Simultaneously Approximating All Norms for Massively Parallel Correlation Clustering

Nairen Cao [*]
Department of Computer Science
Boston College
Chestnut Hill, MA, United States
nc1827@nyu.edu

Shi Li [†]
School of Computer Science,
Nanjing University,
Nanjing, Jiangsu Province, China
shili@nju.edu.cn

Jia Ye [†]
School of Computer Science,
Nanjing University,
Nanjing, Jiangsu Province, China
jiaye@smail.nju.edu.cn

Abstract

We revisit the simultaneous approximation model for the correlation clustering problem introduced by Davies, Moseley, and Newman [DMN24]. The objective is to find a clustering that minimizes given norms of the disagreement vector over all vertices.

We present an efficient algorithm that produces a clustering that is simultaneously a 63.3-approximation for all monotone symmetric norms. This significantly improves upon the previous approximation ratio of 6348 due to Davies, Moseley, and Newman [DMN24], which works only for ℓ_p-norms.

To achieve this result, we first reduce the problem to approximating all top-k norms simultaneously, using the connection between monotone symmetric norms and top-k norms established by Chakrabarty and Swamy [CS19]. Then we develop a novel procedure that constructs a 12.66-approximate fractional clustering for all top-k norms. Our 63.3-approximation ratio is obtained by combining this with the 5-approximate rounding algorithm by Kalhan, Makarychev, and Zhou [KMZ19].

We then demonstrate that with a loss of ϵ in the approximation ratio, the algorithm can be adapted to run in nearly linear time and in the MPC (massively parallel computation) model with poly-logarithmic number of rounds.

By allowing a further trade-off in the approximation ratio to $(359 + \epsilon)$, the number of MPC rounds can be reduced to a constant.

[*]The work of NC was supported by NSF grant CCF-2008422.

[†]The work of SL and JY was supported by the State Key Laboratory for Novel Software Technology, and the New Cornerstone Science Laboratory.

Contents

1 Introduction

Clustering is a classic problem in unsupervised machine learning. It aims to classify a given set of data elements based on their similarities, with the goal of maximizing the similarity between elements within the same class and minimizing the similarity between elements in different classes. Among the various graph clustering problems, correlation clustering stands out as a classic model. Initially proposed by Bansal, Blum, and Chawla [BBC04], the model has numerous practical applications, including automated labelling [CKP08, AHK⁺09], community detection and mining [CSX12, VGW18, SDE⁺21], and disambiguation task [KCMNT08], among others.

The input of the standard correlation clustering problem is a complete graph over a set V of n vertices, where edges are partitioned into the set E^+ of +edges and the set E^- of −edges. The output of the problem is a clustering (or a partition) $\mathcal{C}$ of V that minimizes the number of edges in disagreement: an edge $uv \in \binom{V}{2}$ is in disagreement if $uv \in E^+$ but u and v are in different clusters in $\mathcal{C}$, or $uv \in E^-$ but u and v are in a same cluster in $\mathcal{C}$. Throughout the paper, we shall use a graph $G = (V, E)$ to denote a correlation clustering instance, with E being E^+ and $\binom{V}{2} \setminus E$ being E^-.

This problem is known to be APX-Hard [CGW05]. There has been a long stream of $O(1)$-approximation algorithms for the problem [BBC04, CGW05, ACN08, CMSY15, CLN22, CLLN23, CCAL⁺24], with the current best approximation ratio being 1.437 [CCAL⁺24]. In the same paper, the authors presented an improved hardness of 24/23 for the problem, which also made the constant explicit.

Besides the standard setting, other objectives have been studied recently, with the goal of minimizing some norm of the *disagreement vector* of the clustering $\mathcal{C}$ over vertices. For a clustering $\mathcal{C}$ of V, the disagreement vector of $\mathcal{C}$ is defined as $\mathrm{cost}_{\mathcal{C}} \in \mathbb{Z}^n_{\geq 0}$, where $\mathrm{cost}_{\mathcal{C}}(u)$ for every $u \in V$ is the number of edges incident to u that are in disagreement with respect to $\mathcal{C}$. Given some norm $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$ [1], the goal of the problem is to minimize $f(\mathrm{cost}_{\mathcal{C}})$. Notice that the standard correlation clustering problem corresponds to the case where f is the ℓ_1 norm.

Puleo and Milenkovic [PM16] initiated the study of correlation clustering with the goal of minimizing the ℓ_p norm of the disagreement vector, where $p \in [1, \infty]$. They proved that the problem is NP-hard for the ℓ_∞-norm objective. Given a fixed $p \in [1, \infty]$, for the ℓ_p-norms objective, they gave a 48-approximation algorithm. The approximation ratio was subsequently improved by Charikar, Gupta, and Schwartz [CGS17] to 7 for the ℓ_∞-norm, and by Kalhan, Makarychev and Zhou [KMZ19] to 5 for the ℓ_p-norm with any fixed $p \in [1, \infty]$. Very recently, Heidrich, Irmai, and Andres [HIA24] improved the approximate ratio to 4 for the ℓ_∞-norm.

Davies, Moseley and Newman [DMN24] introduced the concept of simultaneous approximation for all ℓ_p-norms. They developed an efficient algorithm that outputs a single clustering $\mathcal{C}$, which is simultaneously an $O(1)$-approximation for the ℓ_p norm for all $p \in [1, \infty]$. This is rather surprising, as it was not known a priori whether such a clustering $\mathcal{C}$ even exists. To achieve the goal, they first construct a fractional clustering x that is simultaneously an $O(1)$-approximation for all ℓ_p norms and then use the 5-approximate rounding algorithm of Kalhan, Makarychev, and Zhou [KMZ19] to round x into an integral clustering $\mathcal{C}$. Crucially, the algorithm of [KMZ19] guarantees a per-vertex 5-approximation, meaning that $\mathrm{cost}_{\mathcal{C}}(u)$ is at most 5 times the fractional number of edges in disagreement incident to u, for every $u \in V$. This strong property is necessary to obtain the final

[1] This means f satisfies $f(\alpha x) = \alpha f(x)$ for every real $\alpha \geq 0$ and $x \in \mathbb{R}^n_{\geq 0}$, and $f(x + y) \leq f(x) + f(y)$ for every $x, y \in \mathbb{R}^n_{\geq 0}$

simultaneous $O(1)$-approximation in [DMN24].

In light of the growing networks, it is imperative to develop efficient parallel algorithms. This urgency is particularly pronounced in machine learning and data mining applications, where timely and efficient processing is essential for extracting meaningful insights from vast datasets. Many works in the literature aim to design efficient parallel algorithms [BFS12, CDK14, PPO+15, FN19, CCMU21, CALM+21, AW22, CKL+24, CHS24]. The MPC model, as a theoretical abstraction of several real-world parallel models such as MapReduce [DG08], is a prevalent methodology employed in these works.

1.1 Our results

In this paper, we revisit and generalize the simultaneous approximation model for the correlation clustering that was introduced by [DMN24]. Instead of considering only ℓ_p norms, we consider all *monotone symmetric norms*. We say a norm $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$ is monotone if for every $x, y \in \mathbb{R}^n_{\geq 0}$ with $x \leq y$, we have $f(x) \leq f(y)$. We say f is symmetric if $f(x) = f(x')$ for every $x, x' \in \mathbb{R}^n_{\geq 0}$ such that x' is a permutation of x. Such norms were considered in [CS19] in the context of load balancing and clustering. Our first result is that there exists simultaneous $O(1)$-approximation for all monotone symmetric norms for correlation clustering and it can be constructed in polynomial time.

Definition 1.1. *Given a correlation clustering instance $G = (V, E)$ and $\alpha \geq 1$, we say a clustering $\mathcal{C}$ over V is simultaneously α-approximate, or a simultaneous α-approximation, for a family F of norms, if we have $f(\text{cost}_{\mathcal{C}}) \leq \alpha \cdot f(\text{cost}_{\text{OPT}_f})$ for every $f \in F$, where OPT_f is the optimum clustering for G under norm f.*

Theorem 1.2. *Given a correlation clustering instance $G = (V, E)$, in polynomial time we can construct a simultaneous 63.3-approximate clustering $\mathcal{C}$ for the family of monotone symmetric norms.*

Next, we are concerned with the running time of the algorithm and its implementation under the MPC model. To state the result, we need a formal description of the MPC model.

The MPC model. In the MPC model, data is distributed across a set of machines, and computation proceeds in synchronous rounds. During each round, each machine first receives messages from other machines, then performs computations based on this information and its own allocated memory, and finally sends messages to other machines to be received at the start of the next round. Each machine has limited local memory, restricting the total number of messages it can receive or send in a round. The efficiency of the algorithm is measured by the number of rounds, the memory used by each machine, the total memory used by all machines, and the running time over all machines, also known as the total work.

In this paper, we consider the MPC model in the *strictly sublinear regime*: Each machine has $O(n^\delta)$ local memory, where n is the input size and $\delta > 0$ is a constant that can be made arbitrarily small. Under this model, we assume the input received by each machine has size $O(n^\delta)$.

We then describe the correlation clustering problem under the MPC model in the strictly sublinear regime. We use $n = |V|$ and $m = |E|$ to denote the number of vertices and edges respectively in the input graph $G = (V, E)$. The edges E are distributed across the machines, where each machine has $O(n^\delta)$ memory for a constant $\delta > 0$ which can be made arbitrarily small.

At the end of the algorithm, each machine needs to store in its local memory the IDs of the clusters for all the vertices incident to its assigned edges.

Our main result regarding MPC algorithm is given as follows,

Theorem 1.3. *Let $\epsilon \in (0,1)$. There exists a randomized MPC algorithm in the strictly sublinear regime that, given a correlation clustering instance $G = (V, E)$, in $O(\log^3 n)$ rounds outputs a simultaneous $(63.3 + O(\epsilon))$ clustering for G for all monotone symmetric norms. This algorithm succeeds with high probability. It uses $\tilde{O}(m/\epsilon^6)$ total memory and $\tilde{O}(m/\epsilon^6)$ total work.*[2]

In particular, the algorithm can be converted into a nearly linear time algorithm that with high probability outputs a $(63.3+O(\epsilon))$-simultaneous approximation for all monotone symmetric norms.

Along the way, we develop an MPC rounding algorithm with a per-vertex $(5 + 55\epsilon)$ approximation guarantee, based on the sequential algorithm due to [KMZ19]. Given its potential independent interest, we state it here for future references.

