is already a significant improvement over the complexity of most current solutions, but we take a step further and ask whether the dependence on n may be avoided altogether by producing *implicit* representations of the cluster mapping.

It is for this reason that we define local clustering as follows. Let us fix, for each finite graph G, a collection $\mathcal{C}^G$ of "high quality" clusterings for G.

Definition 1.1 (Local clustering algorithm) *Let $t \in \mathbb{N}$. A clustering algorithm $\mathcal{A}$ for $\mathcal{G}^n$ is said to be* local *with time (resp., query) complexity t if having oracle access to any graph G, and taking as input $|V(G)|$ and a vertex $v \in V(G)$, $\mathcal{A}$ returns a cluster label $\mathcal{A}^G(v)$ in time t (resp., with t queries).*

Algorithm $\mathcal{A}$ implicitly defines a clustering, described by the cluster label function $\mathrm{c}\ell(v) = \mathcal{A}^G(v)$, where the same sequence r of random bits is used by $\mathcal{A}$ to calculate $\mathcal{A}^G(v)$ for each v. The success probability of $\mathcal{A}$ is the infimum (over all graphs G) of the probability (over r) that the clustering implicitly defined by $\mathcal{A}^G$ belongs to $\mathcal{C}^G$.

Note that t does not depend on $n = |V(G)|$: this means that the cluster label of each vertex can be computed in constant time independently of the others. On the other hand, t could have a (hopefully mild) dependence on the desired *quality* of the clustering produced (which defines the set $\mathcal{C}^G$ for a given G), and the success probability of $\mathcal{A}$. Finally, it is important to note that, in order to define a unique "global" clustering across different vertices, the same sequence r of random coin flips must be used.

Sometimes we also allow *local algorithms with preprocessing p*, meaning (when p denotes time complexity) that $\mathcal{A}$ is allowed to perform computations and queries using total time p before reading the input vertex v. This preprocessing computation/query set is common to all vertices and may only depend on the outcome of $\mathcal{A}$'s internal coin tosses and the edges probed.

1.2 Contributions and practical implications

We focus on approximation algorithms for local correlation clustering with sublinear time and query complexity. Since any multiplicative approximation needs to make $\Omega(n^2)$ queries (Section 6), we need less stringent requirements.[2] One way is to allow an additional ε-fraction of edges to be violated, compared to the optimal clustering of cost OPT. Following Parnas and Ron [33], we study (c, ε) approximations: solutions with at most $c \cdot \mathsf{OPT} + \varepsilon \cdot n^2$ disagreements. These solutions form

[2]We remark that in a different model that uses *neighborhood oracles* [4], it is possible to bypass the $\Omega(n^2)$ lower bound for multiplicative approximations that holds for edge queries. In fact from our analysis we can derive the first sublinear-time constant-factor approximation algorithm for this case; see Section 7.

the set C^G of "high-quality" clusterings in Definition 1.1. Here c is a small constant and $\varepsilon \in (0,1)$ is an accuracy parameter specified by the user. Essentially ε handles the trade-off between the desired accuracy and the run-time: the larger ε the faster then algorithm, but also the further from OPT.

While we provide the formal statement of our results in Section 3, here we highlight the main message of this paper: there exist efficient local clustering algorithms with good approximation guarantees. Namely, in time $t = \mathsf{poly}(1/\varepsilon)$ it is possible to obtain $(O(1), \varepsilon)$-approximations locally. (Typically we think of ε as a user-defined constant.) This yields many practical contributions as by-products:

- **Explicit clustering in time $O(n)$.** Given that $\mathrm{c}\ell(v)$ can be computed in time t for each $v \in V(G)$, one can produce an explicit clustering in time $O(n \cdot t)$. Since $t = O(1)$, this is linear in the number of vertices (not edges) of the graph. More generally, the complexity of finding clusters of a subset $S \subseteq V$ of vertices requested by the user is proportional to the size of this subset.
- **Distributed algorithms.** We can assign vertices to different processors and compute their cluster labels in parallel, provided that the same random seed is passed along to all processors.
- **Streaming algorithms.** Similarly, local clustering algorithms can cluster graphs in the streaming setting, where edges arrive in arbitrary order. In this case the sublinear behaviour is lost because we still need to process every edge. However, the memory footprint of the algorithm can be brought down from $\Omega(n^2)$ to $O(n)$ (called the *semi-streaming* model [18]). Indeed, note that given a fixed random seed, for every vertex v the set of all possible queries Q_v that can be made during the computation of $\mathrm{c}\ell(v)$ has size[3] at most 2^t. This set can be computed before any edge arrives. From then on it suffices to keep in memory the edges (v, w) where $w \in Q_v$, and there are $n \cdot 2^t = O(n)$ of them. In fact, the running time of the local-based algorithm will be dominated by the time it takes to discard the unneeded edges.
- **Cluster edit distance estimators and testers.** We can estimate the degree of clusterability of the input data in constant time by sampling pairs of vertices and using the local clustering algorithm to see how many of them disagree with the input graph. We believe this can be an important primitive to develop new algorithms. Moreover, estimators for cluster edit distance give (tolerant) testers for the property of being clusterable, thereby allowing us to quickly detect data instances where any attempt to obtain a good clustering is bound to failure.
- **Local clustering reconstruction.** Queries of the form *"are x, y in the same cluster?"* can be answered in constant time without having to partition the whole graph: simply compute $\mathrm{c}\ell(x)$ and $\mathrm{c}\ell(y)$, and check for equality. This means that we can "correct" our input graph G (a "corrupted" version of a clusterable graph) so that the modified graph we output is close to the input and satisfies the property of being clusterable. This fits the paradigm of *local property-preserving data reconstruction* of [3] and [35].

To the best of our knowledge, this is the first work about local algorithms for correlation clustering.

[3] This bound can in fact be reduced to t for the non-adaptive algorithms we devise.

2 Background and related work

Correlation clustering. Minimizing disagreements is the same as maximizing agreements for exact algorithms, but the two tasks differ with regard to approximation. Following [25], we refer to these two problems as MaxAgree and MinDisagree, while MaxAgree[k] and MinDisagree[k] refer to the variants of the problem with a bound k on the number of clusters. Not surprisingly MaxAgree and MinDisagree are NP-complete [10,37]; the same holds for their bounded counterparts, provided that $k \geq 2$. Therefore approximate solutions are of interest. For MaxAgree, there is a (randomized) PTAS: the first such result was due to Bansal *et al.* [10] and ran in time $n^2 \exp(O(1/\varepsilon))$, later improved to $n \cdot 2^{\mathsf{poly}(1/\varepsilon)}$ by Giotis and Guruswami [25]. The latter also presented a PTAS for MaxAgree[k] that runs in time $n \cdot k^{O(\varepsilon^{-3}\log(k/\varepsilon))}$. In contrast, MinDisagree is APX-hard [14], so we do not expect a PTAS. Nevertheless, there are constant-factor approximation algorithms [2,10,14]. The best factor (2.5) was given by Ailon *et al.* [2], who also present a simple, elegant algorithm that achieves a slightly weaker expected approximation ratio of 3, called QuickCluster (see Section 4). For MinDisagree[k], PTAS appeared in [25] and [29]. There is also work on correlation clustering on incomplete graphs [10,14,17,25,41].

Sublinear clustering algorithms. Sublinear clustering algorithms for geometric data sets are known [5,9,15,16,32]. Many of these find implicit representations of the clustering they output. There is a natural implicit representation for most of this problems, e.g., the set of k cluster centers. By contrast, in correlation clustering there may be no clear way to define a clustering for the whole graph based on a small set of vertices. The only sublinear-time algorithm known for correlation clustering is the aforementioned result of [25]; it runs in time $O(n)$, but the multiplicative constant hidden in the notation has an exponential dependence on the approximation parameter.

The literature on *active clustering* also contains algorithms with sublinear query complexity (see, e.g., [28]); many of them are heuristic or do not apply to correlation clustering. Ailon *et al.* [1] obtain algorithms for MinDisagree[k] with sublinear query complexity, but the running time of their solutions is exponential in n.

Local algorithms. The following notion of locality is used in the distributed computing literature. Each vertex of a sparse graph is assigned a processor, and each processor can compute a certain function in a constant number of rounds by passing messages to its neighbours (see Suomela's survey [40]). Our algorithms are also local in this sense.

Recently, Rubinfeld *et al.* [34] introduced a model that encompasses notions from several algorithmic subfields, such as locally decodable codes, local reconstruction and local distributed computation. Our definition fits into their framework: it corresponds to *query-oblivious, parallelizable, strongly local* algorithms that compute a cluster label function in constant time.

Finally, we point out the work of Spielman and Teng [39] pertaining local clustering algorithms. In their papers an algorithm is "local" if it can, given a vertex v, output v's cluster in time nearly linear in the cluster's size. Our local clustering algorithms also have this ability (assuming, as they do, that for each vertex we are given a list of its neighbours), although the results are not comparable because [39] attempt to minimize the cluster's conductance.

Testing and estimating clusterability. Our methods can also be used for quickly testing clusterability of a given input graph G, which is related to the task of estimating the *cluster edit distance*, i.e., the minimum number of edge label swaps (from "+" to "–" and viceversa) needed to transform G into a cluster graph. Note that this corresponds to the optimal cost of correlation clustering for the given input G. Clusterability is a hereditary graph property (closed under removal

and renaming of vertices), hence it can be tested with one-sided error using a constant number of queries by the powerful result of Alon and Shapira [8]. Combined with the work of Fischer and Newman [20], this also yields estimators for cluster edit distance that run in time independent of the graph size. Unfortunately, the query complexity of the algorithm given by these results would be a tower exponential of height $\mathsf{poly}(1/\varepsilon)$, where ε is the approximation parameter.

Approximation algorithms for MIN-2-CSP problems [7] also give estimators for cluster edit distance. However, they provide no way of computing each variable assignment in constant time. Moreover, they use time $\sim n^2$ to calculate all assignments, and hence do not lend themselves to sublinear-time clustering algorithms.

