Massively Parallel Correlation Clustering in Bounded Arboricity Graphs

Mélanie Cambus Aalto University, Finland
Davin Choo National University of Singapore, Singapore
Havu Miikonen Aalto University, Finland
Jara Uitto Aalto University, Finland

Abstract
Identifying clusters of similar elements in a set is a common task in data analysis. With the immense growth of data and physical limitations on single processor speed, it is necessary to find efficient parallel algorithms for clustering tasks. In this paper, we study the problem of correlation clustering in bounded arboricity graphs with respect to the Massively Parallel Computation (MPC) model. More specifically, we are given a complete graph where the edges are either positive or negative, indicating whether pairs of vertices are similar or dissimilar. The task is to partition the vertices into clusters with as few disagreements as possible. That is, we want to minimize the number of positive inter-cluster edges and negative intra-cluster edges.

Consider an input graph $G$ on $n$ vertices such that the positive edges induce a $\lambda$-arboricity graph. Our main result is a 3-approximation (in expectation) algorithm to correlation clustering that runs in $O(\log \lambda \cdot \text{poly}(\log \log n))$ MPC rounds in the strongly sublinear memory regime. This is obtained by combining structural properties of correlation clustering on bounded arboricity graphs with the insights of Fischer and Noever (SODA ’18) on randomized greedy MIS and the PIVOT algorithm of Ailon, Charikar, and Newman (STOC ’05). Combined with known graph matching algorithms, our structural property also implies an exact algorithm and algorithms with worst case $(1 + \varepsilon)$-approximation guarantees in the special case of forests, where $\lambda = 1$.

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

Keywords and phrases MPC Algorithm, Correlation Clustering, Bounded Arboricity

Introduction
Graphs are a versatile abstraction of datasets and clustering on graphs is a common unsupervised machine learning task for data-analytical purposes such as community detection and link prediction [13,11].

Here, we study the correlation clustering problem which aims at grouping elements of a dataset according to their similarities. Consider the setting where we are given a complete signed graph $G = (V, E = E^+ \cup E^-)$ where edges are given positive ($E^+$) or negative ($E^-$) labels, signifying whether two points are similar or not. The task is to find a partitioning of the vertex set $V$ into clusters $C_1, C_2, \ldots, C_r$, where $r$ is not fixed by the problem statement.
but can be chosen freely by the algorithm\(^1\) If endpoints of a positive edge belong to the same cluster, we say that the edge is a **positive agreement**; and a **positive disagreement** otherwise. Meanwhile, if endpoints of a negative edge belong to the same cluster, we say that the edge is a **negative disagreement**; and a **negative agreement** otherwise. The goal of correlation clustering is to obtain a clustering that maximizes agreements or minimizes disagreements.

As pointed out by Chierichetti, Dalvi and Kumar [15], the positive degrees of vertices are typically bounded in many applications. This motivates the study of parallel algorithms for correlation clustering as a function of the maximum degree of the input graph. However, many real life networks, such as those modelled by scale-free network models (such as Barabási-Albert), admit structures with a few high degree nodes and a small average degree. To capture such graphs, we generalize the study of bounded degree graphs to the study of low arboricity graphs in this work. In particular, we focus on the case of minimizing disagreements when the positive edges of the input graph induces a \(\lambda\)-arboric graph.

In the complete signed graph setting, one can perform cost-charging arguments via “bad triangles” to prove approximation guarantees. A set of 3 vertices \(\{u, v, w\}\) is a **bad triangle** if \(\{u, v\}, \{v, w\} \in E^+\) and \(\{u, w\} \in E^-\). As edges of any bad triangle induce at least one disagreement in any clustering, one can lower bound the cost of any optimum clustering by the number of edge-disjoint bad triangles in the input graph. **PIVOT** [2] is a well-known algorithm that provides a 3-approximation (in expectation) to the problem of minimizing disagreements in the sequential setting by using a cost-charging argument on bad triangles. It works as follows: as long as the graph is non-empty, pick a vertex \(v\) uniformly at random and form a new cluster using \(v\) and its “positive neighbors” (i.e. joined by a positive edge). One can view **PIVOT** as simulating greedy MIS with respect to a uniform-at-random permutation of vertices\(^2\).

Many of the known distributed algorithms for the correlation clustering problem adapt the **PIVOT** algorithm. The basic building block is to fix a random permutation and to create the clusters by finding, in parallel, local minimums according to the permutation. The **ParallelPIVOT**, **C4** and **ClusterWild!** algorithms [15, 33] all obtain constant approximations in \(O(\log n \cdot \log \Delta)\) synchronous rounds, where \(\Delta\) stands for the maximum positive degree. Meanwhile, with a tighter analysis of randomized greedy MIS algorithm [21], one can obtain a 3-approximation in \(O(\log n)\) rounds by directly simulating **PIVOT**. All above approximation guarantees are in expectation.

### 1.1 Computational model

We consider the **Massive Parallel Computation (MPC) model** [39, 8] which serves as a theoretical abstraction of several popular massively parallel computation frameworks such as Dryad [29], Hadoop [39], MapReduce [19], and Spark [37].

In the MPC model, we have \(M\) machines, each with memory size \(S\), and we wish to solve a problem given an input of size \(N\). In the context of correlation clustering, we may think of

\(^1\) This is in contrast to, for example, the classic \(k\)-means clustering where \(k\) is an input problem parameter.

\(^2\) A subset \(M \subseteq V\) is a maximal independent set (MIS) if (1) for any two vertices \(u, v \in M\), \(u\) and \(v\) are not neighbors, and (2) for any vertex \(v \in V\), either \(v \in M\) or \(v\) has a neighbor in \(M\). Given a vertex ordering \(\pi: [n] \to V\), greedy MIS refers to the process of iterating through \(\pi(1), \ldots, \pi(n)\) and adding each vertex to \(M\) if it has no neighbor of smaller ordering.

\(^3\) Technically speaking, **ParallelPIVOT** does not compute a greedy MIS. Instead, it computes random independent sets in each phase and only uses the initial random ordering to perform tie-breaking, i.e. if a vertex \(u\) has more than one positive neighbor in the independent set, then vertex \(u\) joins the cluster defined by the neighbor with the smallest assigned order.
M. Cambus, D. Choo, H. Miikonen and J. Uitto

Although there is no hard computation constraint in the MPC model, all known MPC algorithms spend polynomial time on each machine in any given round.

In some works, “bounded degree” is synonymous with “maximum degree \( O(1) \).” Here, we mean that the maximum degree is \( O(\lambda) \).

---

1.2 Our contributions

Our goal is to obtain efficient algorithms for correlation clustering in the sublinear memory regime of MPC (see Model 1) when given a complete signed graph \( G \), with maximum positive degree \( \Delta \), where the set of positive edges \( E^+ \) induces a \( \lambda \)-arboric graph. Our main contributions are the following:

1. By combining known techniques, we show that one can compute a randomized greedy MIS, with respect to a uniform-at-random permutation of vertices, in \( O(\log \Delta \cdot \log^2 \log n) \) MPC rounds. If we allow extra global memory (see Model 2), this can be sped up to \( O(\log \Delta \cdot \log \log n) \) MPC rounds. See Theorem 2 for details.

We believe that this result is of independent interest beyond applications to correlation clustering. To the best of our knowledge, our algorithm for greedy MIS improves upon the state-of-the-art for any large than \( O(\log \log \cdot \log \log n) \).

2. Our main result (Theorem 12) is that one can effectively ignore vertices of degrees larger than \( O(\lambda) \) when computing a correlation clustering. Then, the overall runtime and approximation guarantees are inherited from the choice of algorithm used to solve correlation clustering on the remaining bounded degree subgraph.

3. Using our main result, we show how to obtain efficient correlation clustering algorithms for bounded arboricity graphs. By simulating PIVOT on a graph with maximum degree \( O(\lambda) \) via Theorem 6, we get

(i) A 3-approximation (in expectation) algorithm in \( O(\log \lambda \cdot \log^3 \log n) \) MPC rounds.

(ii) A 3-approximation (in expectation) algorithm in \( O(\log \lambda \cdot \log \log n) \) MPC rounds, possibly using extra global memory.

In the special case of forests (where \( \lambda = 1 \)), we show that the optimum correlation clustering is equivalent to computing a maximum matching. Let \( 0 < \epsilon \leq 1 \) be a constant. By invoking three different known algorithms (one for maximum matching and two for maximal matching), and hiding \( 1/\epsilon \) factors in \( O_\epsilon(\cdot) \), we obtain

(iii) An exact randomized algorithm that runs in \( \tilde{O}(\log n) \) MPC rounds.

(iv) A \( (1 + \epsilon) \)-approx. (worst case) det. algo. that runs in \( O_\epsilon(\log \log^* n) \) MPC rounds.

(v) A \( (1 + \epsilon) \)-approx. (worst case) randomized algo. that runs in \( O_\epsilon(1) \) MPC rounds.

Finally, for low-arboricity graphs, the following result may be of interest:

(vi) An \( O(\lambda^2) \)-approx. (worst case) deterministic algo. that runs in \( O(1) \) MPC rounds.

For more details and an in-depth discussion about our techniques, see Section 2.
1.3 Outline and notation

1.3.1 Outline

Before diving into formal details, we highlight the key ideas behind our results in Section 2. Section 3 shows how to efficiently compute a randomized greedy MIS. Our structural result about correlation clustering in bounded arboricity graphs is presented in Section 4. We combine this structural insight with known algorithms in Section 5 to yield efficient correlation clustering algorithms. Finally, we conclude with some open questions in Section 6.

Due to space constraints, we will defer some proof details to Appendix A.

1.3.2 Notation

In this work, we only deal with complete signed graphs \( G = (V, E = E^+ \cup E^-) \) where \( |V| = n, |E| = \binom{n}{2} \), and \( E^+ \) and \( E^- \) denote the sets of positively and negatively labeled edges respectively. For a vertex \( v \), the sets \( N^+(v) \subseteq V \) and \( N^-(v) \subseteq V \) denote vertices that are connected to \( v \) via positive and negative edges, respectively. We write \( \Delta = \max_{v \in V} |N^+(v)| \) as the maximum positive degree in the graph. The \( k \)-hop neighborhood of a vertex \( v \) is the set of vertices that have a path from \( v \) involving at most \( k \) positive edges.

A clustering \( C \) is a partition of the vertex set \( V \). That is, \( C \) is a set of sets of vertices such that (i) \( A \cap B = \emptyset \) for any two sets \( A, B \in C \) and (ii) \( \cup_{A \in C} A = V \). For a cluster \( C \subseteq V \), \( N^+_C(v) = N^+(v) \cap C \) is the set of positive neighbors of \( v \) that lie within cluster \( C \). We write \( d^+_C(v) = |N^+(v) \cap C| \) to denote the positive degree of \( v \) within \( C \). If endpoints of a positive edge do not belong to the same cluster, we say that the edge is a positive disagreement. Meanwhile, if endpoints of a negative edge belong to the same cluster, we say that the edge is a negative disagreement. Given a clustering \( C \), the cost of a clustering \( \text{cost}(C) \) is defined as the total number of disagreements.