Theorem 1.4. *Let $\epsilon \in (0,1)$ be a constant. Given a graph $G = (V, E)$ and a set of LP value $(x_{uv})_{u,v\in V}$ satisfying the approximate triangle inequality, that is, for any $u, v, w \in V$, we have $x_{uv} + x_{uw} + \epsilon \geq x_{vw}$. Let $y_u = \sum_{uv\in E} x_{uv} + \sum_{uv\in\binom{V}{2}\setminus E}(1 - x_{uv})$ be the LP disagreement for node u. There exists an MPC algorithm that computes a clustering $\mathcal{C}$ such that for any node u, we have*

$$\mathrm{cost}_{\mathcal{C}}(u) \leq (5 + 55\epsilon)y_u.$$

This algorithm always succeeds but terminates in $O(\log^3 n/\epsilon)$ rounds with high probability and requires $O(n^\delta)$ memory per machine. Moreover, let $K = E \cup \{uv \in \binom{V}{2} \setminus E \mid x_{uv} < 1\}$ be the set of $+$edges and $-$edges whose LP value is less than 1. The algorithm uses a total memory of $O(|K| \log n)$ and a total work of $O(|K| \log^3 n/\epsilon)$.

The $O(\log^3 n)$ round in the above theorem might not be desirable for many applications. Our next result shows that we can reduce the number of rounds to $O(1)$, albeit with a worse $O(1)$ approximation ratio:

Theorem 1.5. *Let $\epsilon \in (0,1)$ be a constant. There exists a randomized MPC algorithm in the strictly sublinear regime that, given a correlation clustering instance $G = (V, E)$, in $\mathbf{O(1)}$ rounds outputs a clustering that is simultaneously a $(359 + \epsilon)$-approximation, for all monotone symmetric norms. This algorithm succeeds with high probability, and uses a total memory of $\tilde{O}(m/\epsilon^2)$ and a total work of $\tilde{O}(m/\epsilon^2)$.*

Overall, relative to [DMN24], our algorithms demonstrate the following improvements.

1. We generalize the family of norms for the simultaneous approximation from ℓ_p norms to all monotone symmetric norms.
2. We obtain a simpler construction, which leads to a much smaller approximation ratio. Using a result from [CS19], to simultaneously approximate all monotone symmetric norms, it suffices to approximate all top-k norms: the top-k norm of a non-negative vector is the sum of its largest k coordinates. Though being more general mathematically, the top-k norms are more convenient to deal with compared to ℓ_p norms.

[2]As usual, we use $\tilde{O}(\cdot)$ to hide a poly-logarithmic factor in the input size.

3. We can make our algorithm run in nearly linear time. This is the first nearly-linear time simultaneous $O(1)$-approximation algorithm for the problem, even when we restrict to ℓ_p norms. In contrast, the algorithm of [DMN24] runs in nearly linear time only when the graph G has $O(1)$ maximum degree.

4. We can make our algorithm run in the MPC model with $O(1)$ rounds. Our work is the first to consider the problem in the MPC model.

1.2 Overview of Techniques

We then discuss our techniques for each of our main results.

Polynomial Time Construction of Simultaneous $O(1)$-Approximation for All Symmetric Norms By [CS19], we can reduce the problem of approximating all monotone symmetric norms to approximating all top-k norms. We then construct a fractional solution x, which is a metric over V with range $[0, 1]$, such that the fractional disagreement vector for x has top-k norm at most $12.66 \cdot \mathrm{opt}_k$ for any $k \in [n]$, where opt_k is the cost of the optimum clustering under the top-k norm. Then, we can use the 5-approximate rounding algorithm of KMZ [KMZ19], to obtain a simultaneous 63.3-approximation for all top-k norms. The KMZ rounding algorithm has two crucial properties that we need: it does not depend on k and it achieves a per-vertex guarantee.

We elaborate more on how to construct the metric $x : \binom{V}{2} \to [0, 1]$. A natural idea to assign the LP values, that was used by [DMN24], is to set x_{uv} based on the intersection of the neighborhood between u and v. Intuitively, the more common neighbors two nodes share, the closer they should be. A straightforward approach to implementing this idea is to set $x_{uv} = 1 - \frac{|N(u)\cap N(v)|}{\max(d(u),d(v))}$, where $N(u)$ denotes the neighboring nodes of u in G and $d(u) = |N(u)|$ denotes the degree of u; it is convenient to assume $u \in N(u)$. This approach works for the top-1 norm (i.e., the ℓ_∞ norm) as discussed in [DMN23], but fails for the top-n norm (i.e., the ℓ_1 norm). Consider a star graph, where the optimal clustering under the top-n norm has a cost of $n - 2$. This approach will assign $x_{uv} = 1 - \frac{1}{2} = 1/2$ for all $-$edges, leading to an LP cost of $\Theta(n^2)$ and a gap of $\Omega(n)$. [DMN24] addressed the issue by rounding up LP values to 1 for $-$edge, if for a given node, its total $-$edges LP disagreement is larger than the number of its $+$edges. After the transformation, the triangle inequalities are only satisfied approximately, but this can be handled with $O(1)$ loss in the approximation ratio.

We address this issue using a different approach, that is inspired by the pre-clustering technique in [CALM$^+$21]. We first preprocess the graph G by removing edges $uv \in E$ for which $|N(u)\cap N(v)|$ is small compared to $\max\{d(u), d(v)\}$. Let the resulting graph be H. We then set our LP values as $x_{uv} = 1 - \frac{|N_H(u)\cap N_H(v)|}{\max\{d(u),d(v)\}}$ if $u \neq v$, where $N_H(u)$ is the set of neighbors of u in H. We show that this solution is a 12.66-approximation for all top-k norms simultaneously. When compared to [DMN24], in addition to the improved approximation ratio, we obtain a considerably simpler analysis.

Implementation of Algorithm in Nearly-Linear Time and in MPC Model We then proceed to discuss our techniques to improve the running time of the algorithm to nearly-linear. The algorithm contains two parts: the construction of the fractional solution x and the rounding procedure. We discuss the two procedures separately.

Constructing x in nearly linear time poses several challenges. First, the construction of the subgraph H requires us to identify edges $uv \in E$ with small $|N(u) \cap N(v)|$. Second, we can not explicitly assign x values to all $-$edges. Finally, to compute x_{uv}, we need to compute $|N_H(u) \cap N_H(v)|$.

The first and third challenges can be addressed through sampling, with an $O(\log n)$ factor loss in the running time. To avoid considering too many $-$edges, we only consider $-$edges with length at most $1-\epsilon$. Consequently, we only need to consider $-$edges whose other endpoints share at least an ϵ fraction of neighbors with u. Given that each neighbor of u in H has degree similar to u, we demonstrate that there will be at most $O(d(u)/\epsilon)$ $-$edges to consider for each node u. Overall, there will be $\tilde{O}(m \cdot \mathrm{poly}(1/\epsilon))$ $-$edges for which we need to explicitly assign x values. Moreover, the nearly-linear time algorithm for the construction of x can be naturally implemented in the MPC model, with $O(1)$ number of rounds.

Then we proceed to the rounding algorithm for x. We are explicitly given the x values for $+$edges, and for nearly-linear number of $-$edges. For other $-$edges, their x values are 1. The KMZ algorithm works as follows: in each round, the algorithm selects a node u as the cluster center and then includes a ball with some radius, meaning the algorithm includes all nodes v such that $x_{uv} \leq$ radius into the cluster, removes the clustered nodes, and repeats the process on the remaining nodes. The rounding algorithm can be easily implemented in nearly-linear time using a priority queue structure. This leads to a nearly-linear time simultaneous $O(1)$-approximation for correlation clustering for all monotone symmetric norms.

The challenge to implement the algorithm in MPC model is the sequential nature of the algorithm. [KMZ19] observes that in each round, if we select the nodes that maximize $L(u) = \sum_{x_{uv} \leq r}(r - x_{uv})$ as cluster center, we can effectively bound each node's algorithmic cost, where r is the final ratio. However, choosing a node that maximizes some target inherently makes the process sequential. Our key observation is that, instead of selecting the node that maximizes $L(u)$, we can allow some approximation. This strategy still permits achieving a reasonable approximate ratio with an additional $1+\epsilon$ overhead while allowing the selection of multiple nodes as cluster centers, thereby parallelizing the rounding process. In each round, there might be several candidate cluster centers with conflicts. To resolve these conflicts, we employ the classical Luby's algorithm [Lub85, CDK14] to find a maximal independent set, ensuring that none of the cluster centers have conflicts.

Organization We give some preliminary remarks in Section 2. In Section 3, we describe our simultaneous $O(1)$-approximation algorithm for correlation clustering for all top-k norms. The reduction from any monotone symmetric norm to top-k norms is deferred to Appendix A. Combining the results leads to a simultaneous $O(1)$-approximation algorithm for all monotone symmetric norms. Then in Section 4 and 5, we show how we can run the algorithm in the MPC model with nearly linear work. In particular, Section 4 and 5 discuss how to solve the LP and round the LP solution in the MPC model, respectively. The constant round MPC algorithm is described in Section 6. Theorem 1.2, 1.3, 1.4 and 1.5 are proved in Section 3, 5, 5 and 6 respectively.

2 Preliminaries

The input to correlation clustering is a complete graph whose edges are partitioned into $+$edges and $-$edges. We shall use the graph $G = (V, E)$ of $+$edges to denote an instance. Let $n = |V|$ and $m = |E|$. For simplicity, we assume E contains all the n self-loops $uu, u \in V$. So, E is the set of $+$edges, and $\binom{V}{2} \setminus E$ is the set of $-$edges. The graph G is fixed in most part of the paper.

For any graph $H = (V_H, E_H)$, and any vertex $v \in V_H$, let $N_H(u) = \{v \in V_H \mid uv \in E_H\}$. For any vertex $u \in V_H$ and any subset $S \subseteq V_H$, we define $d_H(u, S) = \sum_{v \in S} \mathbb{1}(uv \in E_H)$ as the number of edges between u and S. We simply use $d_H(u)$ for $d_H(u, V_H)$. When the graph H is the input graph G, we omit the subscript. So we use $N(u)$ for $N_G(u)$ and $d(u)$ for $d_G(u)$. Notice that $u \in N(u)$ and $d(u) = |N(u)| \geq 1$ for every $u \in U$. For the input graph $G = (V, E)$ and any two vertex $u, v \in V$, we define $M_{uv} = \max\{d(u), d(v)\}$ as the maximum degree of u and v for simplicity, as this notion will be frequently used. For any two sets X and Y, we denote their symmetric difference by $X \Delta Y$. Algorithms are parameterized by constants $\beta(0 < \beta < 1), \lambda(0 < \lambda < 1)$ that will be determined later.

