Sublinear Time and Space Algorithms for Correlation Clustering via Sparse-Dense Decompositions

Sepehr Assadi
Department of Computer Science, Rutgers University, Piscataway, NJ, USA

Chen Wang
Department of Computer Science, Rutgers University, Piscataway, NJ, USA

Abstract

We present a new approach for solving (minimum disagreement) correlation clustering that results in sublinear algorithms with highly efficient time and space complexity for this problem. In particular, we obtain the following algorithms for n-vertex $(+/-)$-labeled graphs G:

- A sublinear-time algorithm that with high probability returns a constant approximation clustering of G in $O(n \log^2 n)$ time assuming access to the adjacency list of the $(+)$-labeled edges of G (this is almost quadratically faster than even reading the input once). Previously, no sublinear-time algorithm was known for this problem with any multiplicative approximation guarantee.
- A semi-streaming algorithm that with high probability returns a constant approximation clustering of G in $O(n \log n)$ space and a single pass over the edges of the graph G (this memory is almost quadratically smaller than input size). Previously, no single-pass algorithm with $o(n^2)$ space was known for this problem with any approximation guarantee.

The main ingredient of our approach is a novel connection to **sparse-dense graph decompositions** that are used extensively in the graph coloring literature. To our knowledge, this connection is the first application of these decompositions beyond graph coloring, and in particular for the correlation clustering problem, and can be of independent interest.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Computing methodologies → Machine learning

Keywords and phrases Correlation Clustering, Sublinear Algorithms, Semi-streaming Algorithms, Sublinear time Algorithms

Digital Object Identifier 10.4230/LIPIcs.ITCS.2022.10

Related Version *Full Version*: `https://arxiv.org/abs/2109.14528`

Funding This work is supported in part by a NSF CAREER Grant CCF-2047061, and a gift from Google Research.

Acknowledgements We thank the anonymous reviewers of ITCS 2022 for their many insightful suggestions that helped with improving the presentation of this paper.

1 Introduction

Correlation clustering is an extensively studied problem in theoretical computer science and machine learning. In this problem, we are given a complete undirected graph $G = (V, E)$ with edges labeled by $(+)$ or $(-)$. The general goal is to cluster the vertices in a way that $(+)$ edges appear more inside the clusters and $(-)$ edges appear more outside. Correlation clustering has found its applications in various areas, including image segmentation [31], document clustering [10], community detection [38], cross-lingual link detection [26], matrix decomposition [9], among others [12, 17].

One of the most popular optimization objectives for correlation clustering is disagreement minimization, wherein the goal is to minimize the total number of $(+)$ edges that cross different clusters and $(-)$ edges that are inside the same clusters. We study this disagreement minimization variant of correlation clustering in this paper. The problem is known to be both NP-hard and APX-hard, and there is a classical polynomial-time algorithm that achieves 2.06-approximation [10]. Disagreement minimization has been explored under various contexts, including the semi-random model [19], fair clustering [2], quantum approximation [42], and local clustering [11, 30], among others.

Nevertheless, for applications to modern massive datasets, even the efficiency of the polynomial-time approximation algorithms become insufficient. In particular, for a modern massive graph, even simple tasks like storing and processing all the edges *once* becomes challenging. Therefore, there is a quest for obtaining *sublinear* algorithms for correlation clustering. In such algorithms, the resource costs are usually asymptotically smaller than the input size, which allows correlation clustering to scale up to massive datasets.

Two of the most canonical examples of sublinear algorithms are *sublinear-time* algorithms and *(sublinear-space) streaming* algorithms. The former model assumes the data is provided to the algorithm in a specific format, say, the adjacency lists of the input graph, and one can query each entry of the input in $O(1)$ time; the goal is then to solve the problem faster than even reading the entire input once. The latter model instead focuses on space of the algorithms by assuming the data is presented to the algorithm in a stream and the goal is to process this stream in a space much smaller than the input size. In light of the above discussion, we study the following fundamental question in this paper:

Can we design sublinear time and/or space algorithms for correlation clustering?

This question and similar variants have already been pursued extensively in the literature. For sublinear-time algorithms, [11, 27] designed algorithms that given access to the adjacency matrix of G, run in $O(n/\varepsilon)$ time and output a 3-multiplicative plus $(\varepsilon \cdot n^2)$-additive approximation to correlation clustering. Moreover, impossibility results by [11, 13] prove that these algorithms are effectively optimal in a sense that one needs[1] $\widetilde{\Omega}(n^2)$ additive error whenever working with $\widetilde{O}(n)$ time algorithms in the adjacency matrix access model. These results however leave open the possibility of other natural access models to the input such as adjacency lists access, employed extensively both in theory and practice.

For sublinear-space algorithms, the "sweet spot" for correlation clustering is considered *semi-streaming* algorithms [24] that have space complexity $\widetilde{O}(n)$ which is proportional to the answer itself [3, 18, 21]. The first semi-streaming algorithm for this problem is due to [18] and obtains $(3+\varepsilon)$-approximation in $O(\frac{\log^2(n)}{\varepsilon})$ passes. This algorithm was improved by [3] to 3-approximation in $O(\log\log n)$ passes. Most recently, [21] presented a novel algorithm with $O(1)$-approximation in $O(1)$ passes[2]. These results however come short of providing any non-trivial guarantees for *single-pass* algorithms, which are by far the most studied and practically appealing variants of (semi-)streaming algorithms[3].

In this work, we answer this fundamental question in the affirmative by designing highly efficient sublinear-time and sublinear-space algorithms for $O(1)$-approximation of correlation clustering in these models: An $\widetilde{O}(n)$-time algorithm assuming adjacency lists access model, and a semi-streaming algorithm in a single pass.

[1] Throughout, we use $\widetilde{\Omega}(f(n))$ and $\widetilde{O}(f(n))$ to suppress dependence on $\operatorname{poly}\log(n)$ factors.
[2] While the constant in number of passes in [21] is not stated in their paper, it appears to be 6 passes.
[3] Beside being quantitatively more efficient, single-pass algorithms are qualitatively more appealing because they can process data generated "on the fly" without ever having to store it even once (e.g., in applications in network monitoring).

1.1 Our Contributions

Our first main result is a sublinear-time algorithm that instead of adjacency matrix in prior work [11, 13, 27], works with the adjacency lists of (+)-labeled edges and bypasses the strong impossibility results of [11, 13]. Formally,

▶ **Theorem 1.** *There exists a randomized algorithm that given the adjacency lists of the* (+)*-labeled subgraph of any labeled graph, with high probability*[4] *outputs an* $O(1)$*-approximation of correlation clustering in* $O(n\log^2 n)$ *time and* $O(n\log n)$ *queries.*

To our knowledge, prior to our work, no $o(n^2)$ time algorithm for multiplicative-approximation of correlation clustering was known (under any access model). We shall formally define the access model in Theorem 1 in Section 2.2 but basically it involves providing the algorithm with query access to the (+)-edges incident on each vertex individually. This seems to be a natural access from a practical point of view in many applications. For instance, in the applications of coreference [20] and cross-lingual link detection [26], the "natural" labels available are often the positive ones (e.g. the "co-occurance" and the "article similarity"), and the negative labels are usually inferred based on the positive edges. In the full version, we further investigate other natural sublinear-time access models such as adjacency lists access to the labeled graph itself or (−)-labeled subgraph instead and prove that no multiplicative approximation is possible in these models in $o(n^2)$ time. This highlights our model as the more theoretically-natural one for this problem also.

Our second main result is a single-pass semi-streaming algorithm for correlation clustering.

▶ **Theorem 2.** *There exists a randomized algorithm that with high probability computes an* $O(1)$*-approximation of correlation clustering in* $O(n\log n)$ *space and a single pass over the edges of any given labeled graph. Moreover, each edge insertion can be processed in* $O(\log(n))$ *time, and the post-processing time is bounded by* $O(n\log^2(n))$*.*

To our knowledge, no $o(n^2)$ space streaming algorithms was known for this problem in a single pass before our work. The only single-pass algorithm for this problem that we are aware of is due to [3] that requires $\widetilde{O}(n+m)$ space on graphs with m (−)-labeled edges which can be $\Omega(n^2)$ space[5]. We further show (in the full version) that our algorithm in Theorem 2 can be extended to other streaming models such as when only (+)- or (−)-labeled edges are arriving, or even to dynamic streams, still in $\widetilde{O}(n)$ space.