The arboricity \( \lambda_G \) of a graph \( G = (V, E) \) is defined as \( \lambda_G = \max_{S \subseteq V} \left[ \frac{|E(S)|}{|S| - 1} \right] \), where \( E(S) \) is the set of edges induced by \( S \subseteq V \). We drop the subscript \( G \) when it is clear from context. A graph with arboricity \( \lambda \) is said to be \( \lambda \)-arboric. We denote the set \( \{1, 2, \ldots, n\} \) by \( [n] \). We hide absolute constant multiplicative factors and multiplicative factors logarithmic in \( n \) using standard notations: \( \Theta(\cdot), \Omega(\cdot), \) and \( \tilde{\Omega}(\cdot) \). The notation \( \log^* n \) refers to the smallest integer \( t \) such that the \( t \)-iterated logarithm of \( n \) is at most \( 1 \).\footnote{That is, \( \log^{(t)} n \leq 1 \). For all practical values of \( n \), one may treat \( \log^* n \leq 5 \).} An event \( E \) on a \( n \)-vertex graph holds with high probability if it happens with probability at least \( 1 - n^{-c} \) for an arbitrary constant \( c > 1 \), where \( c \) may affect other constants (e.g. those hidden in the asymptotics).

We now fix the parameters in our model of computation. Model 1 is the standard definition of strongly sublinear MPC regime while Model 2 is a relaxed variant which guarantees that there are at least \( M \geq n \) machines. While the latter model may utilize more global memory than the standard strongly sublinear regime, it facilitates conceptually simpler algorithms.

\begin{itemize}
  \item \textbf{Model 1 (Strongly sublinear MPC regime).} Consider the MPC model. The input graph with \( n \) vertices is of size \( N \in \Omega(n) \). We have \( M \in \Omega(N/S) \) machines, each having memory size \( S \in \tilde{\Theta}(n^\delta) \), for some constant \( 0 < \delta < 1 \). The total global memory usage is \( M \cdot S \geq N \).
  \item \textbf{Model 2 (Strongly sublinear MPC regime with at least \( n \) machines).} Consider the MPC model. The input graph with \( n \) vertices is of size \( N \in \Omega(n) \). We have \( M \geq n \) machines and each vertex is given access to a machine with memory size \( S \in \tilde{\Theta}(n^\delta) \), for some constant \( 0 < \delta < 1 \). The total global memory usage is \( \max\{\tilde{\Theta}(n^{1+\delta}) , M \cdot S\} \geq N \).
\end{itemize}
To avoid unnecessary technical complications for Model 2, we assume throughout the paper that $\Delta \in \mathcal{O}(S)$. This assumption can be lifted using the virtual communication tree technique described by Ghaffari and Uitto [26].

**Remark 3 (Role and motivation for Model 2).** From an algorithmic design perspective, the slightly relaxed Model 2 allows one to focus on keeping the amount of “local memory required by each vertex” to the sublinear memory regime. Oftentimes algorithms are first described in relaxed models (such as Model 2) or by simply allowing more total global memory used with a simple-to-understand analysis before using further complicated argument/analysis to show that it in fact also works in Model 1.

### 1.4 Further related work

Correlation clustering on complete signed graphs was introduced by Bansal, Blum and Chawla [4]. They showed that computing the optimal solution to correlation clustering is NP-complete, and explored two different optimization problems: maximizing agreements, or minimizing disagreements. While the optimum clusterings to both problems are the same (i.e. a clustering minimizes disagreements if and only if it maximizes agreements), the complexity landscapes of their approximate versions are wildly different.

Maximizing agreements is known to admit a polynomial time approximation scheme in complete graphs [4]. Furthermore, Swamy [35] gave a 0.7666-approximation on general weighted graphs via semidefinite programming.

On the other hand, for minimizing disagreements, the best known approximation ratio for complete graphs is 2.06, due to CMSY [13], via probabilistic rounding of a linear program (LP) solution. This 2.06-approximation uses the same LP as the one proposed by Ailon, Charikar and Newman [2] but performs probabilistic rounding more carefully, nearly matching the integrality gap of 2 shown by Charikar, Guruswami and Wirth [12]. In general weighted graphs, the current state of the art, due to DEFI [19], gives an $\mathcal{O}(\log n)$-approximation through an LP rounding scheme.

In a distributed setting, PPORRJ [33] presented two random algorithms (C4 and ClusterWild!) to address the correlation clustering problem in the case of complete graphs, aiming at better time complexities than KwikCluster. The C4 algorithm gives a 3-approximation in expectation, with a polylogarithmic number of rounds where the greedy MIS problem is solved on each round. The ClusterWild! algorithm gives up on the independence property in order to speed up the process, resulting in a $(3 + \varepsilon)$-approximation. Both those algorithms are proven to terminate after $\mathcal{O}\left(\frac{1}{\varepsilon} \cdot \log n \cdot \log \Delta\right)$ rounds with high probability. A third distributed algorithm for solving correlation clustering is given by Chierichetti, Dalvi and Kumar [15] for the MapReduce model. Their algorithm, ParallelPivot, also gives...
Massively Parallel Correlation Clustering in Bounded Arboricity Graphs

a constant approximation in polylogarithmic time, without solving a greedy MIS in each round. Using a tighter analysis, Fischer and Noever [21] showed that randomized greedy MIS terminates in $O(\log n)$ round with high probability, which directly implies an $O(\log n)$ round simulation of PIVOT in various distributed computation models.

For our approach, the randomized greedy MIS plays a crucial role in terms of the approximation ratio. Blelloch, Fineman and Shun [10] showed that randomized greedy MIS terminates in $O(\log^2 n)$ parallel rounds with high probability. This was later improved to $O(\log n)$ rounds by Fischer and Noever [21]. Faster algorithms are known for finding an MIS that may not satisfy the greedy property. For example, Ghaffari and Uitto [26] showed that there is an MIS algorithm running in $O(\sqrt{\log \Delta \cdot \log \log n})$ MPC rounds. This algorithm was later adapted to bounded arboricity with runtime of $O(\sqrt{\log \lambda \cdot \log \log \lambda + \log \log n})$ by BBDFHKU [9] and improved to $O(\sqrt{\log \lambda \cdot \log \log \lambda + \log \log n})$ by Ghaffari, Grunau and Jin [24]. There is also a deterministic MIS algorithm that runs in $O(\log \Delta + \log \log n)$ MPC rounds due to Czumaj, Davies and Parter [16].

2 Techniques

In this section, we highlight the key ideas needed to obtain our results described in Section 1.2. We begin by explaining some computational features of the MPC model so as to set up the context needed to appreciate our algorithmic results. By exploiting these computational features together with a structural result of randomized greedy MIS by Fischer and Noever [21], we explain how to compute a randomized greedy MIS in $O\left(\sqrt{\log \Delta \cdot \log \log n}\right)$ MPC rounds. We conclude this section by explaining how to obtain our correlation clustering results by using our structural lemma that reduces the maximum degree of the input graph to $O(\lambda)$.

2.1 Computational features of MPC

2.1.1 Detour: The classical models of LOCAL and CONGEST

To better appreciate of the computational features of MPC, we first describe the classical distributed computational models of LOCAL and CONGEST [32, 34].

In the LOCAL model, all vertices are treated as individual computation nodes and are given a unique identifier – some binary string of length $O(\log n)$. Computation occurs in synchronous rounds where each vertex does the following: perform arbitrary local computations, then send messages (of unbounded size) to neighbors. As the LOCAL model does not impose any restrictions on computation or communication costs (beyond a topological restriction), the performance of LOCAL algorithms is measured in the number of rounds used. Furthermore, since nodes can send unbounded messages, every vertex can learn about its $k$-hop neighborhood in $k$ LOCAL rounds.

The CONGEST model is identical to the LOCAL model with an additional restriction: the size of messages that can be sent or received per round can only be $O(\log n)$ bits across each edge. This means that CONGEST algorithms may no longer assume that they can learn about the $k$-hop topology for every vertex in $k$ CONGEST rounds.

Since the MPC model does not restrict computation within a machine, one can directly simulate any $k$-round LOCAL or CONGEST algorithm in $O(k)$ MPC rounds, as long as each machine sends and receives messages of size at most $O(S)$. This often allows us to directly invoke existing LOCAL and CONGEST algorithms in a black-box fashion.
2.1.2 Round compression

First introduced by CLMMOSP [17], the goal of round compression is to simulate multiple rounds of an iterative algorithm within a single MPC round. To do so, one gathers “sufficient amount of information” into a single machine. For example, if an iterative algorithm $A$ only needs to know the $k$-hop neighborhood to perform $r$ steps of an algorithm, then these $r$ steps can be compressed into a single MPC round once the $k$-hop neighborhood has been gathered.

2.1.3 Graph exponentiation

One way to speed up computation in an all-to-all communication setting (such as MPC) is the well-known graph exponentiation technique of Lenzen and Wattenhofer [31]. The idea is as follows: Suppose each vertex is currently aware of its $2^{k-1}$-hop neighborhood, then by sending this $2^{k-1}$ topology to all their current neighbors, each vertex learns about their respective $2^k$-hop neighborhoods in one additional MPC round. In other words, every vertex can learn about its $k$-hop neighborhood in $O(\log k)$ MPC rounds, as long as the machine memory is large enough. See Figure 1 for an illustration. This technique is motivated by the fact that once a vertex has gathered its $k$-hop neighborhood, it can execute any LOCAL algorithm that runs in $k$ rounds in just a single MPC round.

![Figure 1](after_round_k.png) After round $k$, vertex $u$ knows the graph topology within its $2^k$-hop neighborhood.

2.1.4 Combining graph exponentiation with round compression

Suppose we wish to execute a $k$-round LOCAL algorithm but the machine memory of a single machine is too small to contain entire $k$-hop neighborhoods. To get around this, one can combine graph exponentiation with round compression:

1. All vertices collect the largest possible neighborhood using graph exponentiation.
2. Suppose $\ell$-hop neighborhoods were collected, for some $\ell < k$. All vertices simulate $\ell$ steps of the LOCAL algorithm in a single MPC round using round compression.
3. All vertices update their neighbors about the status of their computation.
4. Repeat steps 2-3 for $O(k/\ell)$ phases.

This essentially creates a virtual communication graph where vertices are connected to their $\ell$-hop neighborhoods. This allows a vertex to derive, in one round of MPC, all the messages that reaches it in the next $\ell$ rounds of message passing. Using one more MPC round and the fact that local computation is unbounded, a vertex can inform all its neighbors in the virtual graph about its current state in the simulated message passing algorithm. See Figure 2.

Remark 4. In Section 2.1.4, we make the implicit assumption that the states of the vertices are small and hence can be communicated with small messages. In many algorithms (e.g,
Vertices gather ℓ-hop neighborhood via graph exponentiation

Figure 2 Suppose ℓ = 2. After each vertex collects their ℓ-hop neighborhood, computation within each collected neighborhood can be performed in a single compressed MPC round. While the vertices u and v were originally 8 hops apart, they can communicate in 2 MPC rounds through vertex w’s collected neighborhood in the virtual communication graph. Observe that this virtual communication graph has a smaller effective diameter compared to the original input graph.

for solving MIS, matching, coloring), including ours, the vertices maintain very small states. Hence, we omit discussion of individual message sizes in the scope of this paper.