3 Statement of results

All our graphs are undirected and simple. For a vertex v, $\Gamma^+(v)$ is the set of positive edges incident with v; similarly define $\Gamma^-(v)$. We extend this notation to sets of vertices in the obvious manner. The *distance* between two graphs $G = (V, E)$ and $G' = (V, E')$ is $|E \oplus E'|$. Their fractional distance is their distance divided by n^2 (note this is in the interval $[0, 1/2)$). Two graphs are *ε-close* to each other if their distance is at most most εn^2. A *k-clusterable* graph is a union of at most k vertex-disjoint cliques. A graph is *clusterable* if it is k-clusterable for some k.

The following folklore lemma says that approximate k-clustering algorithms yield approximate clustering algorithms with an unbounded number of clusters:

Lemma 3.1 *If G is clusterable, then it is ε-close to $(1 + 1/\varepsilon)$-clusterable.*

Proof. Take the optimal clustering for G. Let B be the set of vertices in clusters of size $< \varepsilon n$. Now re-cluster the elements of B arbitrarily into clusters of size $\lfloor \varepsilon n \rfloor$ (except possibly one). This introduces at most $\varepsilon n \cdot |B| \leq \varepsilon n^2$ additional errors. All but one of the clusters of the resulting clustering have size $\geq \lfloor \varepsilon n \rfloor$, hence it has at most $1 + 1/\varepsilon$ clusters. □

Corollary 3.2 *Any $(1, \varepsilon/2)$ approximation to the optimal $(1 + 2/\varepsilon)$-clustering is also a $(1, \varepsilon)$ approximation to the optimal clustering.*

Proof. Immediate from the triangle inequality for graph distances. □

We are now ready to summarize our results. All our algorithms are (necessarily) randomized, and succeed with probability no less than 2/3 (which can be amplified). Our first result concerns the standard setting where the clusters of all vertices need to be explicitly computed. We present a $(4, \varepsilon)$-approximation[4] that runs in time $O(n/\varepsilon)$; compare the $\Omega(n^2)$ complexity of most other clustering methods. Our algorithm is optimal up to constant factors.

Theorem 3.3 *Given $\varepsilon \in (0, 1)$, a $(4, \varepsilon)$-approximate clustering for* MinDisagree *can be found in time $O(n/\varepsilon)$. Moreover, finding an $(O(1), \varepsilon)$-approximation with constant success probability requires $\Omega(n/\varepsilon)$ queries.*

[4]We can also produce an *expected* $(3, \varepsilon)$-approximation. Because we insist on algorithms that work with constant success probability, we talk about $(4, \varepsilon)$-approximations, where the constant 4 could be replaced with any number ≥ 3.

In other words, with a "budget" of q queries we can obtain a $(4, O(n/q))$-approximation. In fact, the upper bound of Theorem 3.3 can be derived from our next result. It states that the same approximation can be *implicitly* constructed in constant time, regardless of the size of the graph.

Theorem 3.4 *Given $\varepsilon \in (0,1)$, a $(4,\varepsilon)$-approximate clustering for* MinDisagree *can be found locally in time $O(1/\varepsilon^2)$, or in time $O(1/\varepsilon)$ after preprocessing that uses $O(1/\varepsilon^2)$ non-adaptive queries and time. Moreover, finding an $(O(1),\varepsilon)$-approximation with constant success probability requires $\Omega(1/\varepsilon)$ adaptive queries.*

As a corollary we obtain a partially tolerant tester of clusterability. We stress that the tester is efficient both in terms of query complexity and time complexity, unlike many results in property testing.

Corollary 3.5 *There is a non-adaptive, two-sided error tester which accepts graphs that are $\varepsilon/5$-close to clusterable and rejects graphs that are ε-far from clusterable. It runs in time $O(1/\varepsilon^2)$.*

So far these results do not allow us to obtain clusterings that are arbitrarily close to the optimal one. To overcome this issue, we also show (using different techniques) that a purely additive approximation can still be found with $\mathsf{poly}(1/\varepsilon)$ queries, but with an exponentially larger running time.

Theorem 3.6 *Given $\varepsilon \in (0,1)$, there is a local clustering algorithm that achieves an $(1,\varepsilon)$ approximation to the cost of the optimal clustering. Its local time complexity is $\mathsf{poly}(1/\varepsilon)$ after preprocessing that uses $\mathsf{poly}(1/\varepsilon)$ queries and $2^{\mathsf{poly}(1/\varepsilon)}$ time.*

For the explicit versions we obtain the following.

Corollary 3.7 *There is a $(1,\varepsilon)$-approximate clustering algorithm for* MinDisagree *(and hence* MaxAgree *too) that runs in time $n \cdot \mathsf{poly}(1/\varepsilon) + 2^{\mathsf{poly}(1/\varepsilon)}$. In particular there is a* PTAS *for* MaxAgree *with the same running time.*

The "in particular" part follows from the observation that the optimum value for MaxAgree[2] is $\Omega(n^2)$ (see, e.g., [25, Theorem 3.1]). The best PTAS in the literature [25] ran in time $n \cdot 2^{\Omega(\varepsilon^{-3}\log^2(1/\varepsilon))}$. In our result, the dominating term (depending on n) has an exponentially smaller multiplicative constant (polynomial in $1/\varepsilon$), and then we have an additive term exponential in $1/\varepsilon$ (and independent of n). As for lower bounds, observe that the $\Omega(n/\varepsilon)$ bound from Theorem 3.3 still applies, while the presence of a term of the form $2^{(1/\varepsilon)^{\Omega(1)}}$ for very small ε seems hard to avoid due to the NP-completeness of the problems, as an optimal solution can be found upon setting $\varepsilon = 1/n^2$.

These results are established via the study of the corresponding problems with a prespecified number k of clusters; such algorithms yield additive approximations to the general case upon setting $k = O(1/\varepsilon)$ in view of Lemma 3.1. For fixed k, the bounds for our algorithms have the same form after replacing ε by k/ε (see Section 3.6). For example, we get a PTAS for MaxAgree$[k]$ in time $n \cdot \mathsf{poly}(k/\varepsilon) + 2^{\mathsf{poly}(k/\varepsilon)}$.

Corollary 3.8 *For any $0 < \varepsilon_1 < \varepsilon_2$, there is a non-adaptive, one-sided error tester which accepts graphs that are ε_1-close to clusterable and rejects graphs that are ε_2-far from clusterable. It has query complexity $\mathsf{poly}(1/\varepsilon)$ and runs in time $2^{\mathsf{poly}(1/\varepsilon)}$, where $\varepsilon = \varepsilon_2 - \varepsilon_1$.*

Techniques and roadmap. Our first local algorithm (Theorem 3.4) is inspired by the QuickCluster algorithm of Ailon *et al.* [2], which resembles the greedy procedure for finding maximal independent sets. The main idea to make a local version is to define the clusters "in reverse". We find a small set P of "cluster centers" or "pivots" by looking at a small induced subgraph, and then we show a simple rule to define an extended clustering for the whole graph in terms of the adjacencies of each particular vertex with P. As it turns out, such P can be obtained by a procedure that finds a constant-sized "almost-dominating" set of vertices that are within distance two of most other vertices in the graph, in such a way that we can combine the expected 3-approximation guarantee of [2] with an additive error term. The algorithm and its analysis are given in Section 4.

The second local algorithm (Theorem 3.6) borrows ideas from the PTAS for dense MaxCut of Frieze and Kannan [24] and uses low-rank approximations to the adjacency matrix of the graph. (Interestingly, while such approximations have been known for a long time, their implications for correlation clustering have been overlooked.) Notably, implicit descriptions of these approximations are locally computable in constant time (polynomial in the inverse of the approximation parameter). We show that in order to look for near-optimal clusterings, we can restrict the search to clusterings that "respect" a sufficiently fine weakly regular partition of the graph. Then we argue that this can be used to implicitly define a good approximate clustering: to cluster a given vertex, we first determine its piece in a regular partition, and then we look at which cluster contains this piece in the best coarsening of the partition. The details are in Section 5.

The lower bounds, proven in Section 6, are applications of Yao's lemma [42]. Broadly speaking, we give the candidate algorithm a perfect clustering of most vertices of the graph into $t = O(1/\varepsilon)$ clusters of equal size, and for each of the remaining vertices a "secret" cluster is chosen at random among these t. The optimal clustering of the resulting graph has fractional cost ε/c for some constant $c > 1$. We then ask the algorithm to find clusters for the remaining vertices, and show that it must make $\Omega(n/\varepsilon)$ adaptive queries if it is to output a clustering with fractional cost no larger than ε.

Finally, in Section 7 we discuss several extensions, including the case of non-binary similarity measure.

4 $(3, \varepsilon)$-approximations

First we describe the QuickCluster algorithm of Ailon et al. [2]. It selects a random pivot, creates a cluster with it and its positive neighborhood, removes the cluster, and iterates on the induced subgraph remaining. Essentially it finds a maximal independent set in the positive graph. When the graph is clusterable, it makes no errors. In [2], the authors show that the expected cost of the clustering found is at most three times the optimum.

Note that determining the positive neighborhood $\Gamma^+(v)$ of a pivot v takes $n - 1$ queries to the adjacency oracle. The algorithm's worst-case complexity is $\Theta(n^2)$: consider the graph with no positive edges. In fact its time and query complexity is $O(nc)$, where c is the average number of clusters found. This suggests attempting to partition the data into a small number of clusters to minimize query complexity.