1.2 Our Techniques

The earlier work on sublinear algorithms for correlation clustering in [3, 11, 18, 27] were all based on implementing the so-called *Pivot* method of [4] via sublinear algorithms. The Pivot method is based on computing a *random-order maximal independent set* of (+)-labeled edges and achieves a 3-approximation. This method however does not seem particularly suitable for either sublinear-time or (single-pass) streaming algorithms: it is known that computing *any* type of maximal independent set (let alone the one required by the Pivot method) requires $\Omega(n^2)$ time given access to both adjacency lists and matrix of the input graph [7, 8] as well as $\Omega(n^2)$ space in single-pass streams [7, 22].

[4] Throughout, we use the therm "with high probability" to refer to with probability at least $1-\frac{1}{n^c}$ for some large constant $c>0$.

[5] Note that from a purely streaming point of view, one can entirely store a graph with m (−)-labeled edges in $\widetilde{O}(n+m)$ space (even in a dynamic stream), and then solve the problem exactly on the stored graph at the end of the stream in exponential time.

In a recent elegant work, [21] presented an interesting new insight on the problem. Their approach is based on trimming down the edges of the graph in multiple steps into $\widetilde{O}(n)$ edges that can be stored in the memory and finding connected components of this trimmed graph. The authors then show that placing these connected components into their own clusters achieves an $O(1)$-approximation to the problem. The proof of this part is done via a charging scheme that exploits the fact that vertices not in the same connected component have "different neighborhoods" while vertices inside the components are "tightly connected".

The idea demonstrated by [21] is quite neat and inspiring. Yet, their edge trimming approach does *not* give us any sublinear time nor single-pass semi-streaming algorithm. In fact, there are immediate roadblocks to adapt the edge trimming algorithm in [21] to both settings. For sublinear time algorithms, the implementation of edge trimming in [21] requires to compute the common neighborhood between the endpoints of *every* edge, which inevitably introduces an $\Omega(n^2)$ time complexity in the worst case. Moreover, the edge trimming algorithm in [21] processes edges in *steps*, i.e. the removal of the edges in the second step depends on the results of removal in the first step. As a result, it does *not* lend itself to a single-pass stream algorithm (rather, an $O(1)$-pass one as obtained in [21]

In this work, we first observe that this general strategy of partitioning a graph into different-neighborhood vs tightly-connected subgraphs is reminiscent of a classical approach in graph coloring literature referred to as *sparse-dense decompositions*. These decompositions have their root in the work of [32,35–37] (see also [33,34]) in graph theory and more recently have been at the core of several breakthrough results on graph coloring in distributed [15,28,29] and sublinear algorithms [5,7]. A typical sparse-dense decomposition partitions the graph into *sparse* vertices that have many non-edges in their neighborhood, and a collection of *almost-cliques* that are subgraphs which are close to a clique in a property testing sense. It is thus natural to wonder whether such decompositions can be used in place of the trimming step of [21], specially as some earlier work in [7] have already shown ways of finding these decompositions via different sublinear algorithms.

The first challenge in implementing this strategy is that these decompositions are almost exclusively tailored toward maximum-degree Δ of the graph, in the sense that their sparse vertices include all vertices with degree, say, $< 0.9\Delta$, and their almost-cliques are only $\approx \Delta$-cliques. While this is quite natural for graph coloring problems such as $(\Delta+1)$-coloring and alike, such a decomposition would not be particularly helpful for correlation clustering. The only exception that we are aware of is a recent decomposition of [5] for the so-called $(\deg+1)$-coloring problem which actually generates different types of sparse vertices and almost-cliques that are proportional to degree of individual vertices.

It turns out however that the decomposition of [5] is *too rigid* to be used in the context of the correlation clustering and the charging framework of [21] (we elaborate more on this in Section 3). On top of that, the decomposition of [5] is primarily used as a structural result in [5] and its only known algorithmic implementation requires using several instantiations of the algorithm of [7], which does not result in simple nor particularly efficient algorithms for the decomposition[6].

Our main technical ingredient is then to design a new sparse-dense decomposition that remedies this situation. We state our decomposition informally here and postpone the detailed and lengthy definitions to Theorem 5 (see also Section 2 for missing notation).

[6] We emphasize that main results of both [5, 7] rely on *existence* of such a decomposition and do not require an algorithm for finding it (although [7] give such algorithms also). This is different from our purpose of using the decomposition as it is only useful to us if it can be found algorithmically.

A (Yet Another) Sparse-Dense Decomposition: For any small constant $\varepsilon > 0$, vertices of any graph $G = (V, E)$ (not necessarily a labeled graph) can be decomposed into the following sets:

- *Sparse vertices*: each sparse vertex v has approximately $\varepsilon \cdot \deg(v)$ neighbors u such that $N(v)$ and $N(u)$ differ in approximately $\varepsilon \cdot \max\{\deg(v), \deg(u)\}$ vertices[a].
- *Dense vertices*: each dense vertex v belongs to an almost-clique of size approximately $(1 \pm \varepsilon) \cdot \deg(v)$, where an almost-clique is a subgraph of G that can be turned into an actual clique by changing approximately ε-fraction of edges of each of its vertices.

Moreover, there is an algorithm that samples $O(n \log(n))$ edges of G (from a certain non-uniform distribution) and uses degrees of vertices of G to compute this decomposition in $O(n \log^2 n)$ time.

[a] Beside the recovery algorithm, this is the guarantee that is different from [5] and needed for correlation clustering.

We remark that our way of defining and forming the decomposition is quite different from all recent algorithmic approaches for sparse-dense decompositions in [5, 7, 15, 28, 29]. Instead, to be able to provide the per-vertex guarantee needed by our decomposition, we follow the classical work of [36] that seems to give a better handle on the properties of the decomposition. As a result, we also give the *first* efficient implementation of this type of decompositions via a sampling algorithm that is easily implementable in various computational models including sublinear algorithms studied in this paper.

At this point, our task of designing sublinear algorithms is simple. Firstly, we show that given the decomposition of the $(+)$-labeled subgraph of the input, there is a natural way of forming an $O(1)$-approximation correlation clustering (see Theorem 6), following the approach of [21]. Basically, sparse vertices of the decomposition are so costly even for optimum solution that one might as well place them in singleton clusters; on the other hand, each almost-clique of dense vertices is so closely connected that the best strategy is to cluster them together. Secondly, the sampling algorithm that creates this decomposition is simple enough that it can easily be implemented via simple sublinear algorithms (see Theorems 12 and 13, and algorithms under broader settings in the full version).

In conclusion, we found the application of sparse-dense decompositions to correlation clustering (and graph clustering) quite natural and hope our work paves the path for further study of this connection. Moreover, unlike almost all aforementioned work that uses sparse-dense decompositions as a subroutine in much more complicated algorithms and proofs, here the main bulk of work is in designing the decomposition itself; as such, this application can perhaps find its way as a gentle(r) introduction to sparse-dense decompositions.

1.3 Related Work

Correlation clustering is one of the most well-studied clustering problems. Apart from the classical settings where the edges are either $(-)$ or $(+)$, results have also been developed under general graphs, where the edge weights are real numbers and the graphs are not necessarily complete. On this front, the work of [23] gives an $O(\log(n))$-approximation algorithm in polynomial time. The NP-hardness result on labeled (complete) graphs automatically applies to general graphs, and it is further shown that the approximation even on weighted *complete* graphs is APX-hard [16, 23].

Beyond the disagreement minimization objective, another popular optimization target is agreement maximization, which aims to maximize the $(+)$ edges in the same clusters and $(-)$ edges across different clusters. Computing the exact solution of agreement maximization is also NP-hard. However, it admits a PTAS, rendering the objective more tractable for approximation [10]. Furthermore, for general graphs, the work of [16, 39] give algorithms that achieve 0.766 approximation in polynomial time. More recently, [1] proposed a new min-max objective, whose goal is to minimize the maximum number of disagreement edges inside each cluster. It is further shown in [1] that such an objective admits a worst-case $O(\log(n))$ approximation in polynomial time.

The quest for sublinear correlation clustering algorithms also goes outside sublinear-time and streaming models. For instance, correlation clustering under distributed settings, especially under the Massively Parallel Computation (MPC) models, has been extensively studied. On this front, [18] designs an algorithm that achieves $O(1)$-approximation in $O(\log(n))$ parallel rounds. The main technical ingredient of their algorithm is to simulate the Pivot algorithm in $O(\log(n))$ rounds. In the same spirit, one can adapt the streaming algorithm in [3] to a distributed algorithm with $O(\log\log(n))$ rounds, given a near-linear memory for each machine. The recent algorithm of [21] now improves this to an $O(1)$ approximation algorithm in constant many parallel rounds and even sublinear memory per machine (see also [14] for an MPC algorithm on bounded arboricity graphs). Finally, outside the MPC model, distributed correlation clustering algorithms have also been explored under PRAM and LOCAL models; see [25] and references therein.