2.1.5 Broadcast / Convergecast trees

Broadcast trees are a useful MPC data structure introduced by Goodrich, Sitchinava and Zhang [27] that allow us to perform certain aggregation tasks in $O(1/\delta)$ MPC rounds, which is essentially $O(1)$ for constant $\delta$. Suppose we have $O(N)$ global memory and $S = O(n^\delta)$ local memory. We build an $S$-ary virtual communication tree over the machines. That is, within one MPC round, the parent machine can send $O(1)$ numbers to each of its $S$ children machines, or collect one number from each of its $S$ children machines. In $O(\log_S N) \subseteq O(1/\delta)$ rounds, for all vertices $v$ in parallel, one can:

- broadcast a message from $v$ to all neighboring vertices in $N(v)$;
- compute $f(N(v))$, the value of a distributive aggregate function $f$ on set of vertices $N(v)$.

An example of such a function $f$ is computing the sum/min/max of numbers that were originally distributed across all machines. We use broadcast trees in the MPC implementation of the algorithm described in Corollary 32.

2.2 Randomized greedy MIS on bounded degree graphs

The following result of Fischer and Noever [21] states that each vertex only needs the ordering of the vertices within its $O(\log n)$-hop neighborhood in order to compute its own output status within a randomized greedy MIS run.

▶ **Theorem 5** (Fischer and Noever [21]). Given a uniform-at-random ordering of vertices, with high probability, the MIS status of any vertex is determined by the vertex orderings within its $O(\log n)$-hop neighborhood.

10 We borrow some notation from Ghaffari and Nowicki [25, Lemma 3.5]. For $n$-vertex graphs, $N \in O(n^2)$.
11 More specifically, they analyzed the “longest length of a dependency path” and showed that it is $O(\log n)$ with high probability, which implies Theorem 5.
Let \( \pi : [n] \rightarrow V \) be a uniform-at-random ordering of vertices and \( G \) be a graph with maximum degree \( \Delta \). In Section 3, we show that one can compute greedy MIS (with respect to \( \pi \)) in \( \mathcal{O}(\log \Delta \cdot \log \log n) \) MPC rounds.

\textbf{Theorem 6 (Randomized greedy MIS (Informal)).} Let \( G \) be a graph with maximum degree \( \Delta \). Then, randomized greedy MIS can be computed in \( \mathcal{O}(\log \Delta \cdot \log \log n) \) MPC rounds in Model 1, or in \( \mathcal{O}(\log \Delta \cdot \log \log n) \) MPC rounds in Model 2.

Algorithm 1 works in phases. In each phase, we process a prefix graph \( G_{\text{prefix}} \) defined by vertices indexed by a prefix of \( \pi \), where the maximum degree is \( \mathcal{O}(\log n) \) by Chernoff bounds. Algorithms 2 and Algorithm 3 are two subroutines to process prefix graph \( G_{\text{prefix}} \). The latter subroutine is faster by a \( \log^2 \log n \) factor but assumes access to more machines.

For a sufficiently large prefix of \( \pi \), the maximum degree of the input graph after processing \( G_{\text{prefix}} \) drops to \( \Delta/2 \) with high probability. This concludes a phase. Since the maximum degree in the original graph is halved, we can process more vertices in subsequent phases and thus process all \( n \) vertices after \( \mathcal{O}(\log \Delta) \) phases. See Figure 3 for an illustration.

\textbf{Remark 7 (Discussion about maximum degree).} We implicitly assume that \( \Delta > 1 \), which can be checked in \( \mathcal{O}(1) \) rounds. Otherwise, when \( \Delta = 1 \), the graph only contain pairs of vertices and isolated vertices and greedy MIS can be trivially simulated in one round.

\textbf{Algorithm 1} Greedy MIS in sublinear memory regime of the MPC model

1: \textbf{Input:} Graph \( G = (V, E) \) with maximum degree \( \Delta \)
2: Let \( \pi : [n] \rightarrow V \) be an ordering of vertices chosen uniformly at random.
3: for \( i = 0, 1, 2, \ldots, \mathcal{O}(\log \Delta) \) do \( \triangleright \mathcal{O}(\log \Delta) \) phases, or until \( G \) is empty
   4: \hspace{1em} Let prefix size \( t_i = \mathcal{O}\left(\frac{n \log n}{\Delta \cdot \Delta'}\right) \) and prefix offset \( o_i = \sum_{z=0}^{i-1} t_z \).
   5: \hspace{1em} \( G_i \leftarrow \) Prefix graph induced by vertices \( \pi(o_i + 1), \ldots, \pi(o_i + t_i) \) with max. degree \( \Delta' \).
6: \hspace{1em} Process \( G_i \) using Algorithms 2 or Algorithm 3 \( \triangleright \) By Chernoff bounds, \( \Delta' \in \mathcal{O}(\log n) \)
7: end for
8: Process any remaining vertices in \( G \) using additional \( \mathcal{O}(\log \log n) \) MPC rounds.

\textbf{Algorithm 2} Greedy MIS on \( n \)-vertex graph in \( \mathcal{O}(\log^2 \Delta \cdot \log \log n) \) MPC rounds in Model 2

1: \textbf{Input:} Vertex ordering \( \pi \), graph \( G \) on \( n \) vertices with maximum degree \( \Delta \)
2: for \( i = 0, 1, 2, \ldots, \lceil \log_2 \Delta \rceil \) do \( \triangleright \mathcal{O}(\log \Delta) \) phases, or until \( G \) is empty
   3: \hspace{1em} Let chunk size \( c_i = \frac{n \log n}{\Delta \cdot \log \Delta} \cdot n \). \( \triangleright \) Chunk size doubles per phase
   4: \hspace{1em} for \( j = 1, 2, \ldots, 2000 \log \Delta \) do \( \triangleright \mathcal{O}(\log \Delta) \) iterations, or until \( G \) is empty
      5: \hspace{2em} Let offset \( o_{i,j} = c_i \cdot (j - 1) + \sum_{z=0}^{i-1} c_z \cdot 2000 \log \Delta \).
      6: \hspace{2em} Let chunk graph \( G_{i,j} \) be the graph induced by vertices \( \pi(o_{i,j}), \ldots, \pi(o_{i,j} + c_i) \).
    7: \hspace{1em} Process chunk graph \( G_{i,j} \).
8: end for
9: end for

\textbf{Remark 8 (Discussion about Algorithm 2).} Our algorithm is inspired by the idea of graph shattering introduced by BEPS [6]. We break up the simulation of greedy MIS on \( G_{\text{prefix}} \) into \( \mathcal{O}(\log \Delta) \) phases that process chunks of increasing size within the prefix graph. By performing \( \mathcal{O}(\log \Delta) \) iterations within a phase, we can argue that any vertex in the remaining prefix graph has “low degree” with high probability in \( \Delta \). This allows us to prove that the connected components while processing every chunk of vertices is at most \( \mathcal{O}(\log n) \) and
each vertex can learn (within the global memory limits) about the topology of its connected
component in $O(\log \log n)$ rounds via graph exponentiation. The constants 100 and 2000 are
chosen for a cleaner analysis. While the algorithm indexes more than $n$ vertices, we simply
terminate after processing the last vertex in the permutation.\footnote{\footnotemark[12]}

Algorithm 3 Greedy MIS on $n$-vertex graph in $O(\log \log n + \log \Delta)$ MPC rounds in Model \footnote{\footnotemark[12]}
1: Input: Vertex ordering $\pi$, graph $G$ on $n$ vertices with maximum degree $\Delta$
2: Assign a machine to each vertex.  \textbf{In Model} \footnote{\footnotemark[12]} we have $\geq n$ machines.
3: Graph exponentiate and gather $R$-hop neighborhood, where $R \in O\left(\frac{\log n}{\log \Delta}\right)$.
4: Simulate greedy MIS (with respect to $\pi$) in $O(\log \Delta)$ MPC rounds.

\textbf{Remark 9 (Discussion about Algorithm 3).} We know from Theorem 5 that it suffices for
each vertex know its $O(\log n)$-hop neighborhood in order to determine whether it is in the
greedy MIS. However, the $O(\log n)$-hop neighborhoods may not fit in a single machine. So,
we use Section 2.1.4 to obtain a running time of $O\left(\log R + \frac{\log n}{\log \Delta}\right) \subseteq O(\log \log n + \log \Delta)$.

\textbf{Remark 10 (Comparison with the work of Blelloch, Fineman and Shun (BFS) \cite{10}).} The
algorithm of BFS \cite{10} also considered prefixes of increasing size and they have a similar
lemma as our Lemma 22. However, their work does not immediately imply ours. The focus
of BFS \cite{10} was in the PRAM model in which the goal is to obtain an algorithm that is
small work-depth — they gave implementation of their algorithms that does a linear amount
of work with polylogarithmic depth. In this work, we are interested in studying the MPC
model, in particular the sublinear memory regime. Directly simulating their algorithm in
MPC yields an algorithm that runs in $O(\log \Delta \cdot \log n)$ rounds. Here, we crucially exploit
graph exponentiation and round compression to speed up the greedy MIS simulation prefix
graphs, enabling us to obtain algorithms that have an exponentially better dependency on $n$,
i.e. that run in $O(\log \Delta \cdot \poly(\log \log n))$ rounds.

2.3 Correlation clustering on bounded arboricity graphs

Our algorithmic results for correlation clustering derive from the following key structural
lemma that is proven by arguing that a local improvement to the clustering cost is possible
if there exists large clusters.

\textbf{Lemma 11 (Structural lemma for correlation clustering (Informal)).} There exists an optimum
correlation clustering where all clusters have size at most $4\lambda - 2$.

This structural lemma allows us to perform cost-charging arguments against some optimum clustering with bounded cluster sizes. In particular, if a vertex has degree much larger than $\lambda$, then many of its incident edges incur disagreements. This insight yields the following algorithmic implication: we can effectively ignore high-degree vertices.

\textbf{Theorem 12 (Algorithmic implication (Informal)).} Let $G$ be a graph where $E^+$ induces
a $\lambda$-arboric graph. Form singleton clusters with vertices with degrees $O(\lambda/\epsilon)$. Run an
$\alpha$-approximate algorithm $A$ on the remaining subgraph. Then, the union of clusters is a
max\{1 + $\epsilon$, $\alpha$\}-approximation. The runtime and approximation guarantees of the overall
algorithm follows from the guarantees of $A$ (e.g. in expectation / worst case, det. / rand.).
Observe that PIVOT essentially simulates a randomized greedy MIS with respect to a uniform-at-random ordering of vertices. By setting $\varepsilon = 2$ in Theorem 12 and $\Delta = O(\lambda)$ in Theorem 6, we immediately obtain a 3-approximation (in expectation) algorithm for correlation clustering in $O(\log \lambda \cdot \log \log n)$ MPC rounds. Note that we always have $\lambda \leq \Delta \leq n$, and that $\lambda$ can be significantly smaller than $\Delta$ and $n$ in general. Many sparse graphs have $\lambda \in O(1)$ while having unbounded maximum degrees, including planar graphs and bounded treewidth graphs. As such, for several classes of graphs, our result improves over directly simulating PIVOT in $O(\log n)$ rounds.

