TheoretiCS

# A $(3+\varepsilon)$-Approximate Correlation Clustering Algorithm in Dynamic Streams

**Received** Feb 21, 2024
**Revised** Nov 23, 2024
**Accepted** Jan 12, 2025
**Published** Feb 28, 2025

**Key words and phrases**
semi-streaming, correlation clustering, dynamic streams, single pass

**Mélanie Cambus** [a]
**Fabian Kuhn** [b]
**Etna Lindy** [a]
**Shreyas Pai** [c]
**Jara Uitto** [a]

**a** Aalto University