In addition to the distributed setting, the "local" correlation clustering introduced by [11] is another interesting setting for sublinear algorithms. Under the model, an algorithm outputs the cluster of a *single* vertex at a time, and the cluster of each vertex is consistent with the "global" clustering. The algorithm of [11] runs in $O(1)$ time for each vertex; nevertheless, its approximation guarantee is weaker than ours since it introduces a $\Theta(n^2)$ additive error.

2 Preliminaries

Set notation. For two sets A and B, we use $A \triangle B := (A - B) \cup (B - A)$ to denote the symmetric difference of A and B. We say that a collection $\mathcal{S}$ of sets is *laminar* if for any two sets $A, B \in \mathcal{S}$, either $A \cap B = \emptyset$ or $A \subseteq B$ or $B \subseteq A$. For any laminar collection $\mathcal{S}$, we say that a set $S \in \mathcal{S}$ is a *root* if S is not a proper subset of any other set in $\mathcal{S}$.

Graph notation. For any graph $G = (V, E)$, and vertex $v \in V$, we use $N(v)$ to denote the neighbors of v and $E(v)$ to denote the edges incident on v. We say that a pair (u, v) is a *non-edge* in the graph G if there is not an edge between u and v in G.

Concentration inequalities. We use the following standard forms of Chernoff bound in our proofs.

▶ **Proposition 3** (Chernoff Bound; cf. [6]). *Let $X_1, X_2, \cdots, X_m$ be m independent random variables in $[0, 1]$. Define $X = \sum_{i=1}^{m} X_i$. Then, for every $\delta > 0$ and $t \geq 1$,*

$$\Pr\left(|X - \mathbb{E}[X]| \geq \delta \cdot \mathbb{E}[X]\right) \leq 2 \cdot \exp\left(-\frac{\delta^2}{2+\delta} \cdot \mathbb{E}[X]\right);$$

$$\Pr\left(|X - \mathbb{E}[X]| \geq t\right) \leq 2 \cdot \exp\left(-\frac{2t^2}{m}\right).$$

We also use the following form of Bernstein's inequality.

▶ **Proposition 4** (Bernstein's inequality; cf. [40]). *Let $X_1, \ldots, X_m$ be m independent random variables such that $\mathbb{E}[X_i] = 0$ and $|X_i| < M$ for all $i \in [m]$. Then, for any $t \geq 1$,*

$$\Pr\left(\sum_{i=1}^{m} X_i \geq t\right) \leq \exp\left(-\frac{t^2}{2\sum_{i=1}^{m}\mathbb{E}[X_i^2] + {}^2\!/_3 \cdot M \cdot m}\right).$$

2.1 Problem Definition

Throughout, by a *labeled* graph $G = (V, E)$, we mean a complete graph with edges in E labeled in $\{-1, +1\}$. We use G^+ and G^- to denote the subgraphs of G consisting of only $(+)$-edges and $(-)$-edges, respectively. We extend this definition analogously to neighbor-sets $N^+(v)$ and $N^-(v)$, and edge-sets $E^+(v)$ and $E^-(v)$, for every $v \in V$.

Suppose we are given a labeled graph $G = (V, E)$. Let $\mathcal{C}$ be any clustering of vertices of G into disjoints clusters $C_1, \ldots, C_k$. For any vertex $v \in V$, we use $\mathcal{C}(v)$ to denote the cluster $C_i \in \mathcal{C}$ that v belongs to. For any edge $e = (u, v)$, we define the *cost* of e in the clustering $\mathcal{C}$ as:

$$\mathsf{cost}_{\mathcal{C}}(e) = \begin{cases} 1 & \text{if } e \in G^+ \text{ and } \mathcal{C}(u) \neq \mathcal{C}(v) \\ 1 & \text{if } e \in G^- \text{ and } \mathcal{C}(u) = \mathcal{C}(v) \\ 0 & \text{otherwise} \end{cases}. \tag{1}$$

In words, cost of $(+)$-edge is 1 if its endpoints are clustered differently, and cost of a $(-)$-edge is 1 if its endpoints are clustered together. The *total cost* of a clustering $\mathcal{C}$ is then:

$$\mathsf{cost}(\mathcal{C}) = \sum_{e \in G} \mathsf{cost}_{\mathcal{C}}(e). \tag{2}$$

The goal in the correlation clustering problem is to find a clustering $\mathcal{C}$ that minimizes Eq (2).

2.2 Sublinear Algorithms Models

In this paper, we focus on two of the most canonical models of sublinear algorithms, namely, sublinear-time algorithms, and (sublinear-space) streaming algorithms. These models are defined formally as follows.

Sublinear-time algorithms. When working with sublinear-time algorithms, it is important to specify the exact data model as the algorithm does not even have time to read the input once. In this paper, we assume the algorithms are given access to the *adjacency lists* of the $(+)$-graph G^+ of the input labeled graph G. This means that the algorithm can query the following information in $O(1)$ time:

1. *Degree queries:* What is $\deg^+(v)$ of a given vertex $v \in V$?
2. *Neighbor queries:* What is the i-th vertex in $N^+(v)$ of $v \in V$ for $i \leq \deg^+(v)$?

The goal is to return a correlation clustering of G under cost function of Eq (2) quickly.

A remark about this model is in order. The standard query model for graph problems provides access to the adjacency lists (or matrix) of G itself (and not that of G^+). But in the context of labeled graphs, adjacency lists of G itself provides little information: degree queries are entirely uninformative (always return $n - 1$) and neighbor queries only reveal the label of the edge between vertex v to some other vertex u, similar to access to the adjacency matrix. Alternatively, we could have also considered access to the adjacency lists of the $(-)$-graph G^- instead, which at least is more informative than that of G.

Nevertheless, we prove that neither model allows for any non-trivial sublinear-time algorithm for correlation clustering with *any multiplicative* approximation guarantees (in the full version). In light of this impossibility result, and our sublinear-time algorithms, we believe the model we consider for this problem is most natural from the perspective of sublinear-time algorithms.

Semi-streaming algorithms. Semi-streaming algorithms focus on minimizing the space usage as opposed to time. In this model, the vertices of input labeled graph $G = (V, E)$ are known and the edges E arrive one by one in a stream together with their labels. The goal is to read this stream in the given order only once[7] and use only $O(n \cdot \text{polylog}\,(n))$ space measured in machine words of size $O(\log n)$ bits. At the end of the stream, the algorithm should return a correlation clustering of G (under cost function of Eq (2)).

Our streaming model is the same as the one studied by earlier work on this problem. But one can again wonder what would happen if only edges of G^+ or G^- are being streamed instead of G. It turns out unlike the sublinear-time model, these different choices do not matter much for our purpose. In the full version, we show that our algorithm can be extended to handle either of these cases, plus other natural variants such as dynamic (insertion-deletion) streams at the cost of increasing the space by at most $\text{polylog}\,(n)$ factor.

3 A (Yet Another) Sparse-Dense Decomposition

We present our sparse-dense decomposition in this section. Due to space limits, we only provide the statement of the decomposition theorem and high-level remarks. We defer the full proof of theorem and the algorithm that recovers it to the full version.

We state the theorem in a form that allows for its recovery via sublinear algorithms in subsequent sections – however, we opted to present the recovery algorithm in a model-independent manner as this general form can also be applicable in other models of computation not considered in this paper.

▶ **Theorem 5** (Sparse-Dense Decomposition (algorithmic version)). *There are absolute constants $\varepsilon_0, \eta_0 > 0$ such that the following is true. For every $\varepsilon < \varepsilon_0$, vertices of any given graph $G = (V, E)$ can be partitioned into the following sets:*

- ***Sparse vertices** V_{sparse}: Any vertex $v \in V_{sparse}$ has at least $\eta_0 \cdot \varepsilon \cdot \deg(v)$ neighbors u such that:*

$$|N(v) \triangle N(u)| \geq \eta_0 \cdot \varepsilon \cdot \max\{\deg(u), \deg(v)\}.$$

- ***Dense vertices partitioned into almost-cliques** $K_1, \ldots, K_k$: For every $i \in [k]$, each K_i has the following properties. Let $\Delta(K_i)$ be the maximum degree (in G) of the vertices in K_i, then:*
  1. *Every vertex $v \in K_i$ has at most $\varepsilon \cdot \Delta(K_i)$ non-neighbors inside K_i;*
  2. *Every vertex $v \in K_i$ has at most $\varepsilon \cdot \Delta(K_i)$ neighbors outside K_i;*
  3. *Size of each K_i satisfies $(1 - \varepsilon) \cdot \Delta(K_i) \leq |K_i| \leq (1 + \varepsilon) \cdot \Delta(K_i)$.*

Moreover, there is an absolute constant $c > 0$ and an algorithm that given access to only the following information about G, with high probability, computes this decomposition of G in $O(\varepsilon^{-2} \cdot n \log^2 n)$ time:

- ***Degree information:** Set of all vertices $v \in V$ plus their degrees $\deg(v)$;*

[7] Or a few times in case of multi-pass algorithms – our algorithm in this paper however is single-pass.