**Figure 3** Illustration of Algorithm 1 given an initial graph $G$ on $n$ vertices with maximum degree $\Delta$. Let $i \in \{1, \ldots, O(\log \Delta)\}$ and define $t_i = O\left(\frac{n \log n}{\Delta^i} \right)$. For each $i$, with high probability, the induced subgraph $G_i$ has maximum degree $\text{poly}(\log n)$. To process $G_i$, apply Algorithm 2 in $O(\log \Delta \cdot \log \log n)$ MPC rounds, or Algorithm 3 in $O(\log \log n)$ MPC rounds while using extra global memory. By our choice of $t_i$, Lemma 22 tells us that remaining subgraph $H_i$ has maximum degree $\Delta / 2^i$. We repeat this argument until the final subgraph $H_{\text{final}}$ involving $\text{poly}(\log n)$ vertices, which can be processed in another call to Algorithm 2 or Algorithm 3.

**Corollary 13** (General algorithm (Informal)). Let $G$ be a complete signed graph such that $E^+$ induces a $\lambda$-arboric graph. There exists an algorithm that, with high probability, produces a 3-approximation (in expectation) for correlation clustering of $G$ in $O(\log \lambda \cdot \log \log n)$ MPC rounds in Model 1, or $O(\log \lambda \cdot \log \log n)$ MPC rounds in Model 2.

**Remark 14** (On converting “in expectation” to “with high probability”). Note that one can run $O(\log n)$ copies of Corollary 13 in parallel and output the best clustering. Applying this standard trick converts the “in expectation” guarantee to a “with high probability” guarantee with only a logarithmic factor increase in memory consumption.

For forests with $\lambda = 1$, Lemma 11 states that the optimum correlation clustering cost corresponds to the number of edges minus the size of the maximum matching. Instead of computing a maximum matching, Lemma 15 tells us that using an approximate matching suffices to obtain an $\alpha$-approximation (not necessarily maximal) to the correlation clustering problem. Note that maximal matchings are 2-approximations and they always apply.
Lemma 15 (Approximation via approximate matchings (Informal)). Let $G$ be a complete signed graph such that $E^+$ induces a forest. Suppose that the maximum matching size on $E^+$ is $|M^*|$. If $M$ is a matching on $E^+$ such that $\alpha \cdot |M| \geq |M^*|$, for some $1 \leq \alpha \leq 2$, then clustering using $M$ yields an $\alpha$-approximation to the optimum correlation clustering of $G$.

Thus, it suffices to apply known maximum/approximate matching algorithms in sublinear memory regime of MPC to obtain correlation clustering algorithms in the special case of $\lambda = 1$. More specifically, we consider the following results.

- Using dynamic programming, BBDHM [7] compute a maximum matching (on trees) in $O(\log n)$ MPC rounds.
- In the LOCAL model, EMR [20] deterministically solve $(1 + \varepsilon)$-approx. matching in $O\left(\Delta^2 / \varepsilon^3 \cdot \log^* n\right)$ rounds.
- In the CONGEST model, BCGS [5] give an $O\left(2^{O(1/\varepsilon)} \cdot \frac{\log \Delta}{\log \log \Delta}\right)$ round randomized algorithm for $(1 + \varepsilon)$-approx. matching.

These approximation results are heavily based on the Hopcroft-Karp framework [28], where independent sets of augmenting paths are iteratively flipped. Since $\lambda = 1$ and $\varepsilon$ is a constant, we have a subgraph of constant maximum degree by ignoring vertices with degrees $O(\lambda/\varepsilon)$.

On this constant degree graph, each vertex only needs polylogarithmic memory when we perform graph exponentiation, satisfying the memory constraints of Model 1. Applying these matching algorithms together with Theorem 12 and Lemma 15 yields the following result.

Corollary 16 (Forest algorithm (Informal)). Let $G$ be a complete signed graph such that $E^+$ induces a forest and $0 < \varepsilon \leq 1$ be a constant. Hiding factors in $1/\varepsilon$ using $O_\varepsilon()$, there exists:

1. An optimum randomized algorithm that runs in $O(\log n)$ MPC rounds.
2. A $(1 + \varepsilon)$-approximation (worst case) det. algo. that runs in $O_\varepsilon(\log \log^* n)$ MPC rounds.
3. A $(1 + \varepsilon)$-approximation (worst case) randomized algo. that runs in $O_\varepsilon(1)$ MPC rounds.

Finally, we give a simple $O(\lambda^2)$-approximate (worst-case) algorithm in $O(1)$ MPC rounds.

Corollary 17 (Simple algorithm (Informal)). Let $G$ be a complete signed graph such that $E^+$ induces a $\lambda$-arboric graph. Then, there exists an $O(\lambda^2)$-approximation (worst case) deterministic algorithm that runs in $O(1)$ MPC rounds.

The simple algorithm is as follows: connected components which are cliques form clusters, and all other vertices form individual singleton clusters. This can be implemented in $O(1)$ MPC rounds using broadcast trees. We now give an informal argument when the input graph is a single connected component but not a clique. By Lemma 11 there will be $\geq n/\lambda$ clusters and so the optimal number of disagreements is $\geq n/\lambda$. Meanwhile, the singleton clusters incur errors on all positive edges, i.e., $\leq \lambda \cdot n$ since $E^+$ induces a $\lambda$-arboric graph. Thus, the worst possible approximation ratio is $\approx \lambda^2$.

3 Randomized greedy MIS on bounded degree graphs

In this section, we explain how to efficiently compute a randomized greedy MIS in the sublinear memory regime of the MPC model. We will first individually analyze Algorithms 2 and Algorithm 3 and then show how to use them as black-box subroutines in Algorithm 1.

Both Algorithm 2 and Algorithm 3 rely on the result of Fischer and Noever [21] that it suffices for each vertex to know the $\pi$ ordering of its $O(\log n)$-hop neighborhood. We defer the proofs of Lemma 18, Lemma 19, Lemma 21, and Lemma 22 to Appendix A.1.
Algorithm 2 is inspired by the graph shattering idea introduced by BEPS [6]. Our analysis follows a similar outline as the analysis of the maximal independent set of BEPS [6] but is significantly simpler as our “vertex sampling process” in each step simply follows the uniform-at-random vertex permutation \( \pi \); we do not explicitly handle high-degree vertices at the end, but we argue that connected components are still small even if \( \pi \) chooses some of them. The key crux of our analysis is to argue that, for appropriately defined step sizes, the connected components considered are of size \( O(\log n) \).

**Lemma 18.** Consider Algorithm 3. With high probability in \( n \), the connected components in any chunk graph \( G_{i,j} \) have size \( O(\log n) \).

This allows us to argue that all vertices involved can learn the full topology of their connected components via graph exponentiation in \( O(\log \log n) \) MPC rounds in Model 1.

**Lemma 19.** Consider Algorithm 2 in Model 1. Fix an arbitrary chunk graph \( G_{i,j} \). If connected components in \( G_{i,j} \) have size at most \( \text{poly}(\log n) \), then every vertex can learn the full topology of its connected component in \( O(\log \log n) \) MPC rounds.

**Lemma 20.** Consider Algorithm 2 in Model 1. Suppose \( G = (V,E) \) has \( n \) vertices with maximum degree \( \Delta \). Let \( \pi \) be a uniform-at-random ordering of \( V \). Then, with high probability, one can simulate greedy MIS on \( G \) (with respect to \( \pi \)) in \( O(\log^2 \Delta \cdot \log \log n) \) MPC rounds.

**Proof.** There are \( O(\log \Delta) \) phases, each having \( O(\log \Delta) \) iterations, in Algorithm 2. By Lemma 18 and Lemma 19, each iteration can be computed in \( O(\log \log n) \) MPC rounds. \( \square \)

By using at least \( n \) machines, Algorithm 3 presents a simpler and faster algorithm for computing greedy MIS compared to Algorithm 2. It exploits computational features of the MPC model such as graph exponentiation and round compression to speed up computation.

**Lemma 21.** Consider Algorithm 3 in Model 1. Suppose \( G = (V,E) \) has \( n \) vertices with maximum degree \( \Delta \). Let \( \pi \) be a uniform-at-random ordering of \( V \). Then, with high probability, one can simulate greedy MIS on \( G \) (with respect to \( \pi \)) in \( O(\log \log n + \log \Delta) \) MPC rounds.

Recall that Algorithm 1 uses Algorithm 2 or Algorithm 3 as subroutines to compute the greedy MIS on a subgraph induced by some prefix of \( \pi \)’s ordering in each phase. We first prove Lemma 22 which bounds the maximum degree of the remaining subgraph after processing \( t \leq n \) vertices. By our choice of prefix sizes, we see that the maximum degree is halved with high probability in each phase and thus \( O(\log \Delta) \) phases suffice.

**Lemma 22.** Let \( G \) be a graph on \( n \) vertices and \( \pi : [n] \rightarrow V \) be a uniform-at-random ordering of vertices. For \( t \in [n] \), consider the subgraph \( H_t \) obtained after processing vertices \( \{\pi(1), \ldots, \pi(t)\} \) via greedy MIS (with respect to \( \pi \)). Then, with high probability, the maximum degree in \( H_t \) is at most \( O\left(\frac{n \log n}{t}\right) \).

**Remark 23.** Similar statements to Lemma 21 and Lemma 22 were previously known.\(^{13}\)

**Theorem 24.** Let \( G \) be a graph with \( n \) vertices of maximum degree \( \Delta \) and \( \pi : [n] \rightarrow V \) be a uniform-at-random ordering of vertices. Then, with high probability, one can compute greedy MIS (with respect to \( \pi \)) in \( O(\log \Delta \cdot \log^3 \log n) \) MPC rounds in Model 1 or \( O(\log \Delta \cdot \log \log n) \) MPC rounds in Model 2.

\(^{13}\)E.g. see GGKMR [23, Section 3], ACGMW [1, Lemma 27], and BFS [10, Lemma 3.1].
Proof. There are $O(\log \Delta)$ phases in Algorithm $\mathcal{A}$ for $i \in \{1, \ldots, O(\log \Delta)\}$ and appropriate constant factors, we set $t_i = O\left(\frac{n \log n}{\Delta} \right)$ and consider the induced prefix graph $G_i$.

There are two possible subroutines to process $G_i$: Algorithm $\mathcal{A}$ and Algorithm $\mathcal{B}$. By Chernoff bounds, the maximum degree in $G_i$ is $O(\log n)$ with high probability in $n$. So, Lemma $20$ and Lemma $21$ tell us that Algorithm $\mathcal{A}$ and Algorithm $\mathcal{B}$ only need $O(\log^2 n)$ and $O(\log^2 \log n)$ MPC rounds for a single invocation respectively.