A norm on n-dimensional non-negative vectors is a function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$ satisfying $f(\alpha x) = \alpha f(x)$ for every real $\alpha \geq 0$ and $x \in \mathbb{R}^n_{\geq 0}$, and $f(x + y) \leq f(x) + f(y)$ for every $x, y \in \mathbb{R}^n_{\geq 0}$. We say a norm $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$ is monotone if for every $x, y \in \mathbb{R}^n_{\geq 0}$ with $x \leq y$, we have $f(x) \leq f(y)$. We say f is symmetric if $f(x) = f(x')$ for every $x, x' \in \mathbb{R}^n_{\geq 0}$ such that x' is a permutation of x. We say f is the top-k norm for an integer $k \in [n]$ if $f(x)$ is equal to the sum of the k largest coordinates of x. Chakrabarty and Swamy [CS19] showed that any monotone and symmetric norm can be written as the maximum of many ordered norms. This leads to the following lemma which reduces the monotone-symmetric norms to top-k norms. For completeness, we defer its proof to Appendix A.

Lemma 2.1. *For any integer $k \in [n]$, if an algorithm returns a single clustering $\mathcal{C}_{ALG}$ that is simultaneously a ρ-approximation for all top-k norm objectives, then $\mathcal{C}_{ALG}$ is a ρ-approximation for any monotone and symmetric norm $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_+$.*

For a fixed clustering $\mathcal{C}$, we already defined the disagreement vector of $\mathcal{C}$ as $\mathrm{cost}_{\mathcal{C}} \in \mathbb{Z}^n_{\geq 0}$, with $\mathrm{cost}_{\mathcal{C}}(u)$ for every $u \in V$ being the number of edges incident to u that are in disagreement w.r.t $\mathcal{C}$. Given an integer k, and a clustering $\mathcal{C}$, we denote the top-k value by $\mathrm{cost}^k_{\mathcal{C}} = \max_{T \subseteq V, |T| = k} \sum_{u \in T} \mathrm{cost}_{\mathcal{C}}(u)$. Similarly, for any fractional vector $(x_{uv})_{u,v \in V}$, we denote $\mathrm{cost}_x(u) = \sum_{uv \in E} x_{uv} + \sum_{uv \in \binom{V}{2} \setminus E} (1 - x_{uv})$ as the disagreement for u with respect to x. The top-k value of x is defined as $\mathrm{cost}^k_x = \max_{T \subseteq V, |T| = k} \sum_{u \in T} \mathrm{cost}_x(u)$.

We will use the following theorem from [KMZ19]:

Theorem 2.2. *Let $G = (V, E)$ be a correlation clustering instance, and $x \in [0, 1]^{\binom{V}{2}}$ be a metric over V with range $[0, 1]$. There is a polynomial time algorithm that, given G and x, outputs a clustering $\mathcal{C}$ of V such that $\mathrm{cost}_{\mathcal{C}}(u) \leq 5 \cdot \mathrm{cost}_x(u)$ for every $u \in V$.*

We will use the following well-known concentration inequalities.

Theorem 2.3 (Chernoff Bound)**.** *Let $X_1, X_2, ..., X_k$ be independent random variables taking values in $\{0, 1\}$. Let $X = \sum_i X_i$ be the sum of these k random variables. Then the following inequalities hold:*

(2.3a) For any $\epsilon \in (0, 1)$, if $E[X] \leq U$, then $\Pr[X \geq (1 + \epsilon)U] \leq \exp(-\epsilon^2 U/3)$.

(2.3b) For any $\epsilon \in (0, 1)$, if $E[X] \geq U$, then $\Pr[X \leq (1 - \epsilon)U] \leq \exp(-\epsilon^2 U/2)$.

3 Simultaneous $O(1)$-Approximation for Top-k Norms

In this section, we describe our simultaneous 63.3-approximation for correlation clustering for all top-k norms. The algorithm described in this section runs in polynomial time. It first constructs an

Algorithm 1 Construction of norm-oblivious solution x to metric LP.
Input: Graph $G = (V, E)$
Output: $(x_{uv})_{u,v\in V}$

```
1: function AllNormCC($G = (V, E)$)
2:     let $E_H = \{uv \in E : |N(u)\Delta N(v)| \le \beta \cdot M_{uv}\}$ and $H = (V, E_H)$
3:     let $x_{uv} \leftarrow 1 - \frac{|N_H(u)\cap N_H(v)|}{\max(d(u),d(v))}$ for every $uv \in \binom{V}{2}$ and $x_{uu} = 0$ for every $u \in V$
4: end function
```

LP solution to the top-k linear program using a combinatorial procedure. Crucially, the construction does not depend on the value of k. We show that the solution has cost 12.66 times the optimum cost under the top-k norm, for any integer $k \ge 1$. Then we use the rounding algorithm of [KMZ19] to round the LP solution to an integral one. As it gives a vertex-by-vertex 5-approximation guarantee, this leads to a 63.3-approximation for the top-k norm for any k.

The LP for minimizing the top-k norm of the clustering is given in LP (1).

$$\min \qquad \text{cost}_x^k \qquad \text{s.t.} \tag{1}$$

$$x_{uv} + x_{uw} \ge x_{vw},\ \forall u, v, w \in V \quad (2) \qquad x_{uv} \in [0, 1],\ \forall u, v \in V \quad (3) \qquad x_{uu} = 0,\ \forall u \in V \quad (4)$$

In the correspondent integer program, x_{uv} for every $u, v \in V$ indicates if uv is separated or not. We view uv as an unordered pair and thus x_{uv} and x_{vu} are the same variable. So $(x_{uv})_{u,v\in V}$ is a metric with distances in $\{0, 1\}$, which is relaxed to $[0, 1]$ in the linear program. This is captured by constraints (2), (3) and (4). Notice that $\text{cost}_x(u)$ for any $u \in V$ is a linear function of the x variables. The top-k norm of the fractional clustering is defined by $\text{cost}_x^k = \max_{S\subseteq V:|S|=k} \sum_{u\in S} \text{cost}_x(u)$. This could be captured by introducing a variable z and constraints $z \ge \sum_{u\in S} \text{cost}_x(u)$ for any $S \subseteq V$ of size k, and setting z to be the objective to minimize. For simplicity, we use the form as described. Despite having an exponential number of constraints, the LP can be solved efficiently as there is a simple separation oracle. Moreover, we use a combinatorial algorithm to construct a solution x, and thus the algorithm does not solve the LP.

3.1 Algorithm

The algorithm for constructing the LP solution x is given in Algorithm 1. It depends on the parameter $\beta \in (0, 1)$, whose value will be specified later. During the process, we construct a subgraph H by removing any edge $uv \in E$ where u and v have significantly different neighbors. We then set x_{uv} as $1-\frac{|N_H(u)\cap N_H(v)|}{M_{uv}}$ if $u \ne v$ and $x_{uv} = 0$ otherwise. Recall that $M_{uv} = \max\{d(u), d(v)\}$ is the maximum degree for any nodes u and v in graph G. Intuitively, we treat +edges as indicators of whether two nodes belong to the same cluster. The first step is to remove edges that should not be in the same cluster. The second step ensures that the more common neighbors two nodes have, the closer their distance should be.

In the remaining part of this section, we will show

Lemma 3.1. *Let k be any integer in $[n]$. Algorithm 1 outputs a feasible solution $(x_{uv})_{u,v\in V}$ for (1) such that for any k, we have*

$$\text{cost}_x^k \le 12.66 \cdot \text{opt}^k,$$

where opt^k *is the cost of the optimum solution under the top-*k *norm.*

Proof. TOPROVE 1 ◻

We will first show that our x is feasible in Section 3.2, then we will bound the approximate ratio in Section 3.3.

3.2 The validity of x to LP (1)

To show that x is a valid solution to LP (1), it suffices to prove that it is a metric over V with range $[0,1]$. Moreover, (3) and (4) hold trivially. Therefore, it remains to show that the triangle inequality (i.e., constraint (2)) is satisfied:

Lemma 3.2. *For any* $u, v, w \in V$*, we have* $x_{uv} + x_{uw} \geq x_{vw}$*.*

Proof. TOPROVE 2 ◻

3.3 Bounding the Top-k Norm Cost of x

In this section, we compare the top-k norm cost of x to opt^k.

Notations We fix the integer $k \in [n]$. Let $\mathcal{C}$ be the clustering that minimizes the top-k norm of disagreement vector, but our analysis works for any clustering. For every $v \in V$, let $C(v)$ be the cluster in $\mathcal{C}$ that contains v.

For every $u \in V$, let $\mathrm{cost}^+_{\mathcal{C}}(u), \mathrm{cost}^-_{\mathcal{C}}(u)$ and $\mathrm{cost}_{\mathcal{C}}(u)$ respectively be the number of +edges, −edges and edges incident to u that are in disagreement in the clustering $\mathcal{C}$. Recall $\mathrm{cost}^k_{\mathcal{C}} = \max_{S\subseteq V:|S|=k} \sum_{u\in S} \mathrm{cost}_{\mathcal{C}}(u)$ is the top-k norm cost of the clustering $\mathcal{C}$, thus we have $\mathrm{opt}^k = \mathrm{cost}^k_{\mathcal{C}}$.

Let U be the set of k vertices u with the largest $\mathrm{cost}_x(u)$ values. So, $\mathrm{cost}^k_x = \sum_{u\in U} \mathrm{cost}_x(u)$. In order to provide a clear demonstration, we divide all of the edges in G into five parts. First, we separate out the parts that are easily constrained by $\mathrm{cost}^k_{\mathcal{C}}$. Let φ^+_1 be the set of +edges that are cut in $\mathcal{C}$, and φ^-_1 be the set of −edges that are not cut in $\mathcal{C}$. For the remaining +edges in E that are not cut in $\mathcal{C}$, it is necessary to utilize the properties of +edges in E_H. To this end, let φ^+_2 be the set of +edges in $E \setminus E_H$ that are not cut in $\mathcal{C}$, and φ^+_3 be the set of +edges in E_H that are not cut in $\mathcal{C}$. Finally, we define φ^-_2 as the set of −edges that are cut in $\mathcal{C}$. Formally, we set

$$\begin{aligned}
\varphi^+_1 &:= \{uv \mid uv \in E, C(u) \neq C(v)\},\\
\varphi^+_2 &:= \{uv \mid uv \in E \setminus E_H, C(u) = C(v)\},\\
\varphi^+_3 &:= \{uv \mid uv \in E_H, C(u) = C(v)\},\\
\varphi^-_1 &:= \{uv \mid uv \in \binom{V}{2} \setminus E, C(u) = C(v)\},\\
\text{and} \quad \varphi^-_2 &:= \{uv \mid uv \in \binom{V}{2} \setminus E, C(u) \neq C(v)\}.
\end{aligned}$$

Notice that φ^+_3 contains all the self-loops. For every $(i,j) \in \{(1,+),(2,+),(3,+),(1,-),(2,-)\}$ and $u \in V$, we let $\varphi^j_i(u)$ be the set of pairs in φ^j_i incident to u. We use $\phi^j_i(u) = \{v : uv \in \varphi^j_i(u)\}$ to denote the end-vertices of the edges in $\varphi^j_i(u)$ other than u; so $|\phi^j_i(u)| = |\varphi^j_i(u)|$. We let $y^j_i(u)$ denote the cost of edges in $\varphi_i^j(u)$ in the solution x. For every $(i,j) \in \{(1,+),(2,+),(3,+),(1,-),(2,-)\}$, we define $f_i^j = \sum_{u\in U} y_i^j(u)$. Therefore, the top-$k$ norm cost of x is $f_1^+ + f_2^+ + f_3^+ + f_1^- + f_2^-$.