- ***Random edge samples:*** *A collection of sets $N_{\text{sample}}(v)$ of*

  $$t = c \cdot \varepsilon^{-2} \cdot \log n$$

  neighbors of each vertex $v \in V$ chosen independently and uniformly at random (with repetition);
- ***Random vertex samples:*** *A set* `Sample` *of vertices wherein each $v \in V$ is included independently with probability*

  $$p_v := \min\left\{\frac{c \cdot \log n}{\deg(v)}, 1\right\},$$

  together with all the neighborhood $N(v)$ of each sampled vertex $v \in$ `Sample`*.*

(The probability of success is over the random choice of edge and vertex samples.)

The sparse vertices in Theorem 5 are such that "many" of their neighbors have a "different" neighborhood than themselves. Thus, even though we refer to them as "sparse" to be consistent with prior sparse-dense decompositions, these vertices do not necessarily have a sparse neighborhood as in standard decompositions but rather have a 2-hop neighborhood that is very different than their 1-hop neighborhood.

The almost-cliques on the other hand, as the name suggests, are basically induced subgraphs of G on "similar degree" vertices that can be turned into an actual clique by changing a small fraction of edges in their neighborhood. This part is also different from typical decompositions in that the almost-cliques are allowed to have varying sizes tailored to degrees of individual vertices as opposed to a single size based on the maximum degree. The only other sparse-dense decomposition with such guarantees that we know of is that of [5]. However, both in terms of precise guarantees and also the construction, our Theorem 5 is quite different from [5]. To be specific:

- The sparse vertices in Theorem 5 have an *individual* guarantee on their "different" neighbors, while [5] makes an *aggregate* guarantee for the entire neighborhood of a vertex.
- The construction of [5] is based on the notion of *balanced-* and *friend-edges*, which does not seem to allow for the fine-grained guarantees required by our decomposition. Instead, the proof of Theorem 5 involves a more direct approach based on classical sparse-dense decompositions.
- Finally, the decomposition in [5] is a structural result while ours is constructive via the sampling algorithm, which is needed for our sublinear algorithms.

4 Correlation Clustering via the Sparse-Dense Decomposition

We are now ready to present a correlation clustering scheme based on the decomposition results in Section 3, applied to the underling G^+ graph. Our approach is to simply place the sparse vertices in separate singleton clusters and treat each almost-clique of the dense vertices as one disjoint cluster. Formally,

▶ **Theorem 6.** *Suppose $G = (V, E)$ is any labeled graph and $V = V_{sparse} \sqcup K_1 \sqcup \ldots \sqcup K_k$ is an ε-sparse-dense decomposition of G^+ for $\varepsilon > 0$ according to Theorem 5. Let $\mathcal{A}$ be the following clustering:*

- *Any vertex $v \in V_{sparse}$ is placed in a singleton cluster, i.e., $\mathcal{A}(v) = \{v\}$;*
- *Any almost-clique K_i forms a separate cluster, i.e., for any $v \in K_i$, $\mathcal{A}(v) = \{u \mid u \in K_i\}$.*

Then, $\mathcal{A}$ is an $O(\varepsilon^{-2})$-approximation correlation clustering of G.

The intuition behind the proof of Theorem 6 is simple: the (+)-neighborhood of sparse vertices is so different from that of their neighbors that no matter how we cluster them, we will need to pay a cost proportional to their degree; so we might as well cluster them individually. On the other hand, the almost-cliques are so tightly connected to each other and so loosely connected to outside by their (+)-edges that they simply form the best cluster possible themselves; so we cluster them that way also.

We formalize this intuition in this section. Our analysis of Theorem 6 is inspired by the recent work of [21]. The main difference is in using the decomposition of Theorem 5 instead of the rather ad-hoc and "multi-step" partitioning in [21] which is crucial for our sublinear algorithms (the decomposition allows us to also give a more modular proof by focusing on each part of the partition separately).

Throughout this section, fix $\mathcal{O}$ to be a fixed optimal clustering of G and recall that $\mathcal{A}$ denotes the clustering returned by Theorem 6. Similar to [21], we use a **charging scheme**: To any vertex $z \in V$ and any edge f incident on z, i.e., $f \in E^+(z) \sqcup E^-(z)$, we assign a value $\mathsf{charge}(z, f)$ as follows:

Charging scheme for the analysis of Theorem 6.

(i) Initially, $\mathsf{charge}(z, f) = 0$ for all $z \in V$ and $f \in E(z)$;
(ii) For any edge $e \in E$ with $\mathsf{cost}_{\mathcal{A}}(e) = 1$, we will find a collection of vertex-edge pairs, called the **charge-set** of e:

$$\text{ChargeSet}(e) \subseteq \{(z, f) \mid z \in V, \quad f \in E(z), \text{ and } \mathsf{cost}_{\mathcal{O}}(f) = 1\}.$$

For simplicity of notation, we define $\text{ChargeSet}(e) = \emptyset$ if $\mathsf{cost}_{\mathcal{A}}(e) = 0$.
(iii) Increase $\mathsf{charge}(z, f)$ for all $(z, f) \in \text{ChargeSet}(e)$ by $|\text{ChargeSet}(e)|^{-1}$.

The main part in this charging scheme is to find proper charge-sets for all edges. The following lemma establishes our desired property of the charging scheme.

▶ **Lemma 7.** *Suppose there is a choice of* $\text{ChargeSet}(e)$ *for edges* $e \in E$ *in the charging scheme such that for all* $z \in V$ *and* $f \in E(z)$, *we have* $\mathsf{charge}(z, f) \leq \alpha$ *for some* $\alpha \geq 1$. *Then,* $\mathsf{cost}(\mathcal{A}) \leq 2\alpha \cdot \mathsf{cost}(\mathcal{O})$.

By Lemma 7, we only need to find charge-sets of the given edges so that $\mathsf{charge}(z, f)$ is small for all vertex-edge pairs (z, f). This is done for edges of sparse and dense vertices separately in the next subsections.

A helper lemma. Before getting to the main part of the proof, we will prove a helper lemma that simplifies our task of finding charge-sets for (+)-edges in the later parts of the analysis. Roughly speaking, it states that if we have a collection of edges whose endpoints have sufficiently different neighborhood, then we can find a charge-set for all the given edges without increasing charge of any vertex-edge pair by much.

▶ **Lemma 8.** *Let* $\theta \in (0, 1)$ *be a parameter and* $\mathcal{E}$ *be any collection of edges in* E^+ *in the input labeled graph* G *such that for all* $\xi = (\alpha, \beta) \in \mathcal{E}$,

$$N^+(\alpha) \triangle N^+(\beta) \geq \theta \cdot \max\left\{\deg^+(\alpha), \deg^+(\beta)\right\}. \tag{3}$$

Then, there is a choice of $\text{ChargeSet}(\xi)$ *for all* $\xi \in \mathcal{E}$ *such that* $\mathsf{charge}(z, f) = O(\theta^{-1})$ *for all vertex-edge pairs* (z, f) *in* G.

Proof. We define $\mathrm{ChargeSet}(\xi)$ for any $\xi \in \mathcal{E}$ as follows:

- Type-1 charges: when $\mathsf{cost}_{\mathcal{O}}(\xi) = 1$. In this case, we simply set $\mathrm{ChargeSet}(\xi) = \{(\alpha, \xi)\}$ itself.
- Type-2 charges: when $\mathsf{cost}_{\mathcal{O}}(\xi) = 0$. This is the more challenging case. Note that in this case, we have that $\mathcal{O}(\alpha) = \mathcal{O}(\beta)$, and let us denote this cluster as $\mathcal{O}_{\alpha\beta}$. Consider any vertex $w \in N^+(\alpha) \triangle N^+(\beta)$:
  - Case A: $w \in N^+(\alpha)$ and $w \in N^-(\beta)$. In this case, there is $\mathsf{cost}_{\mathcal{O}}((w, \beta)) = 1$ if $\mathcal{O}(w) = \mathcal{O}_{\alpha\beta}$, and $\mathsf{cost}_{\mathcal{O}}((w, \alpha)) = 1$ if $\mathcal{O}(w) \neq \mathcal{O}_{\alpha\beta}$.
  - Case B: $w \in N^+(\beta)$ and $w \in N^-(\alpha)$. In this case, there is $\mathsf{cost}_{\mathcal{O}}((w, \alpha)) = 1$ if $\mathcal{O}(w) = \mathcal{O}_{\alpha\beta}$, and $\mathsf{cost}_{\mathcal{O}}((w, \beta)) = 1$ if $\mathcal{O}(w) \neq \mathcal{O}_{\alpha\beta}$.