By Lemma $22$, the maximum degree of the graph after processing $G_i$ is halved with high probability in $n$. So, after $O(\log \Delta)$ phases, there are at most $\text{poly}(\log n)$ vertices left in the graph. Thus, the maximum degree is at most $\text{poly}(\log n)$ and we apply the subroutine one last time. Finally, since $\Delta \leq n$, we can apply union bound over these $O(\log \Delta)$ phases to upper bound the failure probability. \hfill \qed

4 Structural properties for correlation clustering

In this section, we prove our main result (Theorem $26$) about correlation clustering by ignoring high-degree vertices. To do so, we first show a structural result of optimum correlation clusterings (Lemma $25$): there exists an optimum clustering with bounded cluster sizes. This structural lemma also implies that in the special case of forests (i.e., $\lambda = 1$), a maximum matching on $E^+$ yields an optimum correlation clustering of $G$ (Corollary $27$).

Lemma 25 (Structural lemma for correlation clustering). Let $G$ be a complete signed graph such that positive edges $E^+$ induce a $\lambda$-arboric graph. Then, there exists an optimum correlation clustering where all clusters have size at most $4\lambda - 2$.

Proof sketch. The proof involves performing local updates by repeatedly removing vertices from large clusters while arguing that the number of disagreements does not increase (it may not strictly decrease but may stay the same). See Appendix A.2 for details. \hfill \qed

Theorem 26 (Algorithmic implication of Lemma $25$). Let $G$ be a complete signed graph such that positive edges $E^+$ induce a $\lambda$-arboric graph. For $\varepsilon > 0$, let

$$H = \left\{ v \in V : d(v) > \frac{8(1 + \varepsilon)}{\varepsilon} \cdot \lambda \right\} \subseteq V$$

be the set of high-degree vertices, and $G' \subseteq G$ be the subgraph obtained by removing high-degree vertices in $H$. Suppose $\mathcal{A}$ is an $\alpha$-approximate correlation clustering algorithm and cost($OPT(G)$) is the optimum correlation clustering cost. Then,

$$\text{cost}(\{\{v\} : v \in H\} \cup \mathcal{A}(G')) \leq \max \{1 + \varepsilon, \alpha\} \cdot \text{cost}(OPT(G))$$

where $\{\{v\} : v \in H\} \cup \mathcal{A}(G')$ is the clustering obtained by combining the singleton clusters of high-degree vertices with $\mathcal{A}$’s clustering of $G'$. See Algorithm $\mathcal{A}$ for a pseudocode. Furthermore, if $\mathcal{A}$ is $\alpha$-approximation only in expectation, then the above inequality holds only in expectation.

Proof sketch. Fix an optimum clustering $OPT(G)$ of $G$ where each cluster has size at most $4\lambda - 2$. Such a clustering exists by Lemma $25$ One can then show that cost($OPT(G)$) $\geq \frac{1}{1 + \varepsilon} \cdot |M^+| + \text{disagreements in } U$, where $|M^+|$ is the number of positive edges adjacent to high-degree vertices and $U$ is the set of edges not adjacent to any high-degree vertex. The result follows by combining singleton clusters of high-degree vertices $H$ and the $\alpha$-approximate clustering on low-degree vertices $U$ using $\mathcal{A}$. See Appendix A.2 for details. \hfill \qed
Algorithm 4 Correlation clustering for $G$ such that $E^+$ induces a $\lambda$-arboric graph

1: **Input:** Graph $G$, $\varepsilon > 0$, $\alpha$-approximate algorithm $A$
2: Let $H = \{v \in V : d(v) > \frac{8(1+\varepsilon)}{\varepsilon} \lambda\} \subseteq V$ be the set of high-degree vertices.
3: Let $G' \subseteq G$ be a bounded degree subgraph obtained by removing high-degree vertices $H$.
4: Let $A(G')$ be the clustering obtained by running $A$ on the subgraph $G'$.
5: **Return** Clustering $\{\{v\} : v \in H\} \cup A(G')$.

**Corollary 27** (Maximum matchings yield optimum correlation clustering in forests). Let $G$ be a complete signed graph such that positive edges $E^+$ induce a forest (i.e. $\lambda = 1$). Then, clustering using a maximum matching on $E^+$ yields an optimum cost correlation clustering.

Proof. See Appendix A.2.

5 Minimizing disagreements in bounded arboricity graphs and forests

We now describe how to use our main result (Theorem 26) to obtain efficient correlation clustering algorithms in the sublinear memory regime of the MPC model. Theorem 26 implies that we can focus on solving correlation clustering on graphs with maximum degree $O(\lambda^2)$.

For general $\lambda$-arboric graphs, we simulate PIVOT by invoking Theorem 24 to obtain Corollary 28. For forests, Corollary 27 states that a maximum matching on $E^+$ yields an optimal correlation clustering. Then, Lemma 29 tells us that if one computes an approximate matching (not necessarily maximal) instead of a maximum matching, we still get a reasonable cost approximation to the optimum correlation clustering. By invoking existing matching algorithms, we show how to obtain three different correlation clustering algorithms (with different guarantees) in Corollary 31. Finally, Corollary 32 gives a deterministic constant round algorithm that yields an $O(\lambda^2)$ approximation.

**Corollary 28.** Let $G$ be a complete signed graph such that positive edges $E^+$ induce a $\lambda$-arboric graph. With high probability, there exists an algorithm that produces a $3$-approximation (in expectation) for correlation clustering of $G$ in $O(\log \lambda \cdot \log^2 \log n)$ MPC rounds in Model 1 or $O(\log \lambda \cdot \log \log n)$ MPC rounds in Model 2.

Proof. Run Algorithm 4 with $\varepsilon = 2$ and PIVOT as $A$. The approximation guarantee is due to the fact that PIVOT gives a $3$-approximation in expectation. Since $\varepsilon = 2$, the maximum degree in $G'$ is $12\lambda$. Set $\Delta = 12\lambda$ in Theorem 24.

**Lemma 29.** Let $G$ be a complete signed graph such that positive edges $E^+$ induce a forest. Suppose $|M^*|$ is the size of a maximum matching on $E^+$ and $M$ is an approximate matching on $E^+$ where $\alpha \cdot |M| \geq |M^*|$ for some $1 \leq \alpha \leq 2$. Then, clustering using $M$ yields an $\alpha$-approximation to the optimum correlation clustering of $G$.

Proof. Clustering based on any matching (i.e. forming clusters of size two for each matched pair of vertices and singleton clusters for unmatched vertices) incurs $n - 1 - |M|$ disagreements. By Corollary 27, clustering with respect to a maximum matching yields a correlation clustering of optimum cost. If $|M^*| = |M|$, then the approximation ratio is trivially 1. Henceforth, let
\[ |M| \leq |M^*| - 1. \text{ Observe that } \frac{n-1-\frac{1}{\alpha}}{n-|M^*|} \leq \alpha \iff |M^*| \cdot (1 + \frac{1}{\alpha}) \leq n - 1. \text{ Indeed,} \]
\[ |M^*| \cdot \left( 1 + \frac{1}{\alpha} \right) \leq |M^*| + |M| \quad \text{since } \alpha \cdot |M| \geq |M^*| \]
\[ \leq 2 \cdot |M^*| - 1 \quad \text{since } |M| \leq |M^*| - 1 \]
\[ \leq n - 1 \quad \text{since } |M^*| \leq \frac{n}{2} \text{ for any maximum matching} \]

Thus, the approximation factor for using \( M \) is \( \frac{n-1-|M|}{n-|M^*|} \leq \frac{n-1-\frac{1}{\alpha}}{n-|M^*|} \leq \alpha. \]

\[ \blacktriangleright \textbf{Remark 30.} \text{ The approximation ratio of Lemma 29 tends to 1 as } |M| \text{ tends to } |M^*|. \text{ The worst ratio possible is 2 and this approximation ratio is tight: consider a path of 4 vertices and 3 edges with } |M^*| = 2 \text{ and maximal matching } |M| = 1. \]

\[ \blacktriangleright \textbf{Corollary 31.} \text{ Consider Model 1. Let } G \text{ be a complete signed graph such that positive edges } E^+ \text{ induce a forest. Let } 0 < \varepsilon \leq 1 \text{ be a constant. Then, there exists the following algorithms for correlation clustering:} \]
\[(1) \text{ An optimum randomized algorithm that runs in } \tilde{O}(\log n) \text{ MPC rounds.} \]
\[(2) \text{ A } (1 + \varepsilon)-\text{approx. (worst case) deterministic algo. that runs in } O \left( \frac{1}{n} \cdot (\log \frac{1}{\varepsilon} + \log \log n) \right) \text{ MPC rounds.} \]
\[(3) \text{ A } (1 + \varepsilon)-\text{approx. (worst case) randomized algo. that runs in } O \left( \log \log \frac{1}{\varepsilon} \right) \text{ MPC rounds.} \]

\[ \textbf{Proof sketch.} \text{ For (i), use the algorithm of BBDHM [7]. For (ii) and (iii), apply Theorem 26 with } \lambda = 1, \alpha = 1/(1 + \varepsilon), \text{ then use the deterministic algorithm of Even, Medina and Ron [20] and randomized algorithm of BCGS [5] respectively. See Appendix A.3 for details.} \]

\[ \blacktriangleright \textbf{Corollary 32.} \text{ Consider Model 1. Let } G \text{ be a complete signed graph such that positive edges } E^+ \text{ induce a } \lambda \text{-arboric graph. Then, there exists a deterministic algorithm that produces an } O(\lambda^2) \text{-approximation (worst case) for correlation clustering of } G \text{ in } O(1) \text{ MPC rounds.} \]

\[ \textbf{Proof sketch.} \text{ Consider the following deterministic algorithm: Each connected component (with respect to } E^+) \text{ that is a clique forms a single cluster, then all remaining vertices form singleton clusters. This can be implemented in } O(1) \text{ MPC rounds using broadcast trees. For the approximation ratio, fix an optimum clustering } OPT(G) \text{ of } G \text{ such that each cluster has size at most } 4\lambda - 2. \text{ Such a clustering exists by Lemma 25. One can then show that our algorithm incurs a ratio of at most } O(\lambda^2) \text{ for any arbitrary connected component } H, \text{ with respect to } OPT(G). \text{ See Appendix A.3 for details.} \]

\[ \blacktriangleright \textbf{Remark 33.} \text{ The approximation analysis in Corollary 32 is tight (up to constant factors): consider the barbell graph where two cliques } K_\lambda \text{ (cliques on } \lambda \text{ vertices) are joined by a single edge. The optimum clustering forms a cluster on each } K_\lambda \text{ and incurs one external disagreement. Meanwhile, forming singleton clusters incurs } \approx \lambda^2 \text{ positive disagreements.} \]

\section{Conclusions and open questions}

In this work, we present a structural result on correlation clustering of complete signed graphs such that the positive edges induce a bounded arboricity graph. Combining this with known algorithms, we obtain efficient algorithms in the sublinear memory regime of the MPC model. We also showed how to compute a randomized greedy MIS in \( O(\log \Delta \cdot \log \log n) \) MPC rounds. As intriguing directions for future work, we pose the following questions:
Question 1. For graphs with maximum degree $\Delta \in \text{poly}(\log n)$, can one compute greedy MIS in $O(\log \log n)$ MPC rounds in the sublinear memory regime of the MPC model?