With the notations defined, we can proceed to the analysis. Prior to this, several propositions are presented, which will prove useful in the following analysis. We start with the property of edges in $E \setminus E_H$.

Lemma 3.3. *For every $uv \in \varphi_2^+$, we have $\mathrm{cost}_{\mathcal{C}}(u) + \mathrm{cost}_{\mathcal{C}}(v) \geq \beta \cdot M_{uv}$.*

Proof. TOPROVE 3 □

Then, we show that edges in E_H have similar degrees.

Lemma 3.4. *For every $uv \in E_H$, we have $(1-\beta)\cdot d(v) \leq d(u) \leq \frac{1}{1-\beta}\cdot d(v)$.*

Proof. TOPROVE 4 □

As we are about to analyze the top-k norm objective, we can bound the cost in the solution x using coefficients of $\mathrm{cost}_{\mathcal{C}}$. This key observation can be formally demonstrated as follows:

Claim 3.5. *Let $c \in \mathbb{R}^n_{\geq 0}$ be a vector and $\alpha > 0$ satisfying that $|c|_\infty \leq \alpha$ and $|c|_1 \leq \alpha k$. Then we have*

$$\sum_{r\in V} c(r)\cdot \mathrm{cost}_{\mathcal{C}}(r) \leq \alpha \cdot \mathrm{cost}_{\mathcal{C}}^k.$$

Proof. TOPROVE 5 □

From the definitions of f_1^+ and f_1^-, we can see that it can be constrained by $\mathrm{cost}_{\mathcal{C}}^k$.

Claim 3.6. $f_1^+ + f_1^- \leq \mathrm{cost}_{\mathcal{C}}^k$.

Proof. TOPROVE 6 □

To bound the cost of the remaining +edges, we separately analyze the cost coefficients of vertices in f_2^+ and f_3^+.

Lemma 3.7. *There exists a vector $c_2^+ \in \mathbb{R}^n_{\geq 0}$ with the following properties:*

(3.7a) $f_2^+ \leq \sum_{r\in V} c_2^+(r)\cdot \mathrm{cost}_{\mathcal{C}}(r)$.

(3.7b) $c_2^+(r) \leq \frac{2}{\beta}\cdot \frac{|\varphi_2^+(r)|}{d(r)}$, *for every $r \in V$.*

(3.7c) $|c_2^+|_1 \leq \frac{2}{\beta}\sum_{u\in U} \frac{|\varphi_2^+(u)|}{d(u)}$.

Proof. TOPROVE 7 □

Following the analysis of the cost coefficients c_2^+ of f_2^+, we analyze the cost coefficients of f_3^+ in edge set E_H.

Lemma 3.8. *There exists a vector $c_3^+ \in \mathbb{R}^n_{\geq 0}$ with the following properties:*

(3.8a) $f_3^+ \leq \sum_{r\in V} c_3^+(r)\cdot \mathrm{cost}_{\mathcal{C}}(r)$.

(3.8b) $c_3^+(r) \le 2\left(\frac{|\varphi_2^+(r)|\cdot|\varphi_3^+(r)|}{\beta\cdot d^2(r)} + \frac{|\varphi_2^+(r)|}{\beta\cdot d(r)} + \frac{|\varphi_3^+(r)|}{d(r)}\right)$, *for every* $r \in V$.

(3.8c) $|c_3^+|_1 \le 2\sum_{u\in U}\left(\frac{|\varphi_2^+(u)|\cdot|\varphi_3^+(u)|}{\beta\cdot d^2(u)} + \frac{|\varphi_3^+(u)|}{\beta\cdot d(u)} + \frac{|\varphi_3^+(u)|}{d(u)}\right)$.

Proof. TOPROVE 8 □

With Lemma 3.7 and Lemma 3.8, we can then bound $f_2^+ + f_3^+$:

Lemma 3.9. $f_2^+ + f_3^+ \le \frac{4}{\beta}\cdot \mathrm{cost}_{\mathcal{C}}^k$.

Proof. TOPROVE 9 □

For the cost of the remaining −edges, i.e. f_2^-, we also analyze the cost coefficients of each vertex.

Lemma 3.10. *There exists a vector* $c_2^- \in \mathbb{R}^n_{\ge 0}$ *with the following properties:*

(3.10a) $f_2^- \le \sum_{r\in V} c_2^-(r)\cdot \mathrm{cost}_{\mathcal{C}}(r)$.

(3.10b) $c_2^-(r) \le \frac{2}{1-\beta}$, *for every* $r \in V$.

(3.10c) $|c_2^-|_1 \le \frac{2k}{1-\beta}$.

Proof. TOPROVE 10 □

With Lemma 3.10, we can then bound f_2^-.

Lemma 3.11. $f_2^- \le \frac{2}{1-\beta}\cdot \mathrm{cost}_{\mathcal{C}}^k$.

Proof. TOPROVE 11 □

So the final ratio for Algorithm 1 is

$$1 + \frac{4}{\beta} + \frac{2}{1-\beta}.$$

Let $\beta = 0.5858$, then the ratio is at most 12.66. This finishes the proof of Lemma 3.1.

4 Implementing Algorithm 1 in Nearly Linear Time

In this section, we show how to run Algorithm 1 approximately in nearly linear time. Indeed, the algorithm can be implemented in MPC model with $O(1)$ rounds. More precisely, we will show the following theorem:

Theorem 4.1. *Let* $\epsilon > 0$ *and* $\delta > 0$ *be small enough constants. Given a graph* $G = (V, E)$*, there exists an MPC algorithm that computes a solution* $\{\tilde{x}_{uv}\}_{u,v\in V}$ *such that*

1. *For any integer* $k \in [n]$*, we have* $\mathrm{cost}_{\tilde{x}}^k \le (12.66+\epsilon)\mathrm{opt}^k$.

2. *For any* $u, v, w \in V$*, we have* $\tilde{x}_{uv} + \tilde{x}_{uw} + \epsilon \ge \tilde{x}_{vw}$.

The algorithm succeeds with probability at least $1 - 1/n$*. Moreover, the algorithm runs in* $O(1)$ *rounds, has a total work of* $\tilde{O}(m/\epsilon^6)$*, requires* $O(n^\delta)$ *memory per machine and a* $\tilde{O}(m/\epsilon^6)$ *total memory.*

We give the nearly linear time implementation of Algorithm 1 in Algorithm 2. Line 2 constructs the graph H efficiently. Line 3-6 find the set K of $-$edges we want to assign LP value. For any $-$edge $uv \notin K$, we will simply set its LP value as 1. Last, Line 7 to Line 14 is to set up $\tilde{x}_{uv}$ satisfying the conditions in 4.1. In the remaining of this section, we will discuss these three parts in detail.

Algorithm 2 Nearly Efficient Algorithm for Algorithm 1.
Input: Graph $G = (V, E), \epsilon \in (0, 1)$
Output: $\{\tilde{x}_{uv}\}_{u,v \in V}$ such that is $(12.66 + 26\epsilon)$-approximate and satisfy approximate triangle inequality.

```
1:  function AllNormCCBySampling($G = (V, E)$)
2:      Construct subgraph $H = (V, E_H \subseteq E)$ using Corollary 4.3
3:      for every $u \in V$ do                                         ▷ Compute set $K$
4:          $S_u \leftarrow \emptyset$
5:          for every $v \in d_H(u)$ do: add $v$ to $S_u$ with probability $\min(\frac{8 \log n}{\epsilon d_H(u)}, 1)$
6:      $K \leftarrow \{uv \notin E : \exists w \in S_u, vw \in E_H \text{ or } \exists w \in S_v, uw \in E_H\}$
7:      $\tau \leftarrow \frac{400 \log n}{\epsilon^5}$                                ▷ Compute the $\tilde{x}_{uv}$ values
8:      for every $j \in [1, \log n]$ do
9:          $S(j) \leftarrow \emptyset$
10:         for $v \in V$ do: add $v$ to $S(j)$ with probability $\min(\tau/2^j, 1)$
11:         for every $uv \in E \cup K$ with $M_{uv} \in [2^{j-1}, 2^j)$ do: $\tilde{x}_{uv} \leftarrow 1 - \frac{2^j |N_H(u) \cap N_H(v) \cap S(j)|}{\tau \cdot M_{uv}}$
12:     for every $uv \in E$ do: if $\tilde{x}_{uv} \leq \epsilon$ then $\tilde{x}_{uv} \leftarrow 0$
13:     for every $uv \in K$ do: if $\tilde{x}_{uv} \geq 1 - \epsilon$ then $\tilde{x}_{uv} \leftarrow 1$
14:     for every $uv \in \binom{V}{2} \setminus (E \cup K)$ do: $\tilde{x}_{uv} \leftarrow 1$
15: end function
```

4.1 Line 2: Construction of Subgraph H

To construct graph H in Line 2 of Algorithm 2, we can use the sampling method in [CALM+21]:

Theorem 4.2. *[Lemma 3.11 of [CALM+21]] Let $\delta, \epsilon, \beta \in (0, 1)$ be constants. There exists an MPC algorithm that, given a graph $G = (V, E)$, outputs a "Yes/No" answer for every $uv \in E$, such that the following happens with probability at least $1 - \frac{1}{n^6}$.*

- *For every $uv \in E$ with $|N(u) \Delta N(v)| \leq \beta M_{uv}$, algorithm outputs "Yes" for uv.*
- *For every $uv \in E$ with $|N(u) \Delta N(v)| \geq (1 + \epsilon)\beta M_{uv}$, algorithm outputs "No" for uv.*

The algorithm runs in $O(1)$ rounds, with a total work of $\tilde{O}(m/\epsilon^2)$, $O(n^\delta)$ memory per machine, and $\tilde{O}(m/\epsilon^2)$ total memory.