  Therefore, in both cases, there is exactly one edge $f(w) \in \{(w, \alpha), (w, \beta)\}$ such that $\mathsf{cost}_{\mathcal{O}}(f(w)) = 1$. Let $z(w) \in \{\alpha, \beta\}$ be the vertex other than w incident on $f(w)$. We add all pairs $(z(w), f(w))$ to $\mathrm{ChargeSet}(\xi)$, i.e.,

$$\mathrm{ChargeSet}(\xi) = \left\{(z(w), f(w)) \mid w \in N^+(\alpha) \triangle N^+(\beta)\right\}.$$

  Given the bound on the size of $N^+(\alpha) \triangle N^+(\beta)$, we have that $|\mathrm{ChargeSet}(\xi)| \geq \theta \cdot \max\left\{\deg^+(\alpha), \deg^+(\beta)\right\}$.

An illustration of the type-2 charges can be found in Figure 1.

Let us now bound the distributed charges. We have three different choices for (z, f) that can belong to $\mathrm{ChargeSet}(\xi)$ for some edge $\xi \in \mathcal{E}$ as follows (a graphical exemplification can be found in Figure 2):

- A pair (α, ξ) charged by a type-1 charge, where $\xi \in \mathcal{E}$ and α is an endpoint of ξ: In this case $\mathsf{charge}(\alpha, \xi) = 1$ because there is only a single edge ξ that can make such a charge.
- A pair $(\alpha, f(w))$ charged by a type-2 charge, where $w \in N^+(\alpha) \triangle N^+(\beta)$ and $z(w) = \alpha$: For any such charge, we increase $\mathsf{charge}(\alpha, f(w))$ by

$$|\mathrm{ChargeSet}(\xi)|^{-1} \leq (\theta \cdot \deg^+(\alpha))^{-1}.$$

  At the same time, such a charge can only be made by edges from α to $\beta \in N^+(\alpha)$ (so that $(\alpha, \beta) \in E^+$), which are $\deg^+(\alpha)$ many. Thus, the total charge made in this case leads to $\mathsf{charge}(\alpha, f(w)) = O(\theta^{-1})$.
- A pair $(\beta, f(w))$ charged by a type-2 charge, where $w \in N^+(\alpha) \triangle N^+(\beta)$ and $z(w) = \beta$: For any such charge, we increase $\mathsf{charge}(\beta, f(w))$ by

$$|\mathrm{ChargeSet}(\xi)|^{-1} \leq (\theta \cdot \deg^+(\beta))^{-1}.$$

  At the same time, such a charge can only be made by edges from β to $\alpha \in N^+(\beta)$ (so that $(\alpha, \beta) \in E^+$), which are $\deg^+(\beta)$ many. Thus, the total charge made in this case leads to $\mathsf{charge}(\beta, f(w)) = O(\theta^{-1})$.

This concludes the proof of the lemma. ◀

Part I: Sparse Vertices

We now analyze the cost of edges incident on the sparse vertices V_{sparse}. We define $\mathsf{sparse\text{-}charge}(z, f)$ as the contribution from the sparse vertices to $\mathsf{charge}(z, f)$. We show that,

▶ **Lemma 9.** *There exist sets* $\mathrm{ChargeSet}(e)$ *for every ε-sparse vertex v and $e \in E(v)$ such that*

for all vertex-edge pairs (z, f): $\mathsf{sparse\text{-}charge}(z, f) = O(\varepsilon^{-2})$.

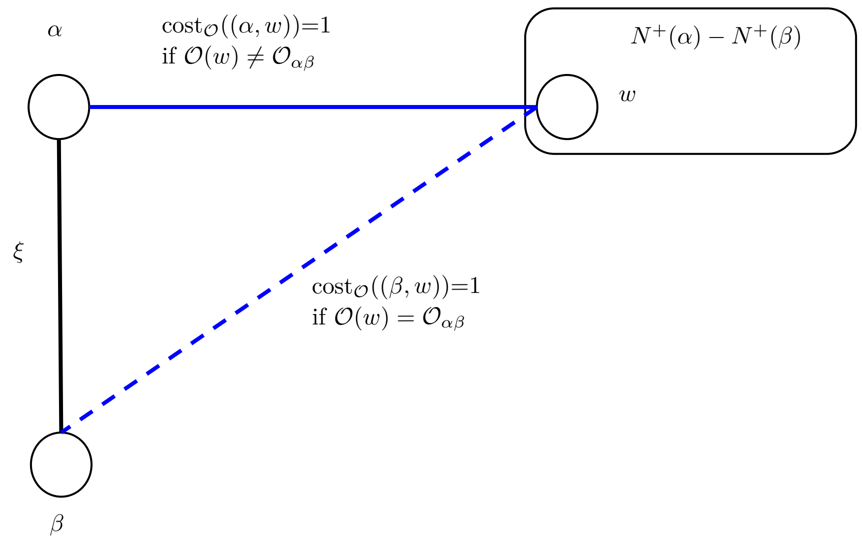
α
$\mathrm{cost}_{\mathcal{O}}((\alpha, w))$=1
if $\mathcal{O}(w) \neq \mathcal{O}_{\alpha\beta}$
$N^+(\alpha) - N^+(\beta)$
w
ξ
$\mathrm{cost}_{\mathcal{O}}((\beta, w))$=1
if $\mathcal{O}(w) = \mathcal{O}_{\alpha\beta}$
β

Figure 1 Illustration of the type-2 charge conditioning on α and β are in the same cluster $\mathcal{O}_{\alpha\beta}$. For each vertex $w \in N^+(\alpha) - N^+(\beta)$, if w is in $\mathcal{O}_{\alpha\beta}$, the negative edge (β, w) induces a cost of 1 in $\mathcal{O}$; otherwise, if w is in a different cluster, the positive edge (α, w) induces a cost of 1 in $\mathcal{O}$. Vertices in $N^+(\beta) - N^+(\alpha)$ works in the same way.

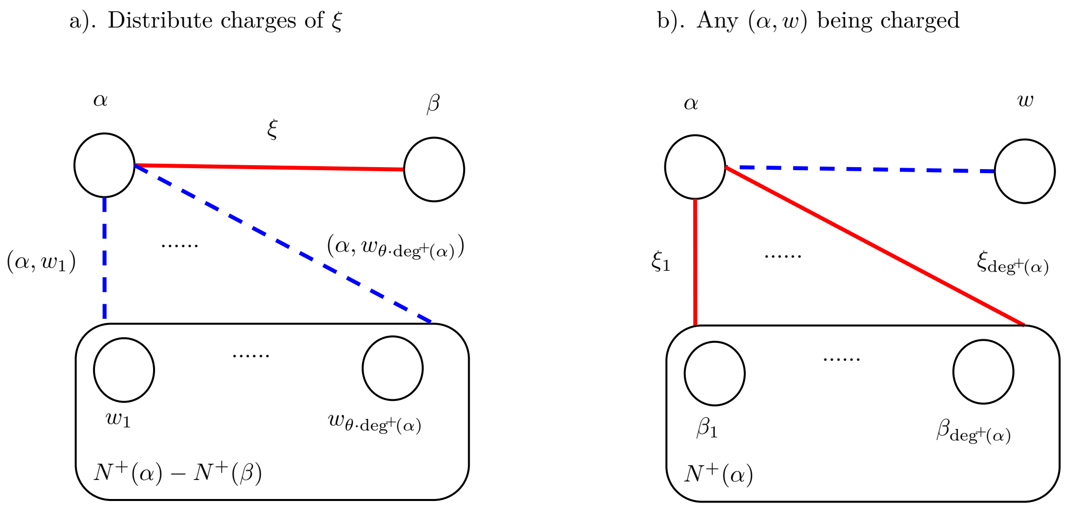
a). Distribute charges of ξ
α
ξ
β
(α, w_1)
......
$(\alpha, w_{\theta \cdot \deg^+(\alpha)})$
......
w_1
$w_{\theta \cdot \deg^+(\alpha)}$
$N^+(\alpha) - N^+(\beta)$
b). Any (α, w) being charged
α
w
ξ_1
......
$\xi_{\deg^+(\alpha)}$
......
β_1
$\beta_{\deg^+(\alpha)}$
$N^+(\alpha)$

Figure 2 Illustration of the bound of charges on each vertex-edge pair. Focus on pairs that include α, we assume w.log. that $\deg^+(\alpha) \geq \deg^+(\beta)$, and use $N^+(\alpha) - N^+(\beta)$ as a special case of $N^+(\alpha) \triangle N^+(\beta)$. There are two types of edges: edges that distribute charges (red solid lines) and edges that are charged (blue dashed lines). For each edge $\xi = (\alpha, \beta)$ that distributes charges, there are at least $\theta \cdot \deg^+(v)$ edges to be charged. Therefore, it suffices to distributed $\frac{1}{\theta \cdot \deg^+(v)}$ charges to each of the edges being charged. For each edge (α, w) to be charged, only edges indent to α can distribute a charge, which is at most $\deg^+(v)$ many. Therefore, every vertex-edge pair that include α is charged at most θ^{-1}.