We know from Lemma 3.1 that any clustering can be ε-approximated by a clustering with pieces of size $\Omega(\varepsilon n)$. So an idea would be to modify QuickCluster so that most clusters output are sufficiently large. Fortunately, QuickCluster tends to do just that on average, provided that the

Algorithm 1 LocalCluster

```
function LocalCluster(v, ε)
    P ← FindGoodPivots(ε)          ▷ This is the preprocessing stage and can be taken outside
    return FindCluster(v, P)

function FindCluster(v, P)
    if v ∉ Γ⁺(P) then return v                          ▷ Cluster v by itself
    else i ← min{j | v ∈ Γ⁺(P_j)}; return P_i           ▷ Find first positive neighbour in P

function FindGoodPivots(ε)
    for i ∈ [16] do
        P^i ← FindPivots(ε/12);
        d̃^i ← estimate of the cost of P^i with O(1/ε) local clustering calls (see Appendix 4)
    j ← arg min{d̃^i | i ∈ [16]}
    return P^j

function FindPivots(ε)
    Let Q ⊆ V be a random sample without replacement of size min(n, 1/(2ε))
    return IndependentSet(Q)

function IndependentSet(Q)
    P ← [] (empty sequence)
    for v ∈ Q do
        if FindCluster(v, P) = v then append v to P
    return P
```

graph of positive edges is sufficiently dense, because the expected size of the next cluster found is precisely one plus the average degree of the remaining graph. Once the graph becomes too sparse, a low-cost clustering of the remaining vertices can be found without even looking at the edges, for example by putting each of them into their own cluster.[5]

Another advantage of finding a small number of clusters is locality. Let $P = P_1, \ldots, P_t$ denote the first t elements of the sequence of pivots found by QuickCluster. Let us pick an arbitrary vertex v contained in the neighbourhood of $\{P_1, \ldots, P_t\}$; all other vertices can be safely ignored because as we shall see they usually will be incident to few edges (for suitably chosen t). Then the pivot of v's cluster is the *first* element of P that is a positive neighbour of v: therefore it can be determined in time $O(t)$, assuming we are given the pivot sequence $P_1, \ldots, P_t$.

Therefore we propose the scheme whose pseudocode is given in Algorithm 1 (the analysis is presented in the next section). Assuming we know a good sequence P, an implicit clustering is defined deterministically in the way described above; two vertices v and v' belong to the same cluster if and only if FindCluster(v) = FindCluster(v'). Similarly to QuickCluster, we can find a set of pivots by finding an independent set of vertices in the graph; to keep it small we restrict the search to an induced subgraph of size $O(1/\varepsilon)$. This is done by FindPivots, which can be seen as a "preprocessing stage" for the local clustering algorithm FindCluster. In the next section the following key lemma will be shown.

Lemma 4.1 *Let $\varepsilon \in (0,1)$ and $r, s > 1$. The expected cost of the clustering determined by* FindPivots(ε) *is at most $3 \cdot \mathsf{OPT} + \varepsilon n^2$, and the probability that it exceeds $3r \cdot \mathsf{OPT} + s \cdot \varepsilon n^2$ is less than $\frac{1}{r} + \frac{1}{s}$.*

[5] Another possibility that works is to cluster all remaining vertices into clusters of size εn, eliminating the need for singleton clusters.

For example, setting $r = 4/3$, $s = 8$ we see that with probability $1/8$, the clustering determined by FindPivots($\varepsilon/4$) is a $(4, \varepsilon)$-approximation to the optimal one. Although this low bound on the success probability may be overly pessimistic, we can amplify it in order to obtain better theoretical guarantees. To do this with confidence $2/3$ we try several samples Q and estimate the cost of the associated local clusterings by sampling random edges.

Lemma 4.2 *Let d denote the fractional cost of the optimal clustering. With probability at least $5/6$,* FindGoodPivots(ε) *returns a pivot set P^i with fractional cost at most $4d + \varepsilon$. Its running time is $O(1/\varepsilon^2)$.*

Finally, to obtain a purely local clustering algorithm with no preprocessing we need each vertex to find the good set of pivots by itself, as per LocalCluster(v, ε). Note it is crucial here that all vertices have access to the same source of randomness so the same set of pivots is found on each separate call. In practical implementations this means introducing an additional parameter of short length, for instance a common random seed to be used.

From Lemmas 4.1 and 4.2 we can easily deduce two of our main results.

Corollary 4.3 (Upper bound of Theorem 3.4) LocalCluster(v, ε) *is a local clustering algorithm for* MinDisagree *achieving a $(4, \varepsilon)$ approximation to the optimal clustering with probability $2/3$. The preprocessing runs in time $O(\min(n/\varepsilon, 1/\varepsilon^2))$, and the clustering time per vertex is $O(1/\varepsilon)$.*

Corollary 4.4 (Upper bound of Theorem 3.3) *An explicit clustering attaining a $(4, \varepsilon)$ approximation can be found with probability $2/3$ in time $O(n/\varepsilon)$.*

By a very similar argument we can produce an *expected* $(3, \varepsilon)$-approximate clustering in time $O(n/\varepsilon)$. A time-efficient property tester for clusterability (Corollary 3.5) is also a simple consequence of the above.

4.1 Analysis of the local algorithm

We prove the approximation guarantees of the algorithm (Lemma 4.1) by comparing it to the clustering found by QuickCluster, which is known to achieve an expected 3-approximation [10]. In this section we consider the input graph $G = G^+$ with negative edges removed, so $\Gamma(v) = \Gamma^+(v)$ for all $v \in G$.

The following is a straightforward consequence of the multiplicative Chernoff bounds:

Lemma 4.5 *Let $c > 1$, $\varepsilon, \delta \in (0, 1)$, $m \in \mathbb{N}^+$ and $n = \frac{3 \cdot c}{(c-1)^2} \frac{\log(m/\delta)}{\varepsilon}$. Suppose $\{X_j^i \mid i \in [m], j \in [n]\}$ are random variables in $[0, 1]$ such that for all $i \in [m]$, the variables $X_1^i, \ldots, X_n^i$ are independent with common mean μ_i. Define $\tilde{\mu}_i = \frac{1}{n} \sum_{j=1}^{n} X_i^j$. Then with probability at least $1 - \delta$, the following holds:*

- *If $\min_{i \in [m]} \mu_i \leq \frac{1}{c} \cdot \varepsilon$, then $\min_{i \in [m]} \tilde{\mu}_i \leq \varepsilon$.*
- *For all $i \in [m]$, if $\mu_i > c \cdot \varepsilon$, then $\tilde{\mu}_i > \varepsilon$.*

Corollary 4.6 *Let $d, \delta \in (0, 1), c > 1$. Let $C_1, \ldots, C_m$ be m clusterings such that at least one of them has fractional cost $\leq d$. Then with probability $1 - \delta$ we can select $i \in [m]$ such that C_i has fractional cost at most $c \cdot d$ using a total of $\frac{3c}{(c-1)^2} \frac{m \log(m/\delta)}{\varepsilon}$ edge queries to $C_1, \ldots, C_m$.*

Proof of Lemma 4.2. Use Lemma 4.1 and Corollary 4.6. □

Proof of Corollary 4.4. Call FindGoodPivots(ε) once to obtain a good pivot sequence P with probability 2/3 in time $O(\min(n, \varepsilon^{-2})) \leq O(n)$. Then run FindCluster$(v, P)$ sequentially for each vertex v in order to determine its cluster label l, appending v to the list of vertices in cluster labelled l. Finally output the resulting clusters. The whole process runs in time $O(n) + O(n/\varepsilon) = O(n/\varepsilon)$. □

A *partial clustering* of V is a clustering of a subset W of V. Its *partial cost* is the number of disagreements between edges that have at least one endpoint in W.

Now consider a clustering $\mathcal{C}$ of V into $C_1, \ldots, C_m \subseteq V$. For $S \subseteq [m]$, the Sth *partial subclustering* of $\mathcal{C}$ is the partition of $V_S = \bigcup_{i \in S} C_i$ into $\{C_i\}_{i \in S}$. Clearly the cost of a clustering upper bounds the partial cost of any of its partial subclusterings.

Lemma 4.7 *Let $P_1, \ldots, P_t$ denote the sequence of pivots found by* FindPivots(ε). *The expected number of edge violations involving vertices within distance ≤ 2 from $P_1, \ldots, P_t$ is at most $3 \cdot \mathsf{OPT}$.*

Proof. To simplify the analysis, in the proof of this lemma we modify QuickCluster and FindPivots slightly so that they run deterministically provided that a random permutation π of the vertex set V is chosen in advance. Concretely, we consider a deterministic version of QuickCluster, denoted QuickCluster$^\pi$, that uses pivot set IndependentSet(Q), where Q lists all vertices of V in ascending order of π. Similarly, deterministic FindCluster$^\pi$ takes for Q the set of the *first* $O(1/\varepsilon)$ elements in increasing order of π. Clearly running FindCluster$^\pi$ on a random permutation π is the same as running the original FindCluster, and likewise for QuickCluster.

Observe that the set $P_1, \ldots, P_{t(\pi)}$ of pivots returned by FindCluster$^\pi$ is a prefix of the set of pivots returned by QuickCluster$^\pi$. Therefore the first $t(\pi)$ clusters are the same as well, i.e., $P_1, \ldots, P_{t(\pi)}$ define a partial subclustering of the one found by QuickCluster$^\pi$. Hence the partial cost of the subclustering determined by FindCluster$^\pi$ is in expectation at most $3 \cdot \mathsf{OPT}$. This is equivalent to the statement of the lemma. □

Next we show that FindCluster returns a small "almost-dominating" set of vertices, in the sense quantified in the following result.

Theorem 4.8 *Let $G = (V, E)$ be a graph and Q be an ordered sample of r independent vertices uniformly chosen with replacement from V. Let $P =$* IndependentSet(Q). *Then the expected number of edges of G not incident with an element of $P \cup \Gamma(P)$ is less than $\frac{n^2}{2r}$.*

Observe that an existential result for an almost-dominating set P is easy to establish by picking pivots in order of decreasing degree in the residual graph. However, doing so would invalidate the approximation guarantees of QuickCluster we are relying on. We defer the proof of Theorem 4.8. Assuming the result, we are ready to prove Lemma 4.1.

Proof of Lemma 4.1. Lemma 4.7 says that the set of pivots found define a partial clustering with expected cost bounded by $3 \cdot \mathsf{OPT}$. Let Q be the random sample used by FindPivots(ε). Theorem 4.8 is stated for sampling with replacement, but this implies the same result for sampling without replacement, so its conclusion still holds. Combining the two results and setting $r = 1/(2\varepsilon)$,

we see that the set of disagreements in the clustering produced can be written as the union of two disjoint random sets $A, B \in V \times V$ with $\mathbb{E}[A] \leq 3 \cdot \mathsf{OPT}$ and $\mathbb{E}[B] \leq \varepsilon n^2$. Thus the total cost is $|A \cup B| = |A| + |B|$. By linearity of expectation, the expected cost is $\mathbb{E}[|A| + |B|] \leq 3 \cdot \mathsf{OPT} + \varepsilon n^2$, and by applying Markov's inequality to the non-negative variables $|A|$ and $|B|$ separately we conclude that

$$\Pr[(|A| > r\,\mathbb{E}[|A|]) \text{ or } (|B| > s\,\mathbb{E}[|B|])] < \frac{1}{r} + \frac{1}{s}.$$

□

The rest of this section is devoted to prove Theorem 4.8. 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 $\{v\} \cup \Gamma(v)$ (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 at random from $V(G_0)$ (note that sometimes $G_{i+1} = G_i$).