In [CALM+21], it is required that the algorithm outputs "Yes" for $uv \in E$ if $|N(u)\Delta N(v)| \leq 0.8\beta M_{uv}$ and outputs "No" for $uv \in E$ if $|N(u)\Delta N(v)| \geq \beta M_{uv}$. However, we can replace 0.8 with any constant less than 1. Additionally, the algorithm succeeds with probability $1 - \frac{1}{n}$, but we can increase this to $1 - \frac{1}{n^c}$ for any constant c. For completeness, we prove Theorem 4.2 in Appendix B. The theorem leads to the following corollary:

Corollary 4.3. *Consider the setting in Theorem 4.2, and let $H = (V, E_H)$ with E_H being the set of edges $uv \in E$ for which the algorithm outputs "Yes". Let $x_{uv} = 1 - \frac{|N_H(u) \cap N_H(v)|}{M_{uv}}$ for every $uv \in \binom{V}{2}$ and $x_{uu} = 0$ for every $u \in V$. Then for any $k \in [1, n]$, we have $\mathrm{cost}_x^k \leq 12.66(1+\epsilon)\mathrm{opt}^k$.*

From now on, we define x_{uv} as in Corollary 4.3: $x_{uv} = 1 - \frac{|N_H(u) \cap N_H(v)|}{M_{uv}}$ for every $uv \in \binom{V}{2}$ and $x_{uu} = 0$ for every $u \in V$. Notice that Algorithm 2 does not compute or main x explicitly; it is introduced only for the sake of analysis.

4.2 Line 3-6: Construction of K

After obtaining H, there might be $O(n^2)$ −edges for which we need to assign x values in Algorithm 1. Fortunately, most of these −edges will have x values greater than $1 - \epsilon$, allowing us to simply disregard them. By doing this, we achieve an approximate triangle inequality rather than a strict triangle inequality.

The key observation is as follows: for any −edge $uv \in \binom{V}{2} \setminus E$ with $x_{uv} \leq 1 - \epsilon$, then $\frac{|N_H(u) \cap N_H(v)|}{M_{uv}} = 1 - x_{uv} \geq \epsilon$, which is equivalent to $|N_H(u) \cap N_H(v)| \geq \epsilon M_{uv}$. If we sample $O\left(\frac{\log n}{\epsilon}\right)$ nodes from $N_H(u)$, at least one node is in $N_H(u) \cap N_H(v)$ with high probability. By considering all neighbors of the sampled nodes for u, we will cover all −edges uv where $x_{uv} \leq 1 - \epsilon$. Additionally, note that for any neighbor $w \in N_H(u)$, their degrees will not differ significantly. Since we only have $O\left(\frac{\log n}{\epsilon}\right)$ nodes to consider, we will only need to consider $O\left(\frac{d(u) \log n}{\epsilon}\right)$ nodes for each node u, which is still a nearly linear number of nodes in total.

We let K be the set constructed in the algorithm.

Lemma 4.4. *$\forall uv \in \binom{V}{2} \setminus E$ with $x_{uv} \leq 1 - \epsilon, \Pr[uv \in K] \geq 1 - n^{-6}$. Moreover $|K| = O\left(\frac{m \log n}{\epsilon}\right)$ with probability at least $1 - \frac{1}{n^7}$.*

Proof. TOPROVE 12 □

4.3 Line 7-14: Computing $\tilde{x}_{uv}$ for $E \cup K$

Once we identify all edges to assign LP values to, we use the sampling method to compute x_{uv}, as described in [DMN23]. The key observation is that the size of a given set can be evaluated by sampling $O(\log n)$ elements. Special care is needed when the set is too small; hence, we set $\tilde{x}_{uv}$ to 0 or 1 if the size of the set is either too small or too large.

Algorithm Description Lines 7 to 14 describe the setup of $\tilde{x}_{uv}$. Let $\tau = \frac{\log n}{\epsilon^2}$. To evaluate $|N(u) \cap N(v)|$, we define the set $S(j)$ as a subset of nodes obtained by sampling every node in the graph independently with probability $\min(\tau/2^j, 1)$ (Line 11). We will have $O(\log n)$ different values for $j \in [1, \log n]$. Define $j_u = \max\{j \mid 2^j \leq d(u)\}$ as the maximum integer that is a power of 2 and at most $d(u)$. For any $uv \in E \cup K$, we then set

$$\tilde{x}_{uv} = 1 - \frac{2^j |N_H(u) \cap N_H(v) \cap S(j_u)|}{\tau M_{uv}}.$$

Additionally, for each $uv \in E$, if $\tilde{x}_{uv} \leq \epsilon$, we set $\tilde{x}_{uv} = 0$. For any $uv \in K$, if $\tilde{x}_{uv} \geq 1 - \epsilon$, we set $\tilde{x}_{uv} = 1$ (Lines 12-14). Our main lemma regarding the approximate LP value $\tilde{x}_{uv}$ is given below:

Lemma 4.5. *With probability at least $1 - 1/n^6$, Algorithm 2 outputs $\tilde{x}_{uv}$ such that*

(4.5a) For any $uv \in E$, we have $\tilde{x}_{uv} \le (1+\epsilon)x_{uv}$. For any $uv \in \binom{V}{2} \setminus E$, we have $1 - \tilde{x}_{uv} \le (1+\epsilon)(1 - x_{uv})$.

(4.5b) For any $u, v, w \in V$, we have $\tilde{x}_{uv} + \tilde{x}_{uw} + 3\epsilon \ge \tilde{x}_{vw}$.

The proof is similar to the sampling method proof presented in [DMN23]. The full proof of Lemma 4.5 is provided in Appendix C.

4.4 Wrapping up: Proof of Theorem 4.1

We can now prove Theorem 4.1. Once we construct the graph H in Algorithm 2, by setting $x_{uv} = 1 - \frac{|N_H(u) \cap N_H(v)|}{\max(d(u), d(v))}$, we achieve $\text{cost}_x^k \le (12.66 + 26\epsilon)\text{opt}^k$. By Lemma 4.5, we know that at the end, the algorithm outputs the approximate LP value $\tilde{x}_{uv}$. Therefore, we can bound the cost as $\text{cost}_{\tilde{x}}^k \le (1+\epsilon)\text{cost}_x^k \le (12.66 + 50\epsilon)\text{opt}^k$, where opt^k is the cost of the optimal correlation clustering solution using the top-k norm objective. The approximate triangle inequality is implied by Lemma (4.5b). The final ratio is obtained by scaling ϵ to $\epsilon/50$.

The running time of Algorithm 2 is derived from the discussion in this section. Constructing H takes $O(1)$ rounds in the MPC model and $\tilde{O}(m/\epsilon^2)$ total work. By Lemma 4.4, we know that we will only assign at most $\tilde{O}(\frac{m}{\epsilon})$ edges. Since the size of $|N_H(u) \cap N_H(v) \cap S(j_u)|$ is bounded by $\tau = O(\frac{\log n}{\epsilon^5})$, each edge takes $\tilde{O}(1/\epsilon^5)$ time to compute $\tilde{x}_{uv}$. In total, Algorithm 2 takes $\tilde{O}(\frac{m}{\epsilon^6})$ work and total memory. Note that all sampling and for-loops are naturally parallelized, so it only takes $O(1)$ rounds to output $\tilde{x}_{uv}$.

5 Rounding

We will present a nearly linear time rounding algorithm. Furthermore, our algorithm only takes $\tilde{O}(1)$ rounds in the MPC model. The purpose of this section is to show

Theorem 1.4. *Let $\epsilon \in (0,1)$ be a constant. Given a graph $G = (V, E)$ and a set of LP value $(x_{uv})_{u,v \in V}$ satisfying the approximate triangle inequality, that is, for any $u, v, w \in V$, we have $x_{uv} + x_{uw} + \epsilon \ge x_{vw}$. Let $y_u = \sum_{uv \in E} x_{uv} + \sum_{uv \in \binom{V}{2} \setminus E}(1 - x_{uv})$ be the LP disagreement for node u. There exists an MPC algorithm that computes a clustering $\mathcal{C}$ such that for any node u, we have*

$$\text{cost}_{\mathcal{C}}(u) \le (5 + 55\epsilon)y_u.$$

This algorithm always succeeds but terminates in $O(\log^3 n/\epsilon)$ rounds with high probability and requires $O(n^\delta)$ memory per machine. Moreover, let $K = E \cup \{uv \in \binom{V}{2} \setminus E \mid x_{uv} < 1\}$ be the set of $+$edges and $-$edges whose LP value is less than 1. The algorithm uses a total memory of $O(|K| \log n)$ and a total work of $O(|K| \log^3 n/\epsilon)$.

We emphasize that even if the LP values satisfy the exact triangle inequality, rather than an approximate triangle inequality, the ϵ terms in the approximate ratio will still be present. These ϵ terms arise from two sources: the approximate inequality itself and the inherent characteristics of our MPC algorithm.

Given Theorem 4.1 and Theorem 1.4, we are now able to show the main result of this paper.

Proof. TOPROVE 13 ☐

5.1 Rounding Algorithm

Assume that we are given an instance graph $G = (V, E)$ and a LP solution $(x_{uv})_{u,v \in V}$ such that $x_{uv} + x_{uw} + \epsilon \geq x_{vw}$ for any $u, v, w \in V^3$. Given a subgraph V_t and node w, radius r, define the ball centering at w with radius r as $\mathrm{Ball}_{V_t}(w, r) = \{u \mid u \in V_t, x_{wu} \leq r\}$. Define

$$L_t(w) = \sum_{u \in \mathrm{Ball}_{V_t}(w,r)} (r - x_{uw})$$

Note that $L_t(w) \geq r$ since w itself is in $\mathrm{Ball}_{V_t}(w, r)$.