For graphs with maximum degree $\Delta \in \text{poly}(\log n)$, Algorithm 2 runs in $O(\log^3 \log n)$ MPC rounds and Algorithm 3 runs in $O(\log \log n)$ MPC rounds assuming access to at least $n$ machines. Is it possible to achieve running time of $O(\log \log n)$ MPC rounds without additional global memory assumptions?

Question 2. Can a randomized greedy MIS be computed in $O(\log \Delta + \log \log n)$ or $O(\sqrt{\log \Delta} + \log \log n)$ MPC rounds?

This would imply that a 3-approximate (in expectation) correlation clustering algorithm in the same number of MPC rounds. We posit that a better running time than $O(\log \Delta \cdot \log \log n)$ should be possible. The informal intuition is as follows: Fischer and Noever’s result [21] tells us that most vertices do not have long dependency chains in every phase, so “pipelining arguments” might work.

Question 3. Is there an efficient distributed algorithm to minimize disagreements with an approximation guarantee strictly better than 3 (in expectation), or worst-case guarantees for general graphs?

For minimizing disagreements in complete signed graphs, known algorithms (see Section 1.4) with approximation guarantees strictly less than 3 (in expectation) are based on probabilistic rounding of LPs. Can one implement such LPs efficiently in a distributed setting, or design an algorithm that is amenable to a distributed implementation with provable guarantees strictly better than 3? In this work, we gave algorithms with worst-case approximation guarantees when the graph induced by positive edges is a forest. Can one design algorithms that give worst-case guarantees for general graphs?

References

1. Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation Clustering in Data Streams. In Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML), pages 2237–2246, 2015.
2. Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating Inconsistent Information: Ranking and Clustering. Journal of the ACM (JACM), 55(5):1–27, 2008.
3. Alexandr Andoni, Zhao Song, Clifford Stein, Zhengyu Wang, and Peilin Zhong. Parallel graph connectivity in log diameter rounds. In 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS), pages 674–685. IEEE, 2018.
4. Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation Clustering. Machine Learning, 56(1-3):89–113, 2004.
5. Reuven Bar-Yehuda, Keren Censor-Hillel, Mohsen Ghaffari, and Gregory Schwartzman. Distributed Approximation of Maximum Independent Set and Maximum Matching. In Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), pages 165–174, 2017.
6. Leonid Barenboim, Michael Elkin, Seth Pettie, and Johannes Schneider. The Locality of Distributed Symmetry Breaking. Journal of the ACM (JACM), 63(3):1–45, 2016.
7. MohammadHossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, and Vahab Mirrokni. Brief Announcement: MapReduce Algorithms for Massive Trees. In 45th International Colloquium on Automata, Languages, and Programming (ICALP), pages 162:1–162:4, 2018.
8. Paul Beame, Paraschos Koutris, and Dan Suciu. Communication Steps for Parallel Query Processing. Journal of the ACM (JACM), 64(6):1–58, 2017.
Massively Parallel Correlation Clustering in Bounded Arboricity Graphs

9 Soheil Behnezhad, Sebastian Brandt, Mahsa Derakhshan, Manuela Fischer, MohammadTaghi Hajighayi, Richard M. Karp, and Jara Uitto. Massively Parallel Computation of Matching and MIS in Sparse Graphs. In Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC), pages 481–490, 2019.

10 Guy Blelloch, Jeremy Fineman, and Julian Shun. Greedy Sequential Maximal Independent Set and Matching Are Parallel on Average. In Proceedings of the Twenty-Fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), pages 308—317, 2012.

11 Nicolò Cesa-Bianchi, Claudio Gentile, Fabio Vitale, and Giovanni Zappella. A Correlation Clustering Approach to Link Classification in Signed Networks. Journal of Machine Learning Research (JMLR), 23:34.1–34.20, 2013.

12 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with Qualitative Information. Journal of Computer and System Sciences, 71(3):360–383, 2005.

13 Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. Near Optimal LP Rounding Algorithm for Correlation Clustering on Complete and Complete \( k \)-partite graphs. In Proceedings of the forty-seventh annual ACM symposium on Theory of Computing (STOC), pages 219–228, 2015.

14 Yudong Chen, Sujay Sanghavi, and Huan Xu. Clustering Sparse Graphs. In Advances in Neural Information Processing Systems (NeurIPS), pages 2204–2212, 2012.

15 Flavio Chierichetti, Nilesh Dalvi, and Ravi Kumar. Correlation Clustering in MapReduce. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge Discovery and Data mining (KDD), pages 641–650, 2014.

16 Artur Czumaj, Peter Davies, and Merav Parter. Graph sparsification for derandomizing massively parallel computation with low space. In Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), pages 175–185, 2020.

17 Artur Czumaj, Jakub Łacki, Aleksander Madry, Slobodan Mitrovic, Krzysztof Onak, and Piotr Sankowski. Round Compression for Parallel Matching Algorithms. SIAM Journal on Computing, 49(5):STOC18–1, 2019.

18 Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM, 51(1):107–113, 2008.

19 Erik D Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation Clustering in General Weighted Graphs. Theoretical Computer Science, 361(2-3):172–187, 2006.

20 Guy Even, Moti Medina, and Dana Ron. Distributed Maximum Matching in Bounded Degree Graphs. In Proceedings of International Conference on Distributed Computing and Networking (ICDCN), 2015.

21 Manuela Fischer and Andreas Noever. Tight Analysis of Parallel Randomized Greedy MIS. In Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 2152–2160, 2018.

22 Mohsen Ghaffari, Massively Parallel Algorithms, 2019. Available at: https://people.inf.ethz.ch/gmohsen/MPA19/Notes/MPA.pdf.

23 Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrović, and Ronitt Rubinfeld. Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover. In Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), pages 129–138, 2018.

24 Mohsen Ghaffari, Christoph Grunau, and Ce Jin. Improved MPC Algorithms for MIS, Matching, and Coloring on Trees and Beyond. In 34th International Symposium on Distributed Computing (DISC), pages 34:1–34:18, 2020.

25 Mohsen Ghaffari and Krzysztof Nowicki. Massively Parallel Algorithms for Minimum Cut. In Proceedings of the 39th Symposium on Principles of Distributed Computing (PODC), pages 119–128, 2020.

26 Mohsen Ghaffari and Jara Uitto. Sparsifying Distributed Algorithms with Ramifications in Massively Parallel Computation and Centralized Local Computation. In Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 1636–1653, 2019.
A Deferred proofs

In this section, we provide the deferred proofs in the main text. For convenience, we will restate them before giving the proofs. We also prove any necessary intermediate results here.

A.1 Proofs for Section 3

Lemma 34 and Lemma 35 are intermediate results needed for Lemma 18. A special case of Lemma 34 (with \( s = 0 \)) was first considered in BEPS [6]. Here, we provide more details while proving a slightly tighter bound.

\textbf{Lemma 34} (Counting connected neighborhoods). Consider a graph \( G = (V, E) \) on \( |V| = n \) vertices with maximum degree \( \Delta \geq 1 \). Let \( L \subseteq V \) be a subset of low-degree vertices with maximum degree \( 1 \leq \delta \leq \Delta \). Then, the number of connected components involving \( s \) vertices from \( L \) and \( t \) vertices from \( V \setminus L \) is at most \( n \cdot 4^{\delta t - 2} \cdot \delta^s \cdot \Delta^t \).

\textbf{Proof.} Since any connected component of size \( (s + t) \) must contain a tree on \( (s + t) \) nodes, counting the number of unlabelled \( (s + t) \)-node trees rooted at every vertex gives an upper bound on the total number of \( (s + t) \)-node connected components in the graph.\(^{14}\)

\(^{14}\)A connected component may contain multiple trees and this upper bound is generally an over-estimation.
One can count unlabelled trees using Euler tours. Given a root in an \((s + t)\)-node tree, there is a 1-to-1 correspondence between an Euler tour and a binary string \(S\) of \(2(s + t - 2)\) bits. See Figure 4 for an illustration.

Given a rooted unlabelled tree \(T\), we now upper bound the number of ways to assign labels to the unlabelled \((s + t - 1)\) vertices by upper bounding the number of possible ways to embed \(T\) into the graph \(G\). Consider labelling \(T\) with an arbitrary ordering of \(s\) vertices from \(L\) and \(t\) vertices from \(V \setminus L\). There are at most \(\delta\) choices when branching off a vertex from \(L\), while there are at most \(\Delta\) choices when branching off a vertex from \(V \setminus L\). Thus, there are at most \(\delta \cdot \Delta\) ways to embed an arbitrary tree \(T\) in \(G\).

Putting everything together, we see that there are \(n\) ways to root a tree, at most \(2^{2(s+t-2)} = 4^{s+t-2}\) possible Euler tours, and at most \(\delta \cdot \Delta\) ways to embed each tour in \(G\). That is, the number of connected components involving \(s\) vertices from \(L\) and \(t\) vertices from \(V \setminus L\) is at most \(n \cdot 4^{s+t-2} \cdot \delta \cdot \Delta\).

\[\Box\]

**Figure 4** Consider graph \(G\). Starting from root vertex \(a\), we annotate all possible Euler tours involving \(k = 3\) vertices with a corresponding \(2(k-2)\)-bit string. Each of the \(k-1\) edges corresponds to 2 bits: a “1” for “moving down the tree” and a “0” for “moving up the tree”. As each Euler tour must return to the root, the first bit is always a “1”, the last bit is always a “0”, and there are an equal number of 1’s and 0’s. Furthermore, there are at least as many 1’s as 0’s in any prefix of the binary string. Finally, observe that while \(2(k-2)\) bits uniquely identifies a rooted Euler tour tree involving \(k\) vertices, it could have multiple embeddings in \(G\).

\[\uparrow\text{Lemma 35.}\] Let vertex \(v\) be an arbitrary vertex in phase \(i \in \{0, 1, \ldots, \lceil \log_2 \Delta \rceil\}\). That is, \(v \in \bigcup_{j=1}^{2000 \log \Delta} V(G_{i,j})\). Then,

\[\Pr \left[ v \text{ has } \frac{\Delta}{2^{i-1}} \text{ neighbors in phases } i, i+1, \ldots, \lceil \log_2 \Delta \rceil \right] \leq \Delta^{-10}\]

**Proof.** When \(i = 0\), the statement trivially holds because all vertices have \(\leq \Delta\) neighbors. In phase \(i > 1\), we have processed at least \(\frac{2^i}{1000 \Delta^2} \cdot n \cdot 2000 \log \Delta\) vertices. If \(v\) has \(\frac{\Delta}{2^{i-1}}\),

\[\text{Again, this is an overestimation.}\]
neighbors in phases \(i, i + 1, \ldots, \lfloor \log_2 \Delta \rfloor\), then none of these neighbors belonged to the earlier phases. Since \(\pi\) is a uniform-at-random permutation, we see that

\[
\Pr \left[ \text{\(v\) has \(\frac{\Delta}{2^{i-1}}\) neighbors in phases \(i, i + 1, \ldots, \lfloor \log_2 \Delta \rfloor\)} \right] \\
\leq \left(1 - \frac{2^{i}}{100 \Delta} \cdot 2000 \log \Delta\right)^{\frac{n}{2^{i-1}}} \leq \Delta^{-10} 
\]

\[\bullet\] \textbf{Lemma 18.} Consider Algorithm \(3\). With high probability in \(n\), the connected components in any chunk graph \(G_{i,j}\) have size \(\mathcal{O}(\log n)\).