Lemma 4.9 *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 $\binom{\tilde{d}+1}{2}$, and the number of degree-0 vertices increases in expectation by at least $\tilde{d} + 1$.*

Proof. Let $V = V(G_0)$, $E = E^+(G_i)$ and let $d_u = |\Gamma(u)|$ denote the positive degree of $u \in V$ on G_i. The claim on the number of degree-0 vertices is easy, so we prove the claim on the number of edges. Consider an edge $\{u, v\} \in E$. It is deleted if the chosen pivot is an element of $\Gamma(u) \cup \Gamma(v)$ (this contains u and v); let X_{uv} be the 0-1 random variable associated with this event. It 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 $D = \sum_{u<v|\{u,v\}\in E} X_{uv}$ be the number of edges deleted. By linearity of expectation, its average is

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

Now we compute

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

hence $\mathbb{E}[D] \geq \frac{\tilde{d}}{2} + \frac{\tilde{d}^2}{2} = \binom{\tilde{d}+1}{2}$. □

Now let $\widetilde{V}(G) = \{v \in V(G) \mid \deg(v) > 0\}$ and define the "actual size" $S(G)$ of a graph by $S(G) = 2 \cdot |E(G)| + |\widetilde{V}(G)| = \sum_{v \in \widetilde{V}(G)} (1 + \deg(v))$. Let $n = |V(G_0)|$ and define $\alpha_i \in [0, 1]$ by $\alpha_i = \frac{s(G_i)}{n^2}$.

Lemma 4.10 *For all $i \geq 1$ the following inequalities hold:*

$$\mathbb{E}[\alpha_i \mid v_1, \ldots, v_{i-1}] \leq \alpha_{i-1}(1-\alpha_{i-1}), \tag{2}$$

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

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

Proof. Inequality (2) is a restatement of Lemma 4.9. Inequality (3) follows from Jensen's inequality: since $\mathbb{E}[\alpha_i] = \mathbb{E}\left[\mathbb{E}[\alpha_i \mid v_1, \ldots, v_{i-1}]\right] \leq \mathbb{E}[\alpha_{i-1}(1-\alpha_{i-1})]$ and the function g mapping x to $g(x) = x(1-x)$ is concave, we have $\mathbb{E}[\alpha_i] \leq \mathbb{E}[g(\alpha_{i-1})] \leq g(\mathbb{E}[\alpha_{i-1}]) = \mathbb{E}[\alpha_{i-1}](1-\mathbb{E}[\alpha_{i-1}])$.

Finally we prove $\mathbb{E}[\alpha_i] < 1/(i+1)$ for all $i \geq 1$. We know that $\mathbb{E}[\alpha_1] \leq \max_{x\in[0,1]} g(x) = g\left(\frac{1}{2}\right) = \frac{1}{4}$, so the claim follows by induction on i as g is increasing on $[0, 1/2]$ and $g\left(\frac{1}{i}\right) = \frac{1}{i} - \frac{1}{i^2} \geq \frac{1}{i} - \frac{1}{i(i+1)} = \frac{1}{i+1}$. □

Remark 4.1 *With a finer analysis (Appendix A), Equation* (4) *can be strengthened to* $\mathbb{E}[\alpha_i] \leq \frac{1}{\frac{1}{\widehat{a}}+i+\Omega(\ln(\widehat{a}\cdot i))}$, *where* $\widehat{a} = \min(\alpha_0, 1-\alpha_0)$. *(This does not affect the asymptotics.)*

Proof of Theorem 4.8. Note that after sampling r vertices $v_1, \ldots, v_r$ with replacement from G_0, the subgraph of G_0 resulting from removing all edges incident to $\bigcup_{i=1}^{r}\{v_i\} \cup \Gamma(v_i)$ is distributed according to G_r. Using Equation (4), we bound $\mathbb{E}[|E(G_r)|] \leq \frac{n^2}{2}\mathbb{E}[\alpha_r] < \frac{n^2}{2r}$. □

5 Fully additive approximations

Here we study $(1, \varepsilon)$-approximations. By Lemma 3.1 and its corollary, it is enough to consider k-clusterings for $k = O(1/\varepsilon)$.

5.1 The regularity lemma.

One of the cornerstone results in graph theory is the regularity lemma of Szemerédi, which has found a myriad applications in combinatorics, number theory and theoretical computer science [31]. It asserts that every graph G can be approximated by a small collection of random bipartite graphs; in fact from G we can construct a small "reduced" weighted graph $\widetilde{G}$ of constant size which inherits many properties of G. If we select an approximation parameter ε, it gives us an equitable partition of the vertex set V of G into a constant number $m = m(\varepsilon)$ of classes $C_1, \ldots, C_m$ such that the following holds: for any two large enough $A, B \subseteq V$, the number of edges between A and B can be estimated by thinking of G as a random graph where the probability of an edge between $v \in C_i$ and $w \in C_j$ is $d(C_i, C_j) = |E(C_i, C_j)|/(|C_i||C_j|)$. (The precise notion of approximation we need will be explained later.) Moreover, it is possible to choose a minimum partition size $m_{\min}$; often $m_{\min}$ is chosen such that "internal" edges among vertices from the same class are few enough to be ignored.

The original result was existential, but algorithms to construct a regular partition are known [6, 19, 23] which run in time polynomial in $|V|$ (for constant ε). This naturally suggests trying to use the partition classes in order to obtain an approximation of the optimal clustering. Nevertheless, to the best of our knowledge, the only prior attempts to exploit the regularity lemma for clustering

are the papers of Speroto and Pelillo [38] and Sárközy, Song, Szemerédi and Trivedi [36]. They use the constructive versions of the lemma to find the reduced graph $\widetilde{G}$, and apply standard clustering algorithms to $\widetilde{G}$. Since the partition size m required by the lemma is an $1/\varepsilon^5$-level iterated exponential of m_{min} (and this kind of growth rate is necessary [26]), they propose heuristics to avoid this tower-exponential behaviour. However, the running time of their algorithms is at least n^ω, where $\omega \in [2, 2.373)$ is the exponent for matrix multiplication. Moreover, no theoretical bounds are provided on the quality of the clustering found by working with the reduced graph, even if no heuristics were applied.

To address these issues, we opt to use a weaker variant of the regularity lemma due to Frieze and Kannan [21, 22]. It has better quantitative parameters and gives an implicit description of the partition, which opens the door for local clustering.

5.2 Cut decompositions of matrices

The idea of Frieze and Kannan is to take any real matrix A with row set $\mathcal{R}$ and column set $\mathcal{S}$ with bounded entries and approximate it by a low-rank matrix of a certain form. (The case of interest for us is when A is the adjacency matrix of a graph.) Let $m = |\mathcal{R}|$ and $n = |\mathcal{S}|$. Given $R \subseteq \mathcal{R}, S \subseteq \mathcal{S}$ and a real density d, the *cut matrix* $D = CUT(R, S, d)$ is the rank-1 matrix defined by $D_{ij} = d$ if $(i, j) \in R \times S$, and 0 otherwise. We identify R and S with with column indicator vectors of length m and n, respectively. Then we can write $CUT(R, S, d) = d \cdot RS^t$. We define $A(R, S) = R^t AS = \sum_{i \in R} \sum_{j \in S} A_{ij}$. A *cut decomposition* of a matrix A *with relative error* ε is a set of cut matrices $\{D_i\}_{i \in [s]}$, where $D_i = CUT(R_i, S_i, d_i)$, such that for all $R \subseteq \mathcal{R}, S \subseteq \mathcal{S}$,

$$\left| A(R, S) - \sum_{i \in [s]} D_i(R, S) \right| \leq \varepsilon \cdot \|R\|_2 \|S\|_2 \cdot \sqrt{mn}.$$

Such a decomposition is said to have *width* s and *coefficient length* $\sqrt{d_1^2 + \ldots d_s^2}$.

Theorem 5.1 (Cut decompositions [22, Th. 1]) *Suppose A is a $\mathcal{R} \times \mathcal{S}$ matrix with entries in $[-1, 1]$ and $\varepsilon, \delta \in (0, 1)$ are reals. Then in time $\mathsf{poly}(1/\varepsilon)/\delta$ we can, with probability $1 - \delta$, find implicitly a cut decomposition of width $\mathsf{poly}(1/\varepsilon)$, relative error at most ε and coefficient length at most 6.*

Regarding the meaning of "implicit". By *implicitly finding* a cut decomposition $B = \sum_{i \in [s]} CUT(R_i, S_i, d_i)$ of a matrix A in time t, we mean that for any given pair $(x, y) \in \mathcal{C} \times \mathcal{R}$, we can compute all of the following in time t by making queries to A:

- the rational values $d_1, \ldots, d_s$;
- the indicator functions $\mathbb{I}[x \in R_i]$ and $\mathbb{I}[y \in S_i]$, for all $i \in [s]$;
- the value of the entry $B_{x,y}$.

In Appendix B we give a sketch of how Frieze and Kannan achieve this. We also observe that their algorithm is non-adaptive.

Specifying a maximum cut-set size. Suppose we start with arbitrary equitable partitions of the row set and column set of A into t pieces. We can then find cut decompositions of the t^2 submatrices induced by the partition, and combine them into a cut decomposition of the original matrix that satisfies $|S_i| \leq m/t, |T_i| \leq n/t$; the reader may verify that this preserves the bound on

relative error. This process can only increase the query and time complexities by an $O(t^2)$ factor (c.f. [22, Section 5.1]).