Algorithm Description Our algorithm works as follows: in each round, we choose a set of cluster centers and choose the ball with radius $\frac{2}{5}$ as a cluster. More precisely. At step t, V_t is the set of unclustered vertices. We first compute $L_t^{\max}$ to ensure that for each $u \in V_t$, we have $L_t(u) < (1+\epsilon)L_t^{\max}$. For any node u, if $L_t(u) \geq L_t^{\max}$, we will add u to the set of candidate cluster centers M_t (Line 6-10). Then, we compute cluster centers by adding vertices in the M_t set to the S_t set with probability p_t, where the more vertices in M_t are in $\mathrm{Ball}(radius = \frac{2}{5})$ with each other, the smaller the probability (Line 11-14). After that, to avoid conflicts, we remove some cluster centers in S_t if they are too close to each other, and derive the final cluster center set H_t (Line 15). Let $F_t = \mathrm{Ball}_{V_t}(H_t, \frac{2}{5})$ be the nodes clustered at step t. Then we add each $u \in F_t$ to the cluster from H_t with minimum ID. We will remove the clustered nodes and repeat the above process until all vertices are clustered.

5.2 Approximate Ratio

Analysis Framework We will follow the proof of the sequential rounding algorithm [KMZ19], which is also used in [DMN23] for approximate triangle inequality. However, we suffer different difficulties that each round we choose multiple nodes whose L_t value is within $\frac{1}{1+\epsilon}$ times the maximum L_t value. Let $LP(uv)$ be the LP cost of the edge $uv : LP(uv) = x_{uv}$ if $uv \in E$ and $uv : LP(uv) = 1 - x_{uv}$ if $uv \in \binom{V}{2} \setminus E$. Let $ALG(uv) = \mathbb{1}(uv \text{ is in disagreement})$. We can define

$$\mathrm{profit}(u) = \sum_{uv \in E} LP(uv) - r \sum_{uv \in E} ALG(uv)$$

where $r = \frac{1}{5} - 2\epsilon$. We want to show that $\mathrm{profit}(u) \geq 0$ for all nodes. Let

$$\Delta E_t = \{uv : u \in F_t \text{ or } v \in F_t\}$$

be the set of edges removed at step t. Next, let

$$\mathrm{profit}_t(uv) = \begin{cases} LP(uv) - r \cdot ALG(uv), & uv \in \Delta E_t \\ 0, & otherwise \end{cases}$$

and the profit at step t is defined as

$$\mathrm{profit}_t(u) = \sum_{v \in V_t} \mathrm{profit}_t(uv) = \sum_{uv \in \Delta E_t} LP(uv) - r \cdot \sum_{uv \in \Delta E_t} ALG(uv)$$

Algorithm 3 The Rounding Algorithm.

Input: Graph $G = (V, E)$, LP solution $(x_{uv})_{u,v \in V}$ satisfying $x_{uv} + x_{uw} + \epsilon \geq x_{vw}$ for any $u, v, w \in V^3$.

Output: A function of clustering $\mathcal{C}$; i.e., $C(u) = C(v)$ iff u and v belongs to the same cluster.

```
1:  function ParallelRounding($G = (V, E), (x_{uv})$)
2:      $V_1 \leftarrow V$.
3:      $C(v) \leftarrow \emptyset$ for all $v \in V$.
4:      $t \leftarrow 1$.
5:      while $V_t \neq \emptyset$ do
6:          $M_t \leftarrow \emptyset, S_t \leftarrow \emptyset$
7:          $L_t^{\max} = \max\{r(1+\epsilon)^j | r(1+\epsilon)^j \leq \max_{w \in V_t} L_t(w)$, where $j$ is an integer$\}$
8:          for $u \in V_t$ do                                  ▷ Find cluster center candidate
9:              if $L_t(u) \geq L_t^{\max}$ then.
10:                 $M_t \leftarrow M_t \cup \{u\}$
11:         $\Delta_t \leftarrow \max_{u \in M_t} |\mathrm{Ball}_{V_t}(u, \frac{2}{5}) \cap M_t|$.
12:         $p_t \leftarrow 1/(2\Delta_t)$.
13:         for $u \in M_t$ do
14:             Add $u$ to $S_t$ with probability $p_t$          ▷ Choose cluster centers in parallel.
15:         $H_t = \{u \in S_t \mid \nexists v \in S_t \cap V_t$ such that $x_{uv} \leq \frac{2}{5}\}$   ▷ remove cluster centers that conflict.
16:         $F_t \leftarrow H_t \cup \mathrm{Ball}_{V_t}(H_t, \frac{2}{5})$            ▷ the set of settled vertices in round $t$.
17:         for $u \in F_t$ do
18:             Let $v$ be the vertex with minimum ID among $\mathrm{Ball}_{V_t}(u, \frac{2}{5}) \cap H_t$.
19:             $C(u) \leftarrow v$                               ▷ adding $u$ to the cluster of $v$.
20:         $V_{t+1} \leftarrow V_t \setminus F_t$, $t \leftarrow t + 1$
21:     return $\mathcal{C}$
22: end function
```

We will later show that $\mathrm{profit}_t(u) \geq 0$ for each t. Thus, we have

$$\mathrm{profit}(u) = \sum_t \mathrm{profit}_t(u) \geq 0$$

and we can bound the cost of the algorithm by

$$\sum_{uv \in E} ALG(uv) \leq \frac{1}{r} \sum_{uv \in E} LP(uv) \leq (5 + 55\epsilon) \sum_{uv \in E} LP(uv)$$

which gives us the following lemma,

Lemma 5.1. *Assume that $\epsilon \leq 1/20$. Given graph $G = (V, E)$ and a set of LP value $(x_{uv})_{u,v \in V}$ such that $x_{uv} + x_{uw} + \epsilon \geq x_{vw}$ for any $u, v, w \in V^3$. Let $y_u = \sum_{uv \in E} x_{uv} + \sum_{uv \in \binom{V}{2} \setminus E} (1 - x_{uv})$ be the LP disagreement for node u. Algorithm 3 outputs a clustering $\mathcal{C}$ such that for any node u,*

$$\mathrm{cost}_{\mathcal{C}}(u) \leq (5 + 55\epsilon) y_u$$

The remaining part of this section is to show $\mathrm{profit}_t(u) \geq 0$. We first show that we can bound the profit of any $-$edge uv. Recall that $r = \frac{1}{5} - 2\epsilon$.

Lemma 5.2 (Analogue of 5.3 of [KMZ19]). *Let $u, v \in V_t$ and $uv \in \binom{V}{2} \setminus E$, then* $\text{profit}_t(uv) \geq 0$.

Proof. TOPROVE 14 $\square$

To show that $\text{profit}_t(u) \geq 0$, we take a different view on adding nodes to the cluster center: We sort cluster centers H_t by the ID in increasing order. Let $w_1, w_2, \ldots$, be the cluster centers from H_t after sorting, then we will add nodes $\text{Ball}_{V_t}(w_1, \frac{2}{5})$ to w_1's cluster, remove the clustered nodes, and repeat with the next center on the unclustered nodes. The whole process terminates once we process all cluster centers.

Let w be the first cluster center such that $x_{uw} \leq \frac{4}{5}$ when we do the above process. If all nodes in $\text{Ball}_{V_t}(w, r)$ are still unclustered, [KMZ19] already shows that the cost for short +edges of u can be covered by the profit of $\text{Ball}_{V_t}(w, r)$. The difficulty of the proof mainly comes from the fact that some nodes from $\text{Ball}_{V_t}(w, r)$ may have been clustered. We will divide x_{uw} into several cases and show that in all cases, we have $\text{profit}_t(u) \geq 0$.

The first two cases are when w does not exist or $x_{uw} \leq \frac{1}{5}$, we will show that under these two cases, $\text{profit}_t(uv) \geq 0$ for any edge $uv \in \Delta E_t$. Note that by Lemma 5.2, we have $\text{profit}(uv) \geq 0$ for any $uv \in \binom{V}{2} \setminus E$. Therefore, we only need to consider +edges.

Lemma 5.3. *If the above w does not exist, then* $\text{profit}_t(uv) \geq 0$.

Proof. TOPROVE 15 $\square$

Lemma 5.4. *If $x_{uw} \leq \frac{1}{5}$, then* $\text{profit}_t(uv) \geq 0$.

Proof. TOPROVE 16 $\square$

By Lemma 5.3 and 5.4, we know that when w does not exist or $x_{uw} \leq \frac{1}{5}$, we have $\text{profit}_t(u) = \sum_{uv \in \Delta E_t} \text{profit}_t(uv) \geq 0$. For the remaining cases, we will follow the blue-print of the proof in [KMZ19]. Note that $\text{profit}_t(uv) < 0$ only if $uv \in E$ and $x_{uv} \leq r$. We will charge the negative profit to the profit of $\text{Ball}_{V_t}(w, r)$. More precisely, let

$$\text{profit}_t(u) = \underbrace{\sum_{v \in \text{Ball}_{V_t}(w,r)} \text{profit}_t(uv)}_{P_{\text{high}}(u)} + \underbrace{\sum_{v \in V_t \setminus \text{Ball}_{V_t}(w,r)} \text{profit}_t(uv)}_{P_{\text{low}}(u)}$$

We will show that $P_{\text{high}}(u) \geq (1+\epsilon)L_t(w)$ and $P_{\text{low}}(u) \geq -L_t(u)$. Note that Algorithm 3 always chooses nodes whose L_t value is within $\frac{1}{1+\epsilon}$ times of the maximum L_t value as cluster center, so the profit $P_{\text{low}}(u)$ can be covered by the profit of $\text{Ball}_{V_t}(w, r)$, which is at least $P_{\text{high}}(u)$. The following two lemmas connects $P_{\text{high}}(u)$ with $(1+\epsilon)L_t(w)$. We again have two different cases for x_{uw} and show in each case, we have $\text{profit}_t(uv) \geq (1+\epsilon)(r - x_{vw})$.

Lemma 5.5. *If $x_{uw} \in (\frac{2}{5}, \frac{4}{5}]$, for any $v \in \text{Ball}_{V_t}(w, r)$, we have* $\text{profit}_t(uv) \geq (1+\epsilon)(r - x_{vw})$.

Proof. TOPROVE 17 $\square$

The last case is $x_{uw} \in (\frac{1}{5}, \frac{2}{5}]$.

Lemma 5.6. *If $x_{uw} \in (\frac{1}{5}, \frac{2}{5}]$, for any $v \in \text{Ball}_{V_t}(w, r)$, we have* $\text{profit}_t(uv) \geq (1+\epsilon)(r - x_{vw})$.

Proof. TOPROVE 18 □

Once we bound the profit for each edge uv such that $v \in \mathrm{Ball}_{V_t}(w, r)$, we now are able to lower bound the $P_{\text{high}}(u)$.

Lemma 5.7. *If $x_{uw} \in (\frac{1}{5}, \frac{4}{5}]$, then $P_{high}(u) \geq (1+\epsilon)L_t(w)$.*

Proof. TOPROVE 19 □

We still have to bound the profit of $P_{\text{low}}(u)$, this has been shown in [KMZ19] and [DMN23]. We give the corresponding lemma for completeness.