Proof. Since we place each $v \in V_{\text{sparse}}$ in a separate cluster, $\mathsf{cost}_{\mathcal{A}}(e) = 0$ for every $e \in E^-(v)$. Hence, we can focus only on $e \in E^+(v)$ where $\mathsf{cost}_{\mathcal{A}}(e) = 1$.

Fix any vertex $v \in V_{\text{sparse}}$. By the guarantee of Theorem 5 for G^+, there is a subset $S(v)$ of $N^+(v)$ with size at least $|S(v)| \geq \eta_0 \cdot \varepsilon \cdot \deg^+(v)$ such that for every vertex $u \in S(v)$,

$$\left|N^+(v) \triangle N^+(u)\right| \geq \eta_0 \cdot \varepsilon \cdot \max\left\{\deg^+(u), \deg^+(v)\right\}.$$

Define the set of (+)-edges between v and the vertices in $S(v)$ as $E^+(v, S)$. By the size lower bound of $S(v)$, we can bound the charge of $\mathsf{sparse\text{-}charge}(z, f)$ by the charge of $E^+(v, S)$ with a multiplicative $O(1/\varepsilon)$ factor. Formally, let $\mathsf{sparse\text{-}S\text{-}charge}(z, f)$ denote the part of $\mathsf{sparse\text{-}charge}(z, f)$ contributed by $E^+(v, S)$. One can arbitrarily set the charge-set of every $1/(\varepsilon \cdot \eta_0)$ (+)-edges incident on v to be the same as a single edge in $E^+(v, S)$. Thus,

$$\text{for all vertex-edge pairs } (z, f)\text{:} \quad \mathsf{sparse\text{-}charge}(z, f) \leq \frac{1}{\varepsilon \cdot \eta_0} \cdot \mathsf{sparse\text{-}S\text{-}charge}(z, f). \qquad (4)$$

Therefore, it suffices to upper bound the charge contributed by $E^+(v, S)$. We construct the charge set and upper bound the charge by Lemma 8 as follows.

- We set $\mathcal{E}$ as the set of all edges between vertices $v \in V_{\text{sparse}}$ and $S(v) \subseteq N^+(v)$.
- For any $v \in V_{\text{sparse}}$ and $u \in S(v)$,

  $$\left|N^+(u) \triangle N^+(v)\right| \geq \eta_0 \cdot \varepsilon \cdot \max\left\{\deg^+(u), \deg^+(v)\right\},$$

  so we can set $\theta = \eta_0 \cdot \varepsilon$.

Therefore, by Lemma 8, there exists a charge-set for all edges in $\mathcal{E}$ such that for any vertex-edge pair (z, f), $\mathsf{sparse\text{-}S\text{-}charge}(z, f) = O(\varepsilon^{-1})$ (as η_0 is an absolute constant). Combining this with Eq (4) gives us the desired $O(\varepsilon^{-2})$ bound. ◀

Part II: Almost-Cliques

We now turn to the analysis of the charge contributed by almost-cliques. Here, there are two types of edges to consider: (+)-edges between different almost-cliques (intra cluster edges) and (−)-edges inside each single almost-clique (inter cluster edges); all remaining edges are already handled in the previous part. The following two lemmas handle these cases.

▶ **Lemma 10.** *There exists* ChargeSet(e) *for every* (+)*-edge between two different almost-cliques such that*

$$\text{for all vertex-edge pairs } (z, f)\text{:} \quad \mathsf{outside\text{-}clique\text{-}charge}(z, f) = O(1).$$

▶ **Lemma 11.** *There exists* ChargeSet(e) *for every* (−)*-edge inside any single almost-clique such that*

$$\text{for all vertex-edge pairs } (z, f)\text{:} \quad \mathsf{inside\text{-}clique\text{-}charge}(z, f) \leq 1.$$

We defer the proofs of Lemmas 10 and 11 to the full version, and give some high-level strategies here. The proof of Lemma 10 is similar to the one for sparse vertices in Lemma 9. We show that neighborhood of endpoints of (+)-intra cluster edges are very different, simply because they belong to different almost-cliques, and then apply Lemma 8. In contrast, the proof of Lemma 11 follows a different strategy: we show that the best clustering one can do for almost-cliques individually (in absence of all other edges) is to just cluster them one by one exactly as in $\mathcal{A}$.

In conclusion, the proof of Theorem 6 now follows by Lemma 9 (for handling all edges with at least one sparse endpoint) and Lemmas 10 and 11 (for handling all edges between dense vertices).

5 Sublinear Algorithms for Correlation Clustering

With the sparse-dense decomposition result from Section 3 and the correlation clustering scheme in Section 4 that built upon it, we can now describe our sublinear algorithms. We start with our sublinear-time algorithm.

▶ **Theorem 12** (Formalization of Theorem 1). *There exists a randomized algorithm that given a labeled graph $G = (V, E)$, specified via adjacency lists of its $(+)$-subgraph G^+, with high probability finds an $O(1)$-approximation to the correlation clustering problem on G in $O(n \log n)$ query and $O(n \log^2 n)$ time.*

We remind the reader about our discussion earlier in Section 2.2 on the necessity of access to the adjacency lists of G^+ as opposed to G in Theorem 12.

The second sublinear algorithm we present is a sublinear-space streaming algorithm.

▶ **Theorem 13** (Formalization of Theorem 2). *There exists a randomized single-pass semi-streaming algorithm that given a labeled graph $G = (V, E)$, specified via a stream of edges of G together with their labels, with high probability finds an $O(1)$-approximation to the correlation clustering problem on G in $O(n \log n)$ space. Moreover, the algorithm has $O(\log n)$ processing time per each element of the stream and $O(n \log^2 n)$ post-processing time.*

As we discussed in Section 2.2, this algorithm can be extended to various other streaming scenarios, such as when only edges of G^+ or G^- are streamed, or dynamic (insertion-deletion) and sliding window streams, by increasing the space with at most a $\text{polylog}\,(n)$ factor.

5.1 A Sublinear-Time Algorithm: Proof of Theorem 12

The algorithm is a direct implementation of the recovery algorithm of Theorem 5 for finding the decomposition plus the scheme of Theorem 6. We also need to show that we can provide the recovery algorithm of Theorem 5 with proper information it needs. This is done as follows.

Algorithm 1 A sublinear-time algorithm for correlation clustering.

- **Input:** A labeled graph $G = (V, E)$ specified via adjacency lists access to G^+.

(i) Let $\varepsilon > 0$ be a sufficiently small <u>constant</u> as prescribed by Theorem 5.
(ii) For each vertex $v \in V$, use degree queries to get positive degree $\deg^+(v)$ of v.
(iii) For each vertex v, use neighbor queries to sample $t = (c \cdot \log n)/\varepsilon^2$ neighbors of v from $N^+(v)$ with repetition to get $N_{\text{sample}}(v)$ (for the absolute constant $c > 0$ in Theorem 5).
(iv) Sample each vertex with probability $p_v := \min\left\{\frac{c \log(n)}{\deg^+(v)}, 1\right\}$ and call this set $\texttt{Sample}$. Use neighbor queries to get $N^+(v)$ for $v \in \texttt{Sample}$.
(v) Run the algorithm of Theorem 5 for sparse-dense decomposition with parameter ε and the inputs $\{N_{\text{sample}}(v)\}_{v \in V}$ and $\{N^+(v)\}_{v \in \texttt{Sample}}$ to its recovery algorithm.
(vi) Output clustering $\mathcal{A}$ based on the resulting $V_{\text{sparse}} \sqcup K_1 \sqcup \ldots \sqcup K_k$ as prescribed in Theorem 6.