Application to adjacency matrices. Suppose A is the adjacency matrix of an unweighted graph $G = (V, E)$, and identify the sets $\mathcal{R}$ and $\mathcal{S}$ with V. Then $|\mathcal{R}| = |\mathcal{S}| = |V| = n$. Let $E(R, S)$ denote the number of edges between $R \subseteq V$ and $S \subseteq V$. Then $A(R, S) = E(R, S) + E(R \cap S, R \cap S)$ and the conclusion of Theorem 5.1 can be written as

$$\text{for all } R \subseteq \mathcal{R}, S \subseteq \mathcal{S},$$

$$E(R,S)+E(R\cap S, R\cap S) = \Big(\sum_{i\in[s]} d_i \cdot |R\cap R_i| \cdot |S\cap S_i|\Big) \pm \varepsilon n\sqrt{|R||S|}. \tag{5}$$

The last term can be bounded by εn^2. While the standard regularity lemma supplies a much stronger notion of approximation, this bound suffices for certain applications.

Weakly regular partitions. A *weakly ε-pseudo-regular partition* of V is a partition of V into classes $V^1, \dots, V^\ell$ such that for all disjoint $R, S \subseteq V$, $\big|E(R,S) - \sum_{i,j\in[\ell]} d(V^i, V^j) \cdot |R \cap V^i| \cdot |S \cap V^j|\big| \le \varepsilon n^2$, where $d(V^i, V^j) = \frac{E(V^i,V^j)}{|V^i||V^j|}$. If, in addition, the partition is equitable, it is said to be *weakly ε-regular*.

Given a cut decomposition of a graph with relative error ε and size s, we get an 2ε-weakly pseudo-regular partition of size $\ell \le 2^{2s}$ by taking the classes of the Venn diagram of $R_1, S_1, \dots, R_s, S_s$ with universe V. So we can enforce the condition that the sets $R_1, S_1, R_2, S_2, \dots$ partition the vertex set of G, at an exponential increase in the number of such sets. Furthermore, any weakly ε-pseudo-regular partition of size ℓ may be refined to obtain a weakly 3ε-regular partition of slightly larger size; see [22, Section 5.1].

Often the weak regularity lemma is stated thusly in terms of weakly regular partitions, but the formulation of Theorem 5.1 is stronger in that it allows us to estimate the number of edges between two sets in time $\mathsf{poly}(1/\varepsilon)$ provided that we know the sizes of their intersections with all R_i, S_i, even though the weakly regular partition has size $\ell = 2^{\mathsf{poly}(1/\varepsilon)}$.

5.3 Near-optimal clusterings and the local algorithm

Intuitively, two vertices v, w in the same class of a regular partition have roughly the same number of connections with vertices outside. Hence for any given clustering of the remaining nodes, the cost of placing v into any one of the clusters is roughly the same as the cost of placing w there, suggesting they belong together in an optimal clustering (if we can afford to ignore the cost due to internal edges in the regular partition). In other words, a regular partition can be "coarsened" into a good clustering; the best one can be found by considering all possible combinations of assigning partition classes to clusters and estimating the cost of each resulting clustering.

We can make this argument rigorous by using bounds derived from the weak regularity lemma to approximate the cost of the optimal clustering by a certain quadratic program. If we ignore the terms with a single variable squared, the optimum of this program does not change by much as long as the partition is sufficiently fine. Then one can argue that the modified program attains its optimum for an assignment of variables which can be interpreted as a clustering that puts everything from the same regular partition into the same cluster.

Lemma 5.2 *Let A be the adjacency matrix of a graph $G = (V, E)$ and $k \in \mathbb{N}$. Let $\{CUT(R_i, S_i, d_i)\}_{i\in[s]}$ be a cut decomposition of A with relative error $\frac{\varepsilon}{2k}$ and with $|S_i|, |T_i| \leq \frac{\varepsilon n}{8k}$ for all $i \in [s]$. Denote by C^* the optimal k-clustering, and by C the optimal k-clustering into classes that belong to the σ-algebra generated by $\bigcup_{i\in[s]}\{R_i, S_i\}$ over V. Then* $\mathrm{cost}(C) - \varepsilon n^2 \leq \mathrm{cost}(C^*) \leq \mathrm{cost}(C)$.

Proof. We use Equation (5) to introduce an "idealized" cost function ideal satisfying the following for any clustering X:

1. $|\mathrm{cost}(X) - \mathrm{ideal}(X)| \leq \frac{\varepsilon n^2}{2}$; and
2. $\mathrm{ideal}(C) \leq \mathrm{ideal}(X) + \frac{\varepsilon n^2}{2}$.

Taken together, these two properties imply the result.

For each k-clustering X into $X_1, \ldots, X_k$, define

$$\mathrm{ideal}(X) = -\frac{n}{2} + \sum_{j\in[k]} \left[\sum_{i\in[s]} \left(\frac{1-d_i}{2}\right) |X_j \cap R_i||X_j \cap S_i|\right] + \sum_{\substack{j,j'\in[k]\\ j\neq j'}} \left[\sum_{i\in[s]} d_i |X_j \cap R_i||X_{j'} \cap S_i|\right]. \tag{6}$$

For any $j, j' \in [k]$, $j \neq j'$, using Equation (5) it holds that

$$E(X_j, X_{j'}) = \Big[\sum_{i\in[s]} d_i |X_j \cap R_i||X_{j'} \cap S_i|\Big] \pm \frac{\varepsilon n}{2k}\sqrt{|X_j||X_{j'}|}.$$

Similarly,

$$E(X_j, X_j) = \frac{1}{2}\Big[\sum_{i\in[s]} d_i |X_j \cap R_i||X_j \cap S_i|\Big] \pm \frac{\varepsilon n}{2k}\sqrt{|X_j||X_j|}.$$

Therefore

$$\begin{aligned}
\mathrm{cost}(X) &= \sum_{j\in[k]} \binom{|X_j|}{2} - E(X_j, X_j) + \sum_{\substack{j,j'\in[k]\\ j\neq j'}} E(X_j, X'_j) \\
&= -\frac{n}{2} + \sum_{j\in[k]} \frac{1}{2}|X_j|^2 - E(X_j, X_j) + \sum_{\substack{j,j'\in[k]\\ j\neq j'}} E(X_j, X'_j) \\
&\leq \mathrm{ideal}(X) + \frac{\varepsilon n}{2k} \cdot \sum_{j,j'\in[k]} \sqrt{|X_j||X'_j|} \\
&= \mathrm{ideal}(X) + \frac{\varepsilon n}{2k} \cdot \Big(\sum_{j\in[k]} \sqrt{|X_j|}\Big)^2 \\
&\leq \mathrm{ideal}(X) + \frac{\varepsilon n}{2k} \cdot k\Big(\sum_{j\in[k]} |X_j|\Big) \\
&= \mathrm{ideal}(X) + \frac{\varepsilon n^2}{2},
\end{aligned}$$

where the last inequality is by Cauchy-Schwarz.

It remains to be shown that $\text{ideal}(X) \geq \text{ideal}(C) - \frac{\varepsilon n^2}{2}$; in other words, that there is an almost-optimal k-clustering under the ideal cost function whose pieces are unions of the pieces $V^1, \ldots, V^\ell$ of the Venn diagram of $S_1, T_1, \ldots, S_s, T_s$. To see this, write

$$R_i = \bigcup_{t | V_t \subseteq R_i} V^t, \quad S_i = \bigcup_{t' | V_{t'} \subseteq S_i} V^{t'}.$$

Then

$$|X_j \cap R_i| = \sum_{t | V_t \subseteq R_i} |X_j \cap V^t|, \quad |X_j \cap S_i| = \sum_{t' | V_{t'} \subseteq S_i} |X_j \cap V^{t'}|.$$

Therefore $\text{ideal}(X) + n/2$ is a quadratic form on the $k\ell$ intersection sizes $|X_j \cap V^t|$, $j \in [k], t \in [\ell]$:

$$\text{ideal}(X) + \frac{n}{2} = \sum_{\substack{j,j' \in [k] \\ i \in [s] \\ V^t \subseteq R_i, V^{t'} \subseteq S_i}} \lambda^i_{j,j'} |X_j \cap V^t| |X_{j'} \cap V^{t'}|,$$

where $\lambda^i_{j,j} = (1 - d_i)/2$ and $\lambda^i_{j,j'} = d_i$ when $j \neq j'$.

Now remove from this expression the terms where $t = t'$. Among these, the terms where $j \neq j'$ evaluate to zero because X_j and $X_{j'}$ are disjoint. Each of the terms where $t = t'$ and $j = j'$ has absolute value at most

$$|\lambda^i_{j,j}||X \cap V^t|^2 \leq 4|X||V^t| \leq \frac{\varepsilon n}{2k}|X_j|,$$

since $|\lambda^i_{j,j}| = |\frac{1-d_i}{2}| \leq 4$ from the bound on the coefficient length, and $|V^t| \leq \varepsilon n/(8k)$. Therefore the term removal changes the value of the ideal cost function by at most $\varepsilon n^2/2$.

For $(t, j) \in [\ell] \times [k]$, let $\alpha^t_j = |X_j \cap V^t|$. Let

$$\kappa^{t,t'}_{j,j'} = \mathbb{I}[t \neq t'] \cdot \sum_{\substack{i \in [k] \\ V_t \subseteq R_i \\ V_{t'} \subseteq S_i}} \lambda^i_{j,j'} \mathbb{I}[V^t \subseteq R_i \wedge V^{t'} \subseteq S_i].$$

Then we have seen that

$$\text{ideal}(X) + \frac{\varepsilon n^2}{2} \geq -\frac{n}{2} + \sum_{\substack{t,t' \in [\ell] \\ j,j' \in [k]}} \kappa^{t,t'}_{j,j'} \alpha^t_j \alpha^{t'}_{j'},$$

and $\kappa^{t,t}_{j,j'} = 0$. Hence finding the optimal k-clustering under the idealized cost function can be reduced, up to an additive error of $\frac{\varepsilon}{2}n^2$, to solving the following integer quadratic program:

$$\begin{aligned} \text{minimize} \quad & -\frac{n}{2} + \sum \kappa^{t,t'}_{j,j'} \alpha^t_j \alpha^{t'}_{j'} && (7) \\ \text{subject to} \quad & \sum_{j \in [k]} \alpha^t_j = |V^t|, && \forall t \in [\ell] \\ & \alpha^t_j \geq 0, && \forall t \in [\ell], j \in [k] \\ & \alpha^t_j \in \mathbb{N}. \end{aligned}$$

The reason is that any feasible solution for $\{\alpha_j^t\}$ gives a clustering by assigning α_1^t arbitrary elements of V^t to the first cluster, another α_2^t elements of V^t to the second cluster, and so on.