Lemma 5.8. *If $x_{uw} \in (\frac{1}{5}, \frac{4}{5}]$, then $P_{low}(u) \geq -L_t(u)$.*

Proof. TOPROVE 20 □

Combining the profit for $P_{\text{high}}(u)$ and $P_{\text{low}}(u)$, we can deduce that the profit for any node u is non-negative when $x_{uw} \in (\frac{1}{5}, \frac{4}{5}]$

Lemma 5.9. *If $x_{uw} \in (\frac{1}{5}, \frac{4}{5}]$, then $\mathrm{profit}_t(u) \geq 0$.*

Proof. TOPROVE 21 □

5.3 Running time

Lemma 5.10. *At some round s, let $L_s^{\max}$ and Δ_s be the values we set up at lines 7 and 11. Then, after $O(\log n)$ rounds, at round $e = s + O(\log n)$, with high probability, either $L_e^{\max} < L_s^{\max}$ or $\Delta_e \leq \frac{\Delta_s}{2}$.*

Proof. TOPROVE 22 □

The next lemma gives us an upper bound of the number of rounds,

Lemma 5.11. *Algorithm 3 terminates after at most $O(\log^3 n/\epsilon)$ rounds, w.h.p.*

Proof. TOPROVE 23 □

Now we can prove the main theorem in this section.

Proof. TOPROVE 24 □

6 A Constant Round MPC Algorithm

We will show Theorem 1.5 in this section. We repeat for convenience,

Theorem 1.5. *Let $\epsilon \in (0, 1)$ be a constant. There exists a randomized MPC algorithm in the strictly sublinear regime that, given a correlation clustering instance $G = (V, E)$, in $\mathbf{O(1)}$ rounds outputs a clustering that is simultaneously a $(359 + \epsilon)$-approximation, for all monotone symmetric norms. This algorithm succeeds with high probability, and uses a total memory of $\tilde{O}(m/\epsilon^2)$ and a total work of $\tilde{O}(m/\epsilon^2)$.*

6.1 Algorithm

Theorem 1.3 gives us an $O(\log^3 n)$ rounds MPC algorithm. The bottleneck is the rounding procedure. To achieve a constant rounds MPC algorithm, instead of setting up the LP and rounding, we use the pre-clustering algorithm from [CALM+21], which is very useful for ℓ_1-norm correlation clustering. We show that the pre-clustering algorithm can also provide an $O(1)$-approximate ratio for all monotone symmetric norms simultaneously.

Algorithm Description The algorithm from [CALM+21] is parameterized by β and λ. It has three steps:

1. The first step is the same as the first step of Algorithm 1, where we compute the graph H (Line 2).
2. The algorithm marks a node as light if it loses more than a λ fraction of its neighbors in the first step. Otherwise, it marks the node as heavy. The algorithm removes all edges between two light nodes in H (Line 3 - Line 6).
3. The last step is to output the connected components F of the final graph.

Algorithm 4 Pre-clustering – Algorithm 1 in [CALM+21].
Input: Graph $G = (V, E)$,
Output: Clustering F.

```
1: function PreClustering($G = (V, E)$)
2:     Let $E_H = \{uv \in E : |N(u)\Delta N(v)| \leq \beta \cdot \max(d(u), d(v))\}$ and $H = (V, E_H)$
3:     for $v \in V$ do
4:         if $d_H(v) < (1 - \lambda)d(v)$ then Mark it as light
5:         else   Mark it as Heavy
6:     Let $E_{\tilde{G}} = \{uv \in E_H : u$ or $v$ is heavy$\}$ and $\tilde{G} = (V, E_{\tilde{G}})$
7:     Compute its connected components on $\tilde{G}$, denoted as $F$, return $F$.
8: end function
```

The main reason we can achieve a constant rounds MPC algorithm is the simplicity of steps 2 and 3. [CALM+21] already showed that Algorithm 4 can be implemented within $O(1)$ rounds and is $O(1)$-approximate for correlation clustering under ℓ_1-norm objective. We extend their proof for approximate ratio and show that Algorithm 4 outputs an $O(1)$-approximate clustering F for any top-k norm objective. More concretely, we have the following lemma:

Lemma 6.1. *Assume that $8\beta + \lambda \leq 3/8$. Given any graph G, Algorithm 4 outputs a clustering F such that for any integer $k \in [n]$, we have*

$$\mathrm{cost}_F^k \leq \left(\frac{3}{\beta} + \frac{1}{\lambda} + \frac{1}{\beta\lambda} + 8\right) \cdot \mathrm{opt}^k$$

where opt^k is the cost of the optimum solution under the top-k norm.

Proof. TOPROVE 25 □

6.2 Approximate Ratio

The proof follows the blueprint of the proof in [CALM+21] and resembles the proof used to bound the approximate ratio for Algorithm 1 in Section 3.3. We will first bound the approximate ratio loss for steps 1 and 2, then we compare the final clustering F with the optimal top-k clustering in the graph where we remove all edges deleted in steps 1 and 2.

Notations We fix the integer $k \in [n]$. $\mathcal{C}$ is the clustering that minimizes the top-k norm of the disagreement vector. For every $v \in V$, $C(v)$ is the cluster in $\mathcal{C}$ that contains v. Let U be the set of k vertices u with the largest $\mathrm{cost}_F(u)$ values. So, the top-k norm of F is $\sum_{u \in U} \mathrm{cost}_F(u)$.

Similarly, we divide all +edges in G into three parts. First, we separate out the parts easily constrained by $\mathrm{cost}^k_{\mathcal{C}}$. Let φ_1^+ be the set of +edges in E_G that are cut in $\mathcal{C}$. For the remaining +edges in E_G that are not cut in $\mathcal{C}$, we divide them into two categories depending on when we remove the edge. Let φ_2^+ be the set of +edges in $E_G \setminus E_H$ that are not cut in $\mathcal{C}$, and φ_4^+ be the set of +edges in $E_H \setminus E_{\tilde{G}}$ that are not cut in $\mathcal{C}$. In other words, φ_2^+ and φ_4^+ are edges that are not cut in $\mathcal{C}$ and removed at steps 1 and 2. respectively. Formally, we set

$$\begin{aligned}
\varphi_1^+ &:= \{uv \mid uv \in E_G, C(u) \neq C(v)\}, \\
\varphi_2^+ &:= \{uv \mid uv \in E_G \setminus E_H, C(u) = C(v)\}, \\
\varphi_4^+ &:= \{uv \mid uv \in E_H \setminus E_{\tilde{G}}, C(u) = C(v)\},
\end{aligned}$$

For every $i \in \{1,2,4\}$ and $u \in V$, we let $\varphi_i^+(u)$ be the set of pairs in φ_i^+ incident to u. For every $i \in \{1,2,4\}$, we define $g_i^+ = \sum_{u \in U} |\varphi_i^+(u)|$. Therefore, when we remove edges from the graph G, we will lose at most $g_1^+ + g_2^+ + g_4^+$ number of edges for nodes from U.

We have $g_1^+ \leq \mathrm{cost}^k_{\mathcal{C}}$. Note that in Lemma 3.7, we actually show a stronger argument, $\sum_{u \in U} |\varphi_2^+(u)|$ is bounded by $\frac{2}{\beta}\mathrm{cost}^k_{\mathcal{C}}$. This gave us the following Corollary for g_2^+.

Corollary 6.2. *There exists a vector $c_2^+ \in \mathbb{R}^n_{\geq 0}$ with the following properties:*

(6.2a) $g_2^+ \leq \sum_{r \in V} c_2^+(r) \cdot \mathrm{cost}_{\mathcal{C}}(r)$.

(6.2b) $c_2^+(r) \leq \frac{2}{\beta} \cdot \frac{|\varphi_2^+(r)|}{d(r)} \leq \frac{2}{\beta}$*, for every* $r \in V$.

(6.2c) $|c_2^+|_1 \leq \frac{2}{\beta} \sum_{u \in U} \frac{|\varphi_2^+(u)|}{d(u)} \leq \frac{2k}{\beta}$.

We still need to bound g_4^+.

Lemma 6.3. *There exists a vector $c_4^+ \in \mathbb{R}^n_{\geq 0}$ with the following properties:*

(6.3a) $g_4^+ \leq \sum_{r \in V} c_4^+(r) \cdot \mathrm{cost}_{\mathcal{C}}(r)$.

(6.3b) $c_4^+(r) \leq \frac{1}{\lambda} + \frac{1}{\beta} + \frac{1}{\lambda\beta}$*, for every* $r \in V$.

(6.3c) $|c_4^+|_1 \leq (\frac{1}{\lambda} + \frac{1}{\beta} + \frac{1}{\lambda\beta})k$.

Proof. TOPROVE 26 □

Lemma 6.4. $g_1^+ + g_2^+ + g_4^+ \leq (\frac{3}{\beta} + \frac{1}{\lambda} + \frac{1}{\beta\lambda} + 1)\mathrm{cost}^k_{\mathcal{C}}$

Proof. TOPROVE 27 ☐

Lemma 6.4 gives us a way to bound the cost of deleted edges. Now consider the non-complete graph G_2 obtained from G by removing any +edge (u, v) (i.e., changing it into a ”neutral” edge, and the cost of this edge is 0) where u and v belong to different connected components of $\tilde{G}$. Note that for any clustering, the cost in G_2 is no more than the cost in G since we set some +edge costs to 0. Now we are going to bound the cost of F in G_2, where F is the clustering output by Algorithm 4. This can give us the relationship between F and $\mathcal{C}$ in G_2.

Since we will deal with multiple input graphs G_2, we include G_2 explicitly when necessary. Given a correlation clustering instance G_2, an integer k, and a clustering $\mathcal{C}$, we denote the disagreement vector in G_2 by $\mathrm{cost}_{\mathcal{C}}(G_2)$. Consequently, the disagreement for a node u in G_2 is specified by $\mathrm{cost}_{\mathcal{C}}(G_2, u)$. For a subset $S \subseteq V$, the top-k value on G_2 is represented by $\mathrm{cost}^k_{\mathcal{C}}(G_2, S) = \max_{T\subseteq S,|T|=k} \sum_{u\in T} \mathrm{cost}_{\mathcal{C}}(G_2, u)$. When $k \geq |S|$, then $\mathrm{cost}^k_{\mathcal{C}}(G_2, S) = \sum_{u\in S} \mathrm{cost}_{\mathcal{C}}(G_2, u)$

The following lemma provides the key insights that F is a good clustering in G_2.