We now prove the correctness and the bounds on query and time complexity of Algorithm 1.

Correctness. The information provided to the recovery algorithm of Theorem 5 by Algorithm 1 is exactly as prescribed in the theorem (for the underlying graph G^+). As such, with high probability, the resulting decomposition is a valid sparse-dense decomposition specified by Theorem 5. Conditioned on this event, by Theorem 6, the returned answer is an $O(1)$-approximation to the correlation clustering on G.

Query complexity. The total number of queries made by Algorithm 1 is equal to n degree queries plus the number of edge-samples and neighbors of all vertex-samples. By a simple analysis for the number of edges adjacent to the vertex-samples (which one can find in the full version), this is $O(n \log n)$ edges.

Runtime analysis. The runtime of Lines ii,iii, and iv is equal to the query complexity of the algorithm and is thus $O(n \log n)$ with high probability. The runtime of Item v is equal to the recovery algorithm of Line Theorem 5 which is $O(n \log^2 n)$ with high probability. The runtime of Line vi is equal to $O(n)$ as it only involves a direct partitioning of n vertices as specified in the statement of Theorem 6. This is $O(n \log^2 n)$ time in total.

This concludes the proof of Theorem 12.

5.2 A Semi-Streaming Algorithm: Proof of Theorem 13

We now give a single-pass semi-streaming algorithm for correlation clustering in insertion-only streams (over the edges of the input labeled graph G). For further extensions of this algorithm to other streaming models, please refer to the the full version of the paper.

Our semi-streaming algorithm is also a direct implementation of our recovery algorithm in Theorem 5 and the scheme of Theorem 6 (by focusing on edges of G^+ in the stream and simply skipping any edge of G^-). However, compared to the previous section, for this algorithm we have to be a bit careful in how we exactly provide the required information to the recovery algorithm of Theorem 5. This is primarily because, in a single pass over the stream, we will not know degrees of vertices beforehand so that we can sample the set `Sample` store all their neighbors appropriately. Nevertheless, we show that simple ideas in reservoir sampling [41] can be used to address this problem. Thus we first start by designing a subroutine for obtaining `Sample` and $N(v)$ for $v \in$ `Sample` and then show use to obtain our final semi-streaming algorithm.

Sampling vertices and storing their neighbors. We present the following lemma and algorithm for sampling vertices inversely proportional to their degree and storing all neighbors of sampled vertices (We note that we shall apply the following lemma to the underlying graph G^+). The idea behind this lemma seems standard to us and we present it here for completeness.

▶ **Lemma 14.** *Let $\beta_0 > 0$ be a sufficiently large constant. There is a semi-streaming algorithm that given any arbitrary graph $G = (V, E)$ (not necessarily a labeled graph) specified via a stream of its edges and a parameter $\beta > \beta_0$, at every point of time t during the stream:*

1. *Maintains a collection S_t of vertices together with $N_t(v)$ for all $v \in S_t$ so that each vertex is sampled independently and with probability $\min\{(\beta \cdot \log n)/\deg_t(v), 1\}$ in S_t (here, $N_t(v)$ and $\deg_t(v)$ refer to the set of neighbors of v and degree of v among the edges up to time t in the stream);*
2. *With high probability, uses space of $O(\beta \cdot n \log n)$ throughout the stream and $O(1)$ time per update.*

(We note that the independence guarantee of the algorithm is across the vertices and not time steps.)

Proof. The algorithm is as follows.

```
Sampling algorithm of Lemma 14.

(i) Let $S_1 = V$, $N_1(v) = \emptyset$, $\deg_1(v) = 0$ for $v \in V$.
(ii) For each arriving edge $e_t = (u_t, v_t)$:
    a. Update $N_t(w)$ and $\deg_t(w)$ for $w \in S_{t-1}$ by adding $u_t$ and $v_t$ to the neighborhood of respective vertices and increasing their degree and keeping other neighbor-sets intact.
    b. Update $S_t$ from $S_{t-1}$ by keeping all vertices other than $u_t$ and $v_t$ in $S_t$ and removing $z \in \{u_t, v_t\}$ from $S_t$ with probability $1/\deg_t(z)$ if $\deg_t(z) > (\beta \cdot \log n)$; if $z$ is removed from $S_t$ we also discard $N_t(z)$ from the memory.
```

Fix a vertex $v \in V$ and let $t_0(v)$ denote the first time step t such that $\deg_t(v) > (\beta \cdot \log n)$. For any time $t < t_0(v)$, we have v in S_t as it cannot be removed by the algorithm. For a time $t_1 \geq t_0(v)$, we have,

$$\begin{aligned}
\Pr(v \in S_{t_1}) &= \prod_{t=t_0}^{t_1} \Pr(v \in S_t \mid v \in S_{t-1}) \\
&= \prod_{t=t_0}^{t_1} (1 - \frac{1}{\deg_t(v)}) && \text{(as } v \text{ is removed from } S_t \text{ w.p. } 1/\deg_t(v)) \\
&= \frac{\deg_{t_0(v)} - 1}{\deg_t(v)} && \text{(by a simple cancelation of intermediate terms)} \\
&= \frac{(\beta \cdot \log n)}{\deg_t(v)}.
\end{aligned}$$

Thus, each vertex v belongs to S_t with probability $\min\{(\beta \log n)/\deg_t(v), 1\}$. Moreover, the choice of inclusion or exclusion of different vertices in S_t is independent, proving the first part of the lemma.

For the second part, given the correctness of the first, at each time step t, the information stored by the algorithm consists a random subset S_t of vertices where each vertex is included with probability $\min\{(\beta \log n)/\deg_t(v), 1\}$ plus all edges incident on vertices. For the graph at time t, we can define random variable X_v^t to be 0 if $v \notin S_t$ and to be $\deg(v)$ if $v \in S_t$. Let $X^t = \sum_{v \in V} X_v^t$ be the random variable for the number of edges we store at time step t. By a simple calculation, we can show that $\mathbb{E}[X^t] = O(\beta \cdot n \log n)$. Furthermore, by applying Bernstein's inequality (Proposition 4) on X^t, we can show that the total number of stored edges at step t is $O(\beta \cdot n \log n)$ with high probability. A union bound on at most $\binom{n}{2}$ steps concludes the proof (the bound of $O(1)$ on the update time is immediate). ◀

The semi-streaming algorithm. We can now present our semi-streaming algorithm for Theorem 13.

Correctness. The information provided to the recovery algorithm of Theorem 5 by Algorithm 1 is exactly as prescribed in the theorem (for the underlying graph G^+). As such, with high probability, the resulting decomposition is a valid sparse-dense decomposition specified by Theorem 5. Conditioned on this event, by Theorem 6, the returned answer is an $O(1)$-approximation to the correlation clustering on G.

Algorithm 2 A single-pass semi-streaming algorithm for correlation clustering.

- **Input:** A labeled graph $G = (V, E)$ specified via an arbitrarily ordered stream of edges E.

```
(i)   Let $\varepsilon > 0$ be a sufficiently small constant as prescribed by Theorem 5.
(ii)  For each vertex $v \in V$, use a counter over edges of $E^+(v)$ to maintain $\deg^+(v)$.
(iii) For each vertex $v$, use reservoir sampling to sample $t = (c \cdot \log n)/\varepsilon^2$ neighbors
      of $v$ from $N^+(v)$ with repetition to get $N_{\text{sample}}(v)$ (for the absolute constant
      $c > 0$ in Theorem 5).
(iv)  Run the algorithm of Lemma 14 on the graph $G^+$ with parameter $\beta = c$. Let
      Sample be the final set of vertices maintained by the algorithm and note we have
      $N^+(v)$ for $v \in$ Sample.
(v)   Run the algorithm of Theorem 5 for sparse-dense decomposition with parameter
      $\varepsilon$ and the inputs $\{N_{\text{sample}}(v)\}_{v \in V}$ and $\{N^+(v)\}_{v \in \texttt{Sample}}$ to its recovery algorithm.
(vi)  Output clustering $\mathcal{A}$ based on the resulting $V_{\text{sparse}} \sqcup K_1 \sqcup \ldots \sqcup K_k$ as prescribed
      in Theorem 6.
```

Space complexity. Line ii requires storing $O(n)$ numbers. Line iii requires storing $O(\log n)$ neighbors of each vertex for $O(n \log n)$ space in total. Line iv uses $O(n \log n)$ space with high probability by Lemma 14. The algorithms of Theorems 5 and 6 require space proportional to their input which is $O(n \log n)$ in total. Thus, overall space of the algorithm is $O(n \log n)$.