Because $\kappa_{j,j'}^{t,t} = 0$, there is an optimal solution to (7) in which for all $t \in [\ell]$, exactly one α_t^i is equal to $|V^t|$ and the rest are zero. Indeed, fix $\alpha_j^{t'}$ for all $t' \neq t$ and all j in a solution (which corresponds to fixing a k-clustering of $V \setminus V^t$). Then the objective function becomes a linear combination of $\alpha_1^t, \ldots, \alpha_k^t$, plus a constant term. Therefore it is minimized by picking the cluster $j \in [k]$ with the smallest coefficient and setting $\alpha_j^t = |V^t|$. □

We sketch now our second local algorithm.

Proof of Theorem 3.6. For any $k \in \mathbb{N}, \varepsilon \in (0,1)$, we show a local algorithm that achieves an $(1,\varepsilon)$-approximation to the optimal k-clustering in time $\mathsf{poly}(k/\varepsilon)$, after a preprocessing stage that uses $\mathsf{poly}(k/\varepsilon)$ queries and $2^{\mathsf{poly}(k/\varepsilon)}$ time. Theorem 3.6 then follows by setting $k = O(1/\varepsilon)$.

Fist compute a cut decomposition of A that satisfies the conditions of Lemma 5.2. By Theorem 5.1, it can be computed implicitly in $\mathsf{poly}(k/\varepsilon)$ time. Let $V^1, \ldots, V^\ell$ be the atoms of the σ-algebra, where $\ell = 2^{2s}$ and $s = \mathsf{poly}(k/\varepsilon)$. Observe that they can also be defined implicitly: given $x \in V$ we can compute in $\mathsf{poly}(s)$ time a $2s$-bit label that determines the unique V^t to which x belongs, namely the value of the $2s$ indicator functions $\mathbb{I}[x \in S_i], \mathbb{I}[x \in T_i]$.

Next we proceed to the more expensive preprocessing part. Consider a clustering all of whose classes are unions of $V^1, \ldots, V^\ell$. Any such clustering is defined by a mapping $g : [\ell] \to [k]$ that, for every $i \in [\ell]$, identifies the cluster to which all elements of V^i belong. We can try all the $\ell^k = 2^{\mathsf{poly}(k/\varepsilon)}$ possibilities for g, and for each of them and estimate the cost of the associated clustering to within $\varepsilon/(2k)$ with high enough success probability by sampling. (We omit the details.) If we select the best of them, by Lemma 5.2, it will have cost within εn^2 of the optimal one.

Now we have a "best" mapping g from $[\ell]$ to $[k]$ that, for every $i \in \ell$, tells us the cluster of the elements of V^i. Finally, note that for any $x \in V$, the appropriate $i \in [\ell]$ such that $x \in V^i$ can be determined in time $\mathsf{poly}(s) = \mathsf{poly}(k/\varepsilon)$, and then we can get a cluster label for x in time $\mathsf{poly}(k/\varepsilon)$ by computing $g(i)$. □

6 Lower bounds

We show that our algorithm from Section 4 is optimal up to constant factors by proving a matching lower bound for obtaining $(O(1),\varepsilon)$-approximations. For simplicity we consider expected approximations at first; later we prove that combining upper and lower bounds for expected approximations leads to lower bounds for finding bounded approximations with high confidence.

Theorem 6.1 *Let $c \geq 1$, $\varepsilon \in (1/n, 1/(100c))$. Finding an expected (c,ε)-approximation to the best clustering with probability $1/2$ requires $\frac{n}{4000\varepsilon c^2}$ queries to the similarity matrix.*

In addition, any local clustering algorithm achieving this approximation has query complexity $\Omega(1/(\varepsilon c^2))$. (This remains true even if we allow preprocessing, as long as its running time is bounded by a function of ε and not n.)

Proof. The first part implies the second because any $q(\varepsilon)$-query local clustering algorithm with preprocessing $p(\varepsilon)$ can be turned into an explicit $n \cdot q(\varepsilon) + p(\varepsilon)$-query clustering algorithm. Given a lower bound of $n \cdot l(\varepsilon)$ on the complexity of finding approximate (c,ε)-clusterings for large enough n, we get $n \cdot q(\varepsilon) + p(\varepsilon) \geq n \cdot l(\varepsilon)$ for all large enough n, which implies $q(\varepsilon) \geq l(\varepsilon)$ since $\lim_{n->\infty} p(\varepsilon)/n = 0$. So we prove the first claim.

By Yao's minimax principle, it is enough 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 $\mathbb{E}[\mathsf{OPT}(G)] \leq \frac{\varepsilon n^2}{c}$;
- for any *deterministic* algorithm making at most $n/(4000\varepsilon c^2)$ queries, the expected cost (over G) of the clustering produced exceeds $2\varepsilon n^2 \geq c \cdot \mathbb{E}[\mathsf{OPT}(G)] + \varepsilon n^2$.

Let $\alpha = \frac{1}{4c}$, $k = \frac{1}{32c\varepsilon}$ and $l = \frac{k^2\varepsilon n}{3} \geq \frac{n}{4000c^2\varepsilon}$. We can assume that c, k and $\alpha n/k$ are integral (here we use the fact that $\varepsilon > 1/n$). Let $A = \{1, \ldots, (1-\alpha)n\}$ and $B = \{(1-\alpha)n+1, \ldots, n\}$. Consider the following distribution $\mathcal{G}$ of graphs: partition the vertices of A into exactly k equal-sized clusters $C_1, \ldots, C_k$. The set of positive edges will be the union of the cliques defined by $C_1, \ldots, C_k$, plus an edge joining each vertex $v \in B$ to a randomly chosen element $r_v \in A$. Define the natural clustering of a graph $G \in \mathcal{G}$ by the classes $C'_i = C_i \cup \{v \in B \mid r_v \in C_i\}$ ($i \in [k]$). This clustering will have a few disagreements because of the negative edges between different vertices $v, w \in B$ with $r_v = r_w$. The cost of the optimal clustering of G is bounded by that of the natural clustering N, hence

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

We have to show that any algorithm making l queries to graphs drawn from $\mathcal{G}$ produces a clustering with expected cost larger than $2\varepsilon n^2$. This inequality holds provided that the output clustering C and the natural clustering N are at least 3ε-far apart. Indeed, reasoning about expected distances, N is ε/c-close to G, therefore any clustering that is 2ε-close to G is also $2\varepsilon + \varepsilon/c \leq 3\varepsilon$-close to N from the triangle inequality.

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. Let us analyze the distance between the natural clustering and the clustering found by the algorithm. For $v \in B$, let Q_v denote set of queries it makes from v to A and put $q_v = |Q_v|$. Clearly we can assume $q_v \leq k-1$. The total number of queries made is $q = \sum_{v \in B} q_v$.

As r_v is independent of all edges from $[n] - \{v\}$ to $[n] - \{v\}$, conditioning on the responses to all queries not involving v we still know that the probability that all responses are negative is $\Pr[r_v \notin Q_v] = 1 - q_v/k$. When this happens, the probability that r_v coincides with the algorithm's choice is at most $\frac{1}{k-q_v}$.

All in all we have that the probability that the algorithm puts v into the same cluster as r_v is bounded by $\frac{1}{k-q_v} + \frac{q_v}{k}$. Let us associate a 0-1 random variable a_v with this event and put $R = \sum_{v \in B} a_v$. Consequently,

$$\mathbb{E}[R] \leq \sum_{v \in B} \left(\frac{1}{k-q_v} + \frac{q_v}{k} \right) = \frac{q}{k} + \sum_{v \in B} \frac{q_v}{k-q_v}.$$

We will see below (Lemma 6.2) that the last term can be bounded by $2(m+q)/k$, where $m = |B| = \alpha n$. Therefore $\mathbb{E}[R] \leq \frac{3q+2m}{k}$.

Now note that any vertex with $a_v = 0$ introduces $2(n-m)/k \geq n/k$ new differences with the natural clustering. Thus the expected number of differences is at least

$$\begin{aligned} \left(m - \frac{3q+2m}{k} \right) \frac{n}{k} &= m \left(1 - \frac{2}{k} \right) \frac{n}{k} - \frac{3qn}{k^2} \\ &\geq \alpha \frac{n^2}{2k} - \frac{3qn}{k^2} > 4\varepsilon n^2 - \frac{3qn}{k^2} \geq 3\varepsilon n^2, \end{aligned}$$

because $q \leq l$.

□

Lemma 6.2 *Let $q_1, \ldots, q_m \in [0, k-1]$ with $\sum_{i=1}^{m} q_i = q$. Then*

$$\sum_{i=1}^{m} \frac{1}{k - q_i} \leq \frac{2(m+q)}{k}.$$

Proof. Let $\gamma = \frac{q}{m+q}$. Define the sets

$$A = \{i \in [m] \mid q_i \geq \gamma k\}$$

and

$$B = \{i \in [m] \mid q_i < \gamma k\}.$$

Observe that $|A| \leq \frac{q}{\gamma k} = \frac{m+q}{k}$. Then

$$\sum_{i=1}^{m} \frac{1}{k - q_i} \leq |A| + \frac{|B|}{(1-\gamma)k} \leq |A| + \frac{m}{(1-\gamma)k} \leq \frac{2(m+q)}{k}.$$

□

Finally, we argue that similar bounds hold for algorithms that obtain good approximation with high success probability.