Lemma 6.5 (Lemma 3.4 of [CALM+21]). *Suppose $5\beta + 2\lambda < 1$. Let $CC \in F$ be a connected component of $\tilde{G}$ such that $|CC| \geq 2$. Then for each vertex $u \in CC$ we have that*

$$d(u, CC) \geq (1 - 8\beta - \lambda)|CC|.$$

Lemma 6.5 tells us that each node $u \in V$ is connected to no less than $\frac{5}{8}$ fractions of the vertices that belong to the same cluster in F when assuming $8\beta + \lambda \leq 3/8$. Now we can bound the cost of F in G_2.

Lemma 6.6. *Let G_2 be a non-complete graph obtained from G by removing any + edge u, v (i.e., changing it into a “neutral” edge) where u and v belong to different connected components of $\tilde{G}$. Let $\mathcal{C}^*$ be the top-k optimal clustering for graph G_2. Assuming $8\beta + \lambda \leq 3/8$. Then, our algorithm outputs solution F such that*

$$\mathrm{cost}^k_F(G_2) \leq 7 \cdot \mathrm{cost}^k_{\mathcal{C}^*}(G_2).$$

Proof. TOPROVE 28 ☐

Now we can show our main lemma regarding the approximate ratio.

Proof. TOPROVE 29 ☐

References

[ACN08] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *Journal of the ACM*, 55(5):1–27, 2008. 3

[AHK+09] Rakesh Agrawal, Alan Halverson, Krishnaram Kenthapadi, Nina Mishra, and Panayiotis Tsaparas. Generating labels from clicks. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 172–181, 2009. 3

[AW22] Sepehr Assadi and Chen Wang. Sublinear time and space algorithms for correlation clustering via sparse-dense decompositions. In *Proceedings of the 13th Conference on Innovations in Theoretical Computer Science (ITCS)*, volume 215 of *LIPIcs*, pages 10:1–10:20, 2022. 4

[BBC04] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine learning*, 56(1):89–113, 2004. 3

[BFS12] Guy E Blelloch, Jeremy T 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*, pages 308–317, 2012. 4

[CALM$^+$21] Vincent Cohen-Addad, Silvio Lattanzi, Slobodan Mitrović, Ashkan Norouzi-Fard, Nikos Parotsidis, and Jakub Tarnawski. Correlation clustering in constant many parallel rounds. In *International Conference on Machine Learning*, pages 2069–2078. PMLR, 2021. 4, 6, 13, 20, 21, 22, 25

[CCAL$^+$24] Nairen Cao, Vincent Cohen-Addad, Euiwoong Lee, Shi Li, Alantha Newman, and Lukas Vogl. Understanding the cluster linear program for correlation clustering. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1605–1616, 2024. 3

[CCMU21] Mélanie Cambus, Davin Choo, Havu Miikonen, and Jara Uitto. Massively parallel correlation clustering in bounded arboricity graphs. In *35th International Symposium on Distributed Computing (DISC)*, volume 209 of *LIPIcs*, pages 15:1–15:18, 2021. 4

[CDK14] Flavio Chierichetti, Nilesh Dalvi, and Ravi Kumar. Correlation clustering in MapReduce. In *Proceedings of the 20th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 641–650, 2014. 4, 7

[CGS17] Moses Charikar, Neha Gupta, and Roy Schwartz. Local guarantees in graph cuts and clustering. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 136–147, 2017. 3

[CGW05] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005. 3

[CHS24] Nairen Cao, Shang-En Huang, and Hsin-Hao Su. Breaking 3-factor approximation for correlation clustering in polylogarithmic rounds. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4124–4154. SIAM, 2024. 4

[CKL$^+$24] Mélanie Cambus, Fabian Kuhn, Etna Lindy, Shreyas Pai, and Jara Uitto. A (3+)-approximate correlation clustering algorithm in dynamic streams. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2861–2880. SIAM, 2024. 4

[CKP08] Deepayan Chakrabarti, Ravi Kumar, and Kunal Punera. A graph-theoretic approach to webpage segmentation. In *Proceedings of the 17th International conference on World Wide Web (WWW)*, pages 377–386, 2008. 3

[CLLN23] Vincent Cohen-Addad, Euiwoong Lee, Shi Li, and Alantha Newman. Handling correlated rounding error via preclustering: A 1.73-approximation for correlation clustering. In *Proceedings of the 64rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2023. 3

[CLN22] Vincent Cohen-Addad, Euiwoong Lee, and Alantha Newman. Correlation clustering with Sherali-Adams. In *Proceedings of 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 651–661, 2022. 3

[CMSY15] 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 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 219–228, 2015. 3

[CS19] Deeparnab Chakrabarty and Chaitanya Swamy. Approximation algorithms for minimum norm and ordered optimization problems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 126–137, 2019. 1, 4, 5, 6, 8, 25

[CSX12] Yudong Chen, Sujay Sanghavi, and Huan Xu. Clustering sparse graphs. In *Advances in Neural Information Processing Systems (Neurips)*, pages 2204–2212, 2012. 3

[DG08] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. 4

[DMN23] Sami Davies, Benjamin Moseley, and Heather Newman. Fast combinatorial algorithms for min max correlation clustering. *arXiv preprint arXiv:2301.13079*, 2023. 6, 14, 15, 16, 19

[DMN24] Sami Davies, Benjamin Moseley, and Heather Newman. Simultaneously approximating all lp-norms in correlation clustering. In *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024*, 2024. 1, 3, 4, 5, 6

[FN19] Manuela Fischer and Andreas Noever. Tight analysis of parallel randomized greedy mis. *ACM Transactions on Algorithms (TALG)*, 16(1):1–13, 2019. 4

[HIA24] Holger SG Heidrich, Jannik Irmai, and Bjoern Andres. A 4-approximation algorithm for min max correlation clustering. In *AISTATS*, pages 1945–1953, 2024. 3

[KCMNT08] Dmitri V. Kalashnikov, Zhaoqi Chen, Sharad Mehrotra, and Rabia Nuray-Turan. Web people search via connection analysis. *IEEE Transactions on Knowledge and Data Engineering*, 20(11):1550–1565, 2008. 3

[KMZ19] Sanchit Kalhan, Konstantin Makarychev, and Timothy Zhou. Correlation clustering with local objectives. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 3, 5, 6, 7, 8, 9, 16, 18, 19

[Lub85] Michael Luby. A simple parallel algorithm for the maximal independent set problem. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 1–10, 1985. 7

[PM16] Gregory Puleo and Olgica Milenkovic. Correlation clustering and biclustering with locally bounded errors. In *International Conference on Machine Learning*, pages 869–877. PMLR, 2016. 3

[PPO+15] Xinghao Pan, Dimitris S. Papailiopoulos, Samet Oymak, Benjamin Recht, Kannan Ramchandran, and Michael I. Jordan. Parallel correlation clustering on big graphs. In *Advances in Neural Information Processing Systems (Neurips)*, pages 82–90, 2015. 4

[SDE+21] Jessica Shi, Laxman Dhulipala, David Eisenstat, Jakub Łkacki, and Vahab Mirrokni. Scalable community detection via parallel correlation clustering. *arXiv preprint arXiv:2108.01731*, 2021. 3

[VGW18] Nate Veldt, David F. Gleich, and Anthony Wirth. A correlation clustering framework for community detection. In *Proceedings of the 2018 ACM World Wide Web Conference (WWW)*, pages 439–448, 2018. 3

A Reduction from All Monotone Symmetric Norms to Top-k Norms

Definition A.1 (Ordered Norms). *For any vector $x \in \mathbb{R}^n_{\geq 0}$, let $x^{\downarrow}$ denote the vector x with its coordinates sorted in non-increasing order. Given weight vector $w \in \mathbb{R}^n_{\geq 0}$ with $w_1 \geq w_2 \geq \cdots \geq w_n$, the w-ordered norm of x is defined as $\text{order}(w; x) = \sum_{i=1}^n w_i x_i^{\downarrow}$.*

Lemma A.2 (Lemma 5.2 of [CS19]). *For any monotone and symmetric norm $f : \mathbb{R}^n \to \mathbb{R}_+$, define the set $\mathbb{B}_+(f) := \{x \in \mathbb{R}^n_+ : f(x) \leq 1\}$, and $W = \{w \in \mathbb{R}^n_+ : w_1 \geq w_2 \geq \cdots \geq w_n, w$ is a subgradient of*
f at some $x \in \mathbb{B}_+(f)\}$. Then we have $f(x) = \max_{w \in W} \text{order}(w; x)$ for every $x \in \mathbb{R}^n_{\geq 0}$.

Lemma 2.1. *For any integer $k \in [n]$, if an algorithm returns a single clustering $\mathcal{C}_{ALG}$ that is simultaneously a ρ-approximation for all top-k norm objectives, then $\mathcal{C}_{ALG}$ is a ρ-approximation for any monotone and symmetric norm $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_+$.*

Proof. TOPROVE 30 □

B Proof for Theorem 4.2

We repeat Theorem 4.2 for convenience.

Theorem 4.2. *[Lemma 3.11 of [CALM+21]] Let $\delta, \epsilon, \beta \in (0, 1)$ be constants. There exists an MPC algorithm that, given a graph $G = (V, E)$, outputs a "Yes/No" answer for every $uv \in E$, such that the following happens with probability at least $1 - \frac{1}{n^6}$.*

- *For every $uv \in E$ with $|N(u)\Delta N(v)| \leq \beta M_{uv}$, algorithm outputs "Yes" for uv.*
- *For every $uv \in E$ with $|N(u)\Delta N(v)| \geq (1+\epsilon)\beta M_{uv}$, algorithm outputs "No" for uv.*

The algorithm runs in $O(1)$ rounds, with a total work of $\tilde{O}(m/\epsilon^2)$, $O(n^\delta)$ memory per machine, and $\tilde{O}(m/\epsilon^2)$ total memory.

Proof. TOPROVE 31 □

C Proof for Lemma 4.5

We repeat Theorem 4.5 for convenience.

Lemma 4.5. *With probability at least $1 - 1/n^6$, Algorithm 2 outputs $\tilde{x}_{uv}$ such that*

(4.5a) For any $uv \in E$, we have $\tilde{x}_{uv} \leq (1+\epsilon)x_{uv}$. For any $uv \in \binom{V}{2} \setminus E$, we have $1 - \tilde{x}_{uv} \leq (1+\epsilon)(1 - x_{uv})$.

(4.5b) For any $u, v, w \in V$, we have $\tilde{x}_{uv} + \tilde{x}_{uw} + 3\epsilon \geq \tilde{x}_{vw}$.

Proof. TOPROVE 32 □