Update time and post-processing time. The update time is $O(1)$ for Lines ii and iv and $O(\log n)$ for Line iii. Thus, the update time is $O(\log n)$. The post-processing time is $O(n \log^2 n)$ time by Theorem 5.

This concludes the proof of Theorem 13.

References

1 Saba Ahmadi, Samir Khuller, and Barna Saha. Min-max correlation clustering via multicut. In Andrea Lodi and Viswanath Nagarajan, editors, *Integer Programming and Combinatorial Optimization - 20th International Conference, IPCO 2019, Ann Arbor, MI, USA, May 22-24, 2019, Proceedings*, volume 11480 of *Lecture Notes in Computer Science*, pages 13–26. Springer, 2019. `doi:10.1007/978-3-030-17953-3_2`.

2 Sara Ahmadian, Alessandro Epasto, Ravi Kumar, and Mohammad Mahdian. Fair correlation clustering. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 4195–4205. PMLR, 2020. URL: `http://proceedings.mlr.press/v108/ahmadian20a.html`.

3 Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation clustering in data streams. *Algorithmica*, 83(7):1980–2017, 2021. `doi:10.1007/s00453-021-00816-9`.

4 Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):23:1–23:27, 2008. `doi:10.1145/1411509.1411513`.

5 Noga Alon and Sepehr Assadi. Palette sparsification beyond (Δ+1) vertex coloring. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 6:1–6:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.APPROX/RANDOM.2020.6`.

6 Noga Alon and Joel H Spencer. *The probabilistic method.* John Wiley & Sons, 2004.

7 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta + 1)$ vertex coloring. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 767–786, 2019.

8 Sepehr Assadi and Shay Solomon. When algorithms for maximal independent set and maximal matching run in sublinear time. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, pages 17:1–17:17, 2019.

9 László Aszalós. Decompose boolean matrices with correlation clustering. *Entropy*, 23(7):852, 2021. doi:10.3390/e23070852.

10 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine learning*, 56(1):89–113, 2004.

11 Francesco Bonchi, David García-Soriano, and Konstantin Kutzkov. Local correlation clustering. *CoRR*, abs/1312.5105, 2013. arXiv:1312.5105.

12 Francesco Bonchi, Aristides Gionis, and Antti Ukkonen. Overlapping correlation clustering. *Knowl. Inf. Syst.*, 35(1):1–32, 2013. doi:10.1007/s10115-012-0522-9.

13 Marco Bressan, Nicolò Cesa-Bianchi, Andrea Paudice, and Fabio Vitale. Correlation clustering with adaptive similarity queries. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12510–12519, 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/b0ba5c44aaf65f6ca34cf116e6d82ebf-Abstract.html.

14 Mélanie Cambus, Davin Choo, Havu Miikonen, and Jara Uitto. Massively parallel correlation clustering in bounded arboricity graphs. In Seth Gilbert, editor, *35th International Symposium on Distributed Computing, DISC 2021, October 4-8, 2021, Freiburg, Germany (Virtual Conference)*, volume 209 of *LIPIcs*, pages 15:1–15:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.DISC.2021.15.

15 Yi-Jun Chang, Wenzheng Li, and Seth Pettie. An optimal distributed $(\Delta + 1)$-coloring algorithm? In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 445–456, 2018.

16 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *J. Comput. Syst. Sci.*, 71(3):360–383, 2005. doi:10.1016/j.jcss.2004.10.012.

17 Yudong Chen, Sujay Sanghavi, and Huan Xu. Clustering sparse graphs. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 2213–2221, 2012. URL: https://proceedings.neurips.cc/paper/2012/hash/1e6e0a04d20f50967c64dac2d639a577-Abstract.html.

18 Flavio Chierichetti, Nilesh N. Dalvi, and Ravi Kumar. Correlation clustering in mapreduce. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani, editors, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 641–650. ACM, 2014. doi:10.1145/2623330.2623743.

19 Flavio Chierichetti, Alessandro Panconesi, Giuseppe Re, and Luca Trevisan. Correlation clustering reconstruction in semi-adversarial models. *CoRR*, abs/2108.04729, 2021. arXiv:2108.04729.

20 William W. Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 475–480. ACM, 2002. doi:10.1145/775047.775116.

21 Vincent Cohen-Addad, Silvio Lattanzi, Slobodan Mitrovic, Ashkan Norouzi-Fard, Nikos Parotsidis, and Jakub Tarnawski. Correlation clustering in constant many parallel rounds. In

Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 2069–2078. PMLR, 2021. URL: `http://proceedings.mlr.press/v139/cohen-addad21b.html`.

22 Graham Cormode, Jacques Dark, and Christian Konrad. Independent sets in vertex-arrival streams. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 45:1–45:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPIcs.ICALP.2019.45`.

23 Dotan Emanuel and Amos Fiat. Correlation clustering - minimizing disagreements on arbitrary weighted graphs. In Giuseppe Di Battista and Uri Zwick, editors, *Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003, Proceedings*, volume 2832 of *Lecture Notes in Computer Science*, pages 208–220. Springer, 2003. `doi:10.1007/978-3-540-39658-1_21`.

24 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005.

25 Manuela Fischer and Andreas Noever. Tight analysis of parallel randomized greedy MIS. *ACM Trans. Algorithms*, 16(1):6:1–6:13, 2020. `doi:10.1145/3326165`.

26 Jurgen Van Gael and Xiaojin Zhu. Correlation clustering for crosslingual link detection. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 1744–1749, 2007. URL: `http://ijcai.org/Proceedings/07/Papers/282.pdf`.

27 David García-Soriano, Konstantin Kutzkov, Francesco Bonchi, and Charalampos E. Tsourakakis. Query-efficient correlation clustering. In Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen, editors, *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 1468–1478. ACM / IW3C2, 2020. `doi:10.1145/3366423.3380220`.

28 Magnús M. Halldórsson, Fabian Kuhn, Yannic Maus, and Tigran Tonoyan. Efficient randomized distributed coloring in CONGEST. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1180–1193. ACM, 2021.

29 David G Harris, Johannes Schneider, and Hsin-Hao Su. Distributed $(\Delta + 1)$-coloring in sublogarithmic rounds. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 465–478. ACM, 2016.

30 Jafar Jafarov, Sanchit Kalhan, Konstantin Makarychev, and Yury Makarychev. Local correlation clustering with asymmetric classification errors. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 4677–4686. PMLR, 2021. URL: `http://proceedings.mlr.press/v139/jafarov21a.html`.

31 Sungwoong Kim, Chang Dong Yoo, Sebastian Nowozin, and Pushmeet Kohli. Image segmentation usinghigher-order correlation clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(9):1761–1774, 2014. `doi:10.1109/TPAMI.2014.2303095`.

32 Michael Molloy and Bruce A. Reed. A bound on the total chromatic number. *Combinatorica*, 18(2):241–280, 1998.

33 Michael Molloy and Bruce A. Reed. Asymptotically optimal frugal colouring. *J. Comb. Theory, Ser. B*, 100(2):226–246, 2010.

34 Michael Molloy and Bruce A. Reed. Colouring graphs when the number of colours is almost the maximum degree. *J. Comb. Theory, Ser. B*, 109:134–195, 2014.

35 Bruce Reed. The list colouring constants. *Journal of Graph Theory*, 31(2):149–153, 1999.

36 Bruce A. Reed. ω, Δ, and χ. *Journal of Graph Theory*, 27(4):177–212, 1998.

37 Bruce A. Reed. A strengthening of Brooks' theorem. *J. Comb. Theory, Ser. B*, 76(2):136–149, 1999.

38 Jessica Shi, Laxman Dhulipala, David Eisenstat, Jakub Lacki, and Vahab S. Mirrokni. Scalable community detection via parallel correlation clustering. *Proc. VLDB Endow.*, 14(11):2305–2313, 2021. URL: `http://www.vldb.org/pvldb/vol14/p2305-shi.pdf`.

39 Chaitanya Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In J. Ian Munro, editor, *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 526–527. SIAM, 2004. URL: `http://dl.acm.org/citation.cfm?id=982792.982866`.

40 Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

41 Jeffrey Scott Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985. `doi:10.1145/3147.3165`.

42 Jordi R Weggemans, Alex Urech, Alexander Rausch, Robert Spreeuw, Richard Boucherie, Florian Schreck, Kareljan Schoutens, Jiří Minář, and Florian Speelman. Solving correlation clustering with qaoa and a rydberg qudit system: a full-stack approach. *arXiv preprint arXiv:2106.11672*, 2021.