\textbf{Proof of Lemma 18.} Partition the vertices in \(G\) into \(L\) and \(V \setminus L\), where \(L\) is the set of vertices in \(G_{i,j}\) with degree \(\leq \frac{\Delta}{2^{i-1}}\) and \(V(G_{i,j}) \setminus L\) be the set of vertices in \(G_{i,j}\) with degree \(\geq \frac{\Delta}{2^{i-1}}\). For arbitrary choices of \(s\) and \(t\), let us denote \(C_{s,t}\) as a connected component involving \(s\) vertices from \(L\) and \(t\) vertices from \(V \setminus L\). We will argue that the probability of a connected component \(C_{s,t}\) existing in \(G_{i,j}\) is very small when \(s + t = 100 \log n \in \mathcal{O}(\log n)\).

This implies our desired statement since any connected component of size at least \(100 \log n\) must contain some \(C_{s,t}\) with \(s + t = 100 \log n\).

Fix an arbitrary connected component \(C_{s,t}\). Since \(\pi\) is a uniform-at-random permutation, vertices are assigned to \(G_{i,j}\) independently with probability \(\frac{2}{n}\). By Lemma \(3\), the probability of a vertex in \(G_{i,j}\) belonging in \(T\) is at most \(\Delta^{-10}\). Thus, the probability of \(C_{s,t}\) appearing in \(G_{i,j}\) is at most \(\left(\frac{2}{n}\right)^{s+t} \cdot \Delta^{-10t}\).

By Lemma \(3\), the number of connected components \(C_{s,t}\) is at most \(n \cdot 4^{s+t-2} \cdot \left(\frac{\Delta}{2^{i-1}}\right)^{s} \cdot \Delta^{t}\).

Recall that \(i \leq \lfloor \log_2 \Delta \rfloor\). So, by a union bound over all possible connected components \(C_{s,t}\) with \(s + t = 100 \log n\), we see that

\[
\Pr \left[ \text{There exists \(C_{s,t}\) with \(s + t = 100 \log n\) in \(G_{i,j}\)} \right] \\
\leq n \cdot 4^{s+t-2} \cdot \left(\frac{\Delta}{2^{i-1}}\right)^{s} \cdot \Delta^{t} \cdot \left(\frac{C_i}{n}\right)^{s+t} \cdot \Delta^{-10t} \\
= n \cdot 4^{s+t-2} \cdot \left(\frac{\Delta}{2^{i-1}}\right)^{s} \cdot \left(\frac{2^{i}}{100 \Delta}\right)^{s+t} \cdot \Delta^{-9t} \\
\leq n \cdot \left(\frac{4}{50}\right)^{s} \cdot \Delta^{-9t} \\
\leq n^{-5}
\]

That is, with high probability in \(n\), the connected components in \(G_{i,j}\) have size \(\mathcal{O}(\log n)\). \[\blacklozenge\]

In the above proof of Lemma \(18\), we ignored the fact that some of the vertices in \(G_{i,j}\) may have already been removed from the graph due to some neighbor in earlier phases entering the independent set. However, this can only reduce the size of the connected components that we considered and is beneficial for the purposes of our analysis.

\[\bullet\] \textbf{Lemma 19.} Consider Algorithm \(2\) in Model \(1\). Fix an arbitrary chunk graph \(G_{i,j}\). If connected components in \(G_{i,j}\) have size at most \(\text{poly}(\log n)\), then every vertex can learn the full topology of its connected component in \(\mathcal{O}(\log \log n)\) MPC rounds.

\textbf{Proof of Lemma 19.} Let \(h : [n] \to [M]\) be a hash function chosen uniformly at random from a family \(H\) of pairwise independent hash functions, where \(M = n \cdot S\) is the number
of machines and $S \in \tilde{O}(n^\delta)$. Let us assign each vertex $v$ to machine indexed by $h(v)$.

By Chernoff bounds, the number of vertices assigned to each machine is $O(S)$ with high
probability. By Lemma 18 the connected components have size $O(\log n)$. To learn about
the full topology of their connected component, we perform in $O(\log \log n)$ rounds of graph
exponentiation. Since each connected component only has size $\text{poly}(\log n)$, each vertex
requires at most $\text{poly}(\log n)$ space whilst performing graph exponentiation and thus the total
amount of memory used by any collection of vertices on the same machine fits into the
machine memory.

Lemma 21. Consider Algorithm 5 in Model 2. Suppose $G = (V,E)$ has $n$ vertices with
maximum degree $\Delta$. Let $\pi$ be a uniform-at-random ordering of $V$. Then, with high probability,
one can simulate greedy MIS on $G$ (with respect to $\pi$) in $O(\log \log n + \log \Delta)$ MPC rounds.

Proof of Lemma 21. By Theorem 5 it suffices for any vertex to learn the ordering of
vertices within its $O(\log n)$-hop neighborhood in $G$ to determine whether itself is in the
MIS. However, due to machine memory constraints, vertices may not be able to directly
store their full $O(\log n)$-hop neighborhoods in a single machine. Instead, vertices will
gather their $O(\log n/\log \Delta)$-hop neighborhood via graph exponentiation, then simulate the greedy
MIS algorithm in $O(\log \Delta)$ compressed rounds. The total runtime of this procedure is
$O(\log \log n + \log \Delta)$.

It remains to show that the $O(\log n/\log \Delta)$-hop neighborhood of any vertex fits in a single
machine. Let us be more precise about the constant factors involved. Suppose that the
longest dependency chain in greedy MIS has length $L \cdot \log n$ for some constant $L > 0$. For
some constant $C > 0$, let $R = C \cdot L \cdot \left(\frac{\log n}{\log \Delta}\right)$ denote the $R$-hop neighborhood that we want
to collect into a single machine with memory $\tilde{O}(n^\delta)$. Note that the parameters $L$ and $\delta$
are given to us, and we only have control over the parameter $C$. If we pick $C$ such that
$C \cdot L < \delta < 1$, then

$$R \cdot \log \Delta = C \cdot L \cdot \left(\frac{\log n}{\log \Delta}\right) \cdot \log \Delta = C \cdot L \log n \in O(\delta \cdot \log n) \quad \iff \quad \Delta^R \in O(n^\delta)$$

Thus, with appropriate constant factors, the $O(\log n/\log \Delta)$-hop neighborhood of any vertex fits
in a single machine.

Lemma 22. Let $G$ be a graph on $n$ vertices and $\pi : [n] \to V$ be a uniform-at-random
ordering of vertices. For $t \in [n]$, consider the subgraph $H_t$ obtained after processing vertices
$\{\pi(1), \ldots, \pi(t)\}$ via greedy MIS (with respect to $\pi$). Then, with high probability, the maximum
degree in $H_t$ is at most $O\left(\frac{n \log n}{t}\right)$.

Proof of Lemma 22. For the sake of clarity, we now prove the statement by setting the
maximum degree bound in $H_t$ to $\frac{10n \log n}{t}$. The constant 10 is arbitrary and can be adjusted
based on how this lemma is invoked.

Let $t' \in [t]$ be an arbitrary round and $v$ an arbitrary vertex. Suppose $v$ has degree
d $d_{t'-1}(v)$ after processing the first $(t'-1)$ vertices defined by $\pi(1), \ldots, \pi(t'-1)$. If $t' = 1$,
then nothing has been processed yet and $d_{t'-1}(v) = d_0(v) = d(v)$, where $d(v)$ is the degree of
$v$ in the input graph $G$. If $d_{t'-1}(v) \geq \frac{10n \log n}{t}$, then the vertex $v$ already satisfies the lemma
since vertex degrees never increase while processing $\pi$. Otherwise, $d_{t'-1}(v) > \frac{10n \log n}{t}$. We
now proceed to upper bound the probability of vertex $v$ remaining in the subgraph after
processing $\pi(t')$. 
For vertex $v$ to remain, neither $v$ nor any of its neighbors must be chosen to be $\pi(t' - 1)$. Since $\pi$ is a uniform-at-random ordering of vertices, this happens with probability

$$1 - \frac{1 + d_C^+(v)}{n - 1} \leq 1 - \frac{d_C^+(v)}{n} < 1 - \frac{10\log n}{n} = 1 - \frac{\log n}{t}. \quad \text{Thus, the probability that}$$

vertex $v$ remains in $H_t$, while having $d_C(v) > \frac{10\log n}{t}$ after processing $\pi(t)$, is at most

$$\left(1 - \frac{\log n}{t}\right)^t \leq \exp(-10\log n) = n^{-10}. \quad \text{The lemma follows by taking a union bound over}$$

all vertices. ▶

A.2 Proofs for Section 4

Lemma 25 (Structural lemma for correlation clustering). Let $G$ be a complete signed graph such that positive edges $E^+$ induce a $\lambda$-arboric graph. Then, there exists an optimum correlation clustering where all clusters have size at most $4\lambda - 2$.

Proof of Lemma 25. The proof involves performing local updates by repeatedly removing vertices from large clusters while arguing that the number of disagreements does not increase (it may not strictly decrease but may stay the same).

Consider an arbitrary clustering that has a cluster $C$ of size at least $|C| \geq 4\lambda - 1$. We will show that there exists some vertex $v^* \in C$ such that $d_C^+(v^*) \leq 2\lambda - 1$. Observe that removing $v^*$ to form its own singleton cluster creates $d_C^+(v^*)$ positive disagreements and removes $(|C| - 1) - d_C^-(v^*)$ negative disagreements. Since $d_C^-(v^*) \leq 2\lambda - 1 \leq \frac{|C| + 1}{2} - 1 = \frac{|C| - 1}{2}$, we see that this local update will not increase the number of disagreements. It remains to argue that $v^*$ exists.