Lemma 6.3 *Suppose $\mathcal{A}$ finds a (c, ε)-approximate clustering with success probability $1/2$ using q queries, and $\mathcal{B}$ finds an expected $(c, r \cdot \varepsilon)$-approximate clustering using q queries. Then there is an algorithm $\mathcal{C}$ that finds an expected $\big(c, 2\varepsilon + \log(2r) \cdot \exp(-2\varepsilon^2 q)\big)$-approximation using $2q \cdot \log(2r)$ queries.*

Proof. Algorithm $\mathcal{C}$ does the following:

1. Let $t \leftarrow \log r$.
2. Run t independent instantiations of $\mathcal{A}$ to find clusterings $C_{\mathcal{A}}^1, \ldots, C_{\mathcal{A}}^t$ with qt queries.
3. Run $\mathcal{B}$ independently to find an expected $(c, r \cdot \varepsilon)$-approximate clustering C with q queries.
4. Estimate the quality of these $t+1$ clusterings using q random samples for each of them.
5. Return the clustering with the smallest estimated error.

The query complexity bound of $\mathcal{C}$ is as stated. When one of the $t+1$ clusterings found is (c, ε)-approximate, the probability that we fail to return a $(c, 2\varepsilon)$-approximation is at most $p = \exp(-2\varepsilon^2 q) \cdot (t+1)$. In this case we bound the error of the clustering output by 1. So the contribution to the expected approximation due to this kind of failure is at most $(0, p)$. We assume from now on that this is not the case.

The probability that none of $C_{\mathcal{A}}^1, \ldots, C_{\mathcal{A}}^t$ is a (c, ε)-approximation is at most $2^{-t} \leq \frac{1}{r}$. In this case we output a $(c, 2\varepsilon)$-approximation. On the other hand, with probability at most $\frac{1}{r}$, we output a clustering that in expectation is a $(c, r \cdot \varepsilon)$-approximation. Therefore, the output is an expected $(c, 2\varepsilon) + \frac{1}{r}(c, r) + (0, p)$-approximation. □

Corollary 6.4 *Let $\varepsilon > 10^7/n$. Finding a (c, ε)-approximate clustering with confidence $1/2$ requires $q = \frac{n}{2 \cdot 10^6 \cdot c^2 \varepsilon}$ queries.*

Proof. We may assume $c \geq 3$. Take the algorithm $\mathcal{B}$ from Corollary 4.4 and plug it into Lemma 6.3. This gives an expected approximation of $(\max(c, 3), 2\varepsilon + 25\exp(-2\varepsilon^2 q)) \leq (c, 3\varepsilon)$ using $50q$ queries. The result now follows from Lemma 6.1. □

7 Extensions

Non-binary similarity function. In Section 1 we have introduced correlation clustering in its most general form, with a pairwise similarity function $\text{sim} : V \times V \to [0, 1]$, while the case we ave studied so far is that of a binary similarity function $\text{sim} : V \times V \to \{0, 1\}$. The general case can be reduced to this by "rounding the graph", i.e., by replacing a non-binary similarity score with either 0 or 1 according to which is the closest (breaking ties arbitrarily): Bansal *et al.* [10, Thm. 23] showed that if $\mathcal{A}$ is an algorithm that produces a clustering on a graph G with $0, 1$-edges with approximation ratio ρ, then running $\mathcal{A}$ on the rounding of G achieves an approximation ratio of $2\rho + 1$. Therefore our algorithms also provide $(O(1), \varepsilon)$ approximations for correlation clustering in the more general weighted case.

Neighborhood oracles. If, given v, we can obtain a linked list of the *positive* neighbours of v (in time linear in its length), then it is possible to obtain a multiplicative $(O(1), 0)$-approximation in time $O(n^{3/2})$, which is sublinear. Indeed, Ailon and Liberty [4] argue that with a neighborhood oracle, QuickCluster runs in time $O(n + OPT)$; if $OPT \leq n^{3/2}$ this is $O(n^{3/2})$. On the other hand, if we set $\varepsilon = n^{-1/2}$ in our algorithm, we obtain in time $O(n^{3/2})$ a $(O(1), n^{-1/2})$-approximation, which is also a $(O(1), 0)$-approximation when $OPT \geq n^{3/2}$. So we can run QuickCluster for $O(n^{3/2})$ steps and output the result; if it doesn't finish, we run our algorithm with $\varepsilon = n^{-1/2}$.

Distributed/streaming clustering. In Section 1 we mentioned that there are general transformations from local clustering algorithms into distributed/streaming algorithms. For our local algorithm from Section 4 we can do the following. Suppose that to each processor P is assigned a subset A_P of the pairs $V \times V$, so that P can compute (or has information about) whether there is a positive edge between x and y for the pairs $(x, y) \in A_P$. (The assignment of vertex pairs to processors can be arbitrary, as long as they partition $V \times V$.) Then each processor selects the same random vertex subset $S \subseteq V$ of size $O(1/\varepsilon)$, and discards (or does not query/compute) the edges not incident with S among those it can see (A_P). After this, each processor outputs, for each v, the pairs (v, w) in A_P (note that for each v, there are only $O(1/\varepsilon)$ different such pairs). With this information the pivot set T is the subset of S with no neighbour smaller than itself (in some random order), and then the label of v's cluster is the first element of T adjacent to v. This can be computed easily in another round.

Note that the sum of the memory used by all processors is $O(n/\varepsilon)$, so for constant ε we also get a (semi-)streaming algorithm that makes one pass over the data, with edges arriving in arbitrary order. In two passes we can reduce memory usage to $O(n+1/\varepsilon^2)$: first store the adjacency matrix of the subgraph induced by the random sample S, and compute the set T of pivots. In the second pass, keep an integer for each v that indicates the first element of T that has appeared as a neighbour of v in the edges seen so far. At the end, this integer will be v's cluster.

Variants of correlation clustering. The second algorithm (based on cut matrices) can easily be extended to chromatic correlation clustering [12], and to bi-clustering (co-clustering) [13].

8 Concluding remarks

This paper initiates the investigation into local correlation clustering, devising algorithms with sublinear time and query complexity. The tradeoff between the running time of our algorithms and the quality of the solution found is close to optimal. Moreover, our solutions are amenable to simple implementations and they can also be naturally adapted to the distributed and streaming settings in order to improve their latency or memory usage.

The notion of local clustering introduced in this paper opens an interesting line of work, which might lead to various contributions in more applied scenarios. For instance, the ability of local algorithms to (among others) quickly estimate the cost of the best clustering can provide a powerful a primitive for decision-making, around which to build new data analysis frameworks. The streaming capabilities of the algorithms may also prove useful in clustering large-scale evolving graphs: this might be applied to detect communities in on-line social networks.

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 $O(n^{3/2})$-time constant-factor approximation algorithm using neighborhood oracles that we discussed in Section 7 suggests that this may be a fruitful direction to pursue in further research. The question seems particulary relevant in order to apply local techniques to very sparse graphs.

References

[1] N. Ailon, R. Begleiter, and E. Ezra. Active learning using smooth relative regret approximations with applications. In *Proc. of 25th COLT*, pages 19.1–19.20, 2012.

[2] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *Journal of the ACM*, 55(5), 2008.

[3] N. Ailon, B. Chazelle, S. Comandur, and D. Liu. Property-preserving data reconstruction. *Algorithmica*, 51(2):160–182, 2008.

[4] N. Ailon and E. Liberty. Correlation clustering revisited: The "true" cost of error minimization problems. In *Proc. of 36th ICALP*, pages 24–36, 2009.

[5] N. Alon, S. Dar, M. Parnas, and D. Ron. Testing of clustering. *SIAM Journal on Discrete Mathematics*, 16(3):393–417, 2003.

[6] N. Alon, R. A. Duke, H. Lefmann, V. Rödl, and R. Yuster. The algorithmic aspects of the regularity lemma. *Journal of Algorithms*, 16(1):80–109, 1994.

[7] N. Alon, W. Fernández de la Vega, R. Kannan, and M. Karpinski. Random sampling and approximation of MAX-CSPs. *Journal of Computer and System Sciences*, 67(2):212–243, 2003.

[8] N. Alon and A. Shapira. A characterization of the (natural) graph properties testable with one-sided error. *SIAM Journal on Computing*, 37:1703–1727, 2008.

[9] M.-F. Balcan, A. Blum, and S. Vempala. A discriminative framework for clustering via similarity functions. In *Proc. of 40th STOC*, pages 671–680, 2008.

[10] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[11] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[12] F. Bonchi, A. Gionis, F. Gullo, and A. Ukkonen. Chromatic correlation clustering. In *Proc. of 18th KDD*, pages 1321–1329, 2012.

[13] S. Busygin, O. A. Prokopyev, and P. M. Pardalos. Biclustering in data mining. *Computers & Operations Research*, 35(9):2964–2987, 2008.

[14] M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.

[15] A. Czumaj and C. Sohler. Sublinear-time approximation algorithms for clustering via random sampling. *Random Structures and Algorithms*, 30(1–2):226–256, 2007.

[16] A. Czumaj and C. Sohler. Small space representations for metric min-sum k-clustering and their applications. *Theoretical Computer Science*, 46(3):416–442, 2010.

[17] E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187, 2006.

[18] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005.

[19] E. Fischer, A. Matsliah, and A. Shapira. Approximate hypergraph partitioning and applications. *SIAM Journal on Computing*, 39:3155–3185, 2010.

[20] E. Fischer and I. Newman. Testing versus estimation of graph properties. *SIAM Journal on Computing*, 37(2):482–501, 2007.

[21] A. M. Frieze and R. Kannan. The regularity lemma and approximation schemes for dense problems. In *Proc. of 37th FOCS*, pages 12–20, 1996.

[22] A. M. Frieze and R. Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.

[23] A. M. Frieze and R. Kannan. A simple algorithm for constructing Szemerédi's regularity partition. *Electronic Journal of Combinatorics*, 6, 1999.

[24] A. M. Frieze, R. Kannan, and S. Vempala. Fast Monte Carlo algorithms for finding low-rank approximations. *Journal of the ACM*, 51(6):1025–1041, 2004.

[25] I. Giotis and V. Guruswami. Correlation clustering with a fixed number of clusters. *Theory of Computing*, 2(13):249–266, 2006.

[26] T. Gowers. Lower bounds of tower type for Szemerédi's uniformity lemma. *Geometric and Functional Analysis*, 7(2):322–337, 1997.

[27] O. Hassanzadeh, F. Chiang, R. J. Miller, and H. C. Lee. Framework for evaluating clustering algorithms in duplicate detection. *PVLDB*, 2(1):1282–1293, 2009.

[28] T. Hofmann and J. M. Buhmann. Active data clustering. In *Advances in Neural Information Processing Systems 10 (NIPS)*, 1997.

[29] M. Karpinski and W. Schudy. Linear time approximation schemes for the Gale-Berlekamp game and related minimization problems. In *Proc. of 41st STOC*, pages 313–322, 2009.

[30] S. Kim, S. Nowozin, P. Kohli, and C. D. Yoo. Higher-order correlation clustering for image segmentation. In *NIPS*, pages 1530–1538, 2011.

[31] J. Komlós, A. Shokoufandeh, M. Simonovits, and E. Szemerédi. The regularity lemma and its applications in graph theory. In *Proc. of 19th STACS*, pages 84–112, 2000.

[32] N. Mishra, D. Oblinger, and L. Pitt. Sublinear time approximate clustering. In *Proc. of 12nd SODA*, pages 439–447, 2001.

[33] M. Parnas and D. Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1-3):183–196, 2007.

[34] R. Rubinfeld, G. Tamir, S. Vardi, and N. Xie. Fast local computation algorithms. In *Proc. of second ITCS*, pages 223–238, 2011.

[35] M. E. Saks and C. Seshadhri. Local monotonicity reconstruction. *SIAM Journal on Computing*, 39(7):2897–2926, 2010.

[36] G. N. Sárközy, F. Song, E. Szemerédi, and S. Trivedi. A practical regularity partitioning algorithm and its applications in clustering. *Computing Research Repository*, abs/1209.6540, 2012.

[37] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1-2):173–182, 2004.

[38] A. Sperotto and M. Pelillo. Szemerédi's regularity lemma and its applications to pairwise clustering and segmentation. In *Proc. of sixth Conference in Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 13–27, 2007.

[39] D. A. Spielman and S.-H. Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Computing*, 42(1):1–26, 2013.

[40] J. Suomela. Survey of local algorithms. *ACM Computing Surveys*, 45(2):24:1–24:40, Mar. 2013.

[41] C. Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In *Proc. of 15th SODA*, pages 526–527, 2004.

[42] A. C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proc. of 18th FOCS*, pages 222–227, 1977.

A A sharper bound for Lemma 4.10

Lemma A.1 *Let $a_0 \in (0,1)$ and define a sequence by $a_{i+1} = a_i(1-a_i)$ for $i \geq 1$. Then for all $j \geq 1$,*

$$a_j \leq \frac{1}{\frac{1}{\widehat{a}_0} + j + \ln(\widehat{a}_0 j) - o_j(1)},$$

where $\widehat{a}_0 = \min(a_0, 1-a_0)$.

Proof. Since replacing a_0 with $1-a_0$ does not affect the terms a_j for $j \geq 1$, we can assume $a_0 = \widehat{a}_0 \leq 1/2$, in which case the result holds also for $j = 0$. Set $m_i = \frac{1}{a_i}$ for all $i \geq 0$. Then $m_0 = \frac{1}{a_0} \geq 2$ and

$$m_{i+1} - m_i = \frac{1}{\frac{1}{m_i}(1-\frac{1}{m_i})} - m_i = \frac{m_i^2}{m_i - 1} - m_i = 1 + \frac{1}{m_i - 1}.$$

In other words, for all $i \geq 1$ we have

$$m_i = m_0 + i + \sum_{j=0}^{i-1} \frac{1}{m_j - 1}.$$

Thus $m_i \geq m_1 \geq 4$, $m_i \leq m_0 + \frac{4}{3}i \leq m_0 + 2i + 1$, and

$$m_i \geq m_0 + i + \sum_{j=1}^{i-1} \frac{1}{m_0 + 2j} = m_0 + i + \frac{1}{2}\sum_{j=1}^{i-1} \frac{1}{\frac{m_0}{2} + j}.$$

Since by the integral test

$$\ln\left(\frac{b+1}{a}\right) \leq \sum_{k=a}^{b} \frac{1}{k} \leq \ln\left(\frac{b}{a-1}\right),$$

we have

$$\sum_{j=1}^{i-1} \frac{1}{\frac{m_0}{2} + j} \geq \ln\left(1 + \frac{2(i-1)}{m_0 + 2}\right) \geq \ln\left(1 + \frac{i-1}{m_0}\right),$$

so

$$m_i \geq m_0 + i + \frac{1}{2}\ln\left(1 + \widehat{a}_0 \cdot (i-1)\right),$$

as we wished to show. □

B Finding cut decompositions implicitly

We give here an overview of Frieze and Kannan's method [22]. In essence, the process works with the submatrix induced by certain randomly chosen subsets U, V of size $\mathsf{poly}(1/\varepsilon)$ and defining $\mathbb{I}[x \in R_i]$ and $\mathbb{I}[y \in R_i]$ in terms of the adjacencies (matrix entries) of x and y with U and V.

We start with the following simple exponential-time algorithm for finding cut decompositions. Suppose we have found cut matrices $D_0, \ldots, D_{i-1}$ and we want to find D_i. Let $W_i = A - \sum_{j<i} D_j$ be the residual matrix. While there exist sets R'_i, S'_i with

$$|W_i(R'_i, S'_i)| \geq \varepsilon \sqrt{|R'_i||S'_i|}\sqrt{mn}, \tag{8}$$

let $R_i = R'_i, S_i = S'_i$, $d_i = W_i(R_i, S_i)/(|R_i||S_i|)$ and add $D_i = CUT(R_i, S_i, d_i)$ to the decomposition. An easy computation shows that the squared Frobenius norm of the residual matrix decreases by $W_i(R'_i, S'_i)^2/(|R'_i||S'_i|)$, i.e., at least an ε^2 fraction of $\|A\|_F^2 \leq mn$. Therefore this process cannot go on for more than $1/\varepsilon^2$ steps. This gives a non-constructive proof of existence of cut decompositions.

How to make this procedure run in time independent of the matrix size? We can cut some slack here by replacing ε with some polynomial of ε with a larger exponent. Frieze and Kannan pick a row set $R_i \subseteq \mathcal{R}$ and then use a sampling-based procedure to construct a column set $S_i \subseteq \mathcal{S}$ such that the $R_i \times S_i$ submatrix is sufficiently dense. Provided that the entries in the matrix W_i remain bounded and inequality (8) holds for some R'_i, S'_i, they are able to find $R_i \in \mathcal{R}, S_i \in \mathcal{S}$ such that

$$|W_i(R_i, S_i)| \geq \mathsf{poly}(\varepsilon) \cdot mn,$$

which implies an $\mathsf{poly}(\varepsilon)$-fractional decrease in the squared Frobenius norm of the residual matrix. They show that, with probability at least $\mathsf{poly}(\varepsilon)$, we can take for P_i the set of all $x \in \mathcal{R}$ with $W_i(x, v) \cdot \nu \geq \nu^2$ for some randomly chosen $v \in \mathcal{C}$ and $\nu \in [-1, 1]$; and for S_i the set of all $y \in \mathcal{C}$ with $W_i(R_i, y) \cdot \nu \geq 0$.

We need to deal with how to represent the sets R_i, S_i used in the decomposition in an implicit manner. We will write down a predicate that, given $i \in [s]$ and $x \in \mathcal{R}$ (resp., $y \in \mathcal{S}$), tells us whether $x \in R_i$ (resp., $y \in S_i$) and can be evaluated in time $\mathsf{poly}(1/\varepsilon)$ by making queries to A. Although the size of $R_i \subseteq \mathcal{R}$ may be linear in m, its definition makes it possible to check for membership in R_i with one query to W_i. The set S_i, for its part, does not admit such a quick membership test, so Frieze and Kannan work with an approximation achieved by replacing R_i with a $\mathsf{poly}(1/\varepsilon)$-sized portion thereof in the definition of S_i. With the new definition, membership in R_i and S_i can be computed in time $\mathsf{poly}(1/\varepsilon)$, as we shall see. Also, the density $d_i = W_i(R_i, S_i)/(|R_i||S_i|)$ can be estimated to within $\pm\varepsilon^2 mn/16$ accuracy by sampling with $\mathsf{poly}(1/\varepsilon)$ queries to W_i.

Summarizing, we can build a cut decomposition in the following way. Let $s = \mathsf{poly}(1/\varepsilon)$. At at stage i, $i = 0, 1 \ldots, s-1$, the first i cut matrices $CUT(R_i, S_i, d_i)$ are implicitly known. Given the previous i cut matrices, the residual matrix W_i is given by

$$W_i(x, y) = A(x, y) - \sum_{j<i} d_j \cdot \mathbb{I}[x \in R_j] \cdot \mathbb{I}[y \in S_j];$$

extend the notation to sets in the obvious manner.

The set R_i is defined in terms of a random element $v_i \in \mathcal{C}$ and a random real $\nu_i \in [-1, 1]$ by

$$R_i = \{x \in \mathcal{R} \mid W_i(x, v_i) \cdot \nu \geq \nu^2\}.$$

The set S_i is defined in terms of R_i and a random sample $U_i \subseteq \mathcal{R}$ of size $\mathsf{poly}(1/\varepsilon)$ by

$$S_i = \{y \in \mathcal{S} \mid W_i(U_i \cap R_i, y) \cdot \nu \geq 0\}.$$

Finally, the density d_i is defined in terms of another random sample $Z_i \subseteq \mathcal{R} \times \mathcal{S}$ of size $\mathsf{poly}(1/\varepsilon)$ by

$$d_i = \mathop{\mathbb{E}}_{(x,y)\in Z_i}[W_i(x,y) \mid (x,y) \in R_i \times S_i].$$

Let $U = \bigcup_i (U_i \cup \Pi_1(Z_i))$, $V = \bigcup_i (V_i \cup \Pi_2(Z_i))$. We need to compute compute $W_i(u,v)$, $\mathbb{I}[u \in R_j]$, $\mathbb{I}[v \in S_j]$ and d_j for all $(u,v) \in U \times V$, $j \leq i$. This can be done in time $\mathsf{poly}(s/\varepsilon)$ using dynamic programming and the formulas above. This allows us to compute $W_i(x,y)$ for all x, y in time $\mathsf{poly}(s/\varepsilon) = \mathsf{poly}(1/\varepsilon)$.