Suppose, for a contradiction, that such a vertex $v^*$ does not exist in a cluster of size $|C| \geq 4\lambda - 1$. Then, $d_C^+(v) \geq 2\lambda$ for all vertices $v \in C$. Summing over all vertices in $C$, we see that

$$|E^+(C)| = \frac{1}{2} \sum_{v \in C} d_C^+(v) \geq \frac{1}{2} \cdot |C| \cdot 2\lambda = |C| \cdot \lambda \geq |C| \cdot \frac{|E^+(C)|}{|C| - 1} > |E^+(C)|$$

where the second last inequality follows from the definition of arboricity. This is a contradiction, thus such a vertex $v^*$ exists. Repeating this argument (i.e. removing all vertices like $v^*$ to form their own singleton clusters), we can transform any optimum clustering into one with clusters of size at most $4\lambda - 2$. ▶

Lemma 26 (Algorithmic implication of Lemma 25). Let $G$ be a complete signed graph such that positive edges $E^+$ induce a $\lambda$-arboric graph. For $\varepsilon > 0$, let

$$H = \left\{v \in V : d(v) > \frac{8(1 + \varepsilon)}{\varepsilon} \cdot \lambda\right\} \subseteq V$$

be the set of high-degree vertices, and $G' \subseteq G$ be the subgraph obtained by removing high-degree vertices in $H$. Suppose $A$ is an $\alpha$-approximate correlation clustering algorithm and cost($OPT(G)$) is the optimum correlation clustering cost. Then,

$$\text{cost}\left(\left\{v : v \in H\right\} \cup A(G')\right) \leq \max\{1 + \varepsilon, \alpha\} \cdot \text{cost}(OPT(G))$$

where $\left\{v : v \in H\right\} \cup A(G')$ is the clustering obtained by combining the singleton clusters of high-degree vertices with $A$’s clustering of $G'$. See Algorithm 3 for a pseudocode. Furthermore, if $A$ is $\alpha$-approximation only in expectation, then the above inequality holds only in expectation.
Denote edges incident to high-degree vertices as marked \((M)\), and unmarked \((U)\) otherwise. We further split marked edges into positive \((M^+)\) and negative marked edges \((M^-)\). In other words, we partition the edge set \(E\) into \(M^+ \cup M^- \cup U\). Instead of the usual handshaking lemma\(^\text{16}\), we have
\[
|M^+| \leq \sum_{v \in H} d^+(v) \leq 2 \cdot |M^+| \tag{1}
\]
because high-degree vertices may have low-degree neighbors and marked edges may be counted twice in the sum. See Figure 5 for an illustration.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure5.png}
\caption{High-degree vertices \(H\) are filled and only edges in \(M^+\) are shown. Edges contributing to \(d^+(v)\) are highlighted in red. In the summation of Equation (1), dashed edges are counted only once and solid edges are counted twice, hence \(|M^+| \leq \sum_{v \in H} d^+(v) \leq 2 \cdot |M^+|\).}
\end{figure}

Fix an optimum clustering \(OPT(G)\) of \(G\) where each cluster has size at most \(4\lambda - 2\). Such a clustering exists by Lemma 25. Observe that
\[
cost(OPT(G)) = (\text{disagreements in } M^+) + (\text{disagreements in } M^-) + (\text{disagreements in } U)
\geq (\text{disagreements in } M^+) + 0 + (\text{disagreements in } U)
\]
We defer the proof for the following inequality and first use it to derive our result:
\[
cost(OPT(G)) \geq \frac{1}{1 + \varepsilon} \cdot |M^+| + (\text{disagreements in } U) \tag{2}
\]
Since ignoring high-degree vertices in the clustering from \(OPT(G)\) yields a valid clustering for \(G'\), we see that \(cost(OPT(G')) \leq (\text{disagreements in } U)\). Thus,
\[
\begin{align*}
\text{cost } & \left( \{ v \in \{ v : v \in H \} \cup A(G') \} \right) \\
& = |M^+| + \text{cost}(A(G')) \\
& \leq |M^+| + \alpha \cdot \text{cost}(OPT(G')) \quad \text{since } A \text{ is } \alpha\text{-approximate} \\
& \leq |M^+| + \alpha \cdot (\text{disagreements in } U) \quad \text{since } \text{cost}(OPT(G')) \leq (\text{disagreements in } U) \\
& \leq \max \{ 1 + \varepsilon, \alpha \} \cdot \text{cost}(OPT(G)) \quad \text{by Equation (2)}
\end{align*}
\]
If \(A\) is only \(\alpha\)-approximate in expectation, the same argument yields the same conclusion, but only in expectation.

To prove Equation (2), it suffices to show that \((\text{disagreements in } M^+) \geq \frac{1}{1 + \varepsilon} \cdot |M^+|\).

Consider an arbitrary high-degree vertex \(v \in H\). By choice of \(OPT(G)\) and definition of \(H\),

\[\text{Handshaking lemma: } \sum_{v \in V} d(v) = 2|E|\]
at most
\[
4\lambda - 2 \leq 4\lambda \leq \frac{\varepsilon}{2(1 + \varepsilon)} \cdot d(v)
\]  
(3) of marked edges incident to \( v \) belong in the same cluster and will not cause any external positive disagreements. Let us call these edges \( \text{good edges} \) since they do not incur any cost in \( \text{cost}(\text{OPT}(G)) \). In total, across all high-degree vertices, there are at most \( \sum_{v \in \mathcal{H}} \frac{\varepsilon}{2(1 + \varepsilon)} \cdot d(v) \leq \frac{\varepsilon}{2(1 + \varepsilon)} \cdot 2 \cdot |M^+| = \frac{\varepsilon}{1 + \varepsilon} \cdot |M^+| \) good edges due to Equation (3) and Equation (1). In other words, \( \text{(disagreements in } M^+ \text{)} \geq \left( 1 - \frac{1}{1 + \varepsilon} \right) \cdot |M^+| = \frac{1}{1 + \varepsilon} \cdot |M^+| \).

\[\blacktriangleleft\]

**Corollary 27.** (Maximum matchings yield optimum correlation clustering in forests) Let \( G \) be a complete signed graph such that positive edges \( E^+ \) induce a forest (i.e. \( \lambda = 1 \)). Then, clustering using a maximum matching on \( E^+ \) yields an optimum cost correlation clustering.

**Proof of Corollary 27.** By Lemma 25 with \( \lambda = 1 \), we know that there exists an optimum correlation clustering where clusters are of size 1 or 2. A size-2 cluster reduces one disagreement if the vertices are joined by a positive edge. Hence, the total number of disagreements is minimized by maximizing the number of such size-2 clusters, which is the same as computing the maximum matching on the set of positive edges \( E^+ \).

\[\blacktriangleleft\]

**A.3 Proofs for Section 5**

**Corollary 31.** Consider Model 1. Let \( G \) be a complete signed graph such that positive edges \( E^+ \) induce a forest. Let \( 0 < \varepsilon \leq 1 \) be a constant. Then, there exists the following algorithms for correlation clustering:

1. An optimum randomized algorithm that runs in \( \tilde{O}(\log n) \) MPC rounds.
2. A \((1 + \varepsilon)\)-approx. (worst case) deterministic algo. that runs in \( O \left( \frac{1}{\varepsilon} \cdot (\log \frac{1}{\varepsilon} + \log^* n) \right) \) MPC rounds.
3. A \((1 + \varepsilon)\)-approx. (worst case) randomized algo. that runs in \( O \left( \log \log \frac{1}{\varepsilon} \right) \) MPC rounds.

**Proof of Corollary 31.** For the first algorithm, we use the algorithm of BBDHM [7] to compute a maximum matching in \( \tilde{O}(\log n) \) MPC rounds, and cluster matched vertices together according to Corollary 27.

For the second algorithm, we apply Theorem 26 with \( \lambda = 1 \), \( \alpha = 1/(1 + \varepsilon) \), and \( \mathcal{A} \) as the deterministic approximate matching algorithm of Even, Medina and Ron [20] on the subgraph with maximum degree \( \Delta \in O(1/\varepsilon) \). Their algorithm runs in \( R \in O \left( \Delta^{O(1/\varepsilon)} + \frac{1}{\varepsilon} \cdot \log^* n \right) \) LOCAL rounds. This can be sped up to \( O \left( \frac{1}{\varepsilon} \cdot (\log \frac{1}{\varepsilon} + \log^* n) \right) \) MPC rounds via graph exponentiation since \( \Delta \in O(1/\varepsilon) \) and each \( R \)-hop neighborhood is of logarithmic size.

For the third algorithm, we apply Theorem 26 with \( \lambda = 1 \), \( \alpha = 1 + \varepsilon \), and \( \mathcal{A} \) as the randomized approximate matching algorithm of BCGS [5] on the subgraph with maximum degree \( \Delta \in O(1/\varepsilon) \). Their algorithm runs in \( R \in O \left( \log \frac{1}{\log \log \varepsilon} \right) \subseteq O \left( \log \frac{1}{\varepsilon} \right) \) CONGEST rounds. This can be sped up to \( O(\log \log \frac{1}{\varepsilon}) \) MPC rounds via graph exponentiation since \( \Delta \in O(1/\varepsilon) \) and each \( R \)-hop neighborhood is of logarithmic size.

**Remark 36.** Let \( M^* \) be some maximum matching and \( M \) be some approximate matching. The approximation ratio in [20] is stated as \( |M| = (1 - \varepsilon') \cdot |M^*| \) while we write \( (1 + \varepsilon) \cdot |M^*| = |M^*| \). This is only a constant factor difference: \( \varepsilon' \in \Theta(\varepsilon) \).

**Corollary 32.** Consider Model 1. Let \( G \) be a complete signed graph such that positive edges \( E^+ \) induce a \( \lambda \)-arboric graph. Then, there exists a deterministic algorithm that produces an \( O(\lambda^2) \)-approximation (worst case) for correlation clustering of \( G \) in \( O(1) \) MPC rounds.
Proof. Consider the following deterministic algorithm: Each connected component (with respect to $E^+$) that is a clique forms a single cluster, then all remaining vertices form singleton clusters.

MPC implementation. Any clique in a $\lambda$-arboric graph involves at most $2\lambda$ vertices. Ignoring vertices with degrees larger than $(2\lambda - 1)$, the algorithm can be implemented in $O(1)$ MPC rounds using broadcast trees.

Approximation analysis. Fix an optimum clustering $OPT(G)$ of $G$ such that each cluster has size at most $4\lambda - 2$. Such a clustering exists by Lemma 25. Note that clusters in any optimum clustering are connected components (with respect to $E^+$), otherwise one can strictly improve the cost by splitting up such clusters. By bounding the approximation ratio for an arbitrary connected component in the input graph $G$, we obtain a worst case approximation ratio.

Consider an arbitrary connected component $H$ on $n$ vertices and $m$ positive edges. Since $H$ is $\lambda$-arboric, $m \leq \lambda \cdot n$. If $H$ is a clique, then our algorithm incurs zero disagreements. Otherwise, our algorithm forms singleton clusters and incurs $m \leq \lambda \cdot n$ disagreements. Since each cluster in $OPT(G)$ has size at most $4\lambda - 2$, there must be at least \( \frac{n}{4\lambda - 2} \) clusters of $OPT(G)$ involving vertices in $H$. Since $H$ is a connected component, this means that $OPT(G)$ incurs at least \( \frac{n}{4\lambda - 2} - 1 \) positive external disagreements with respect to $H$. Thus, the worst possible approximation ratio is \( \frac{\lambda \cdot n}{\frac{\lambda \cdot n}{4\lambda - 2} - 1} \in O(\lambda^2) \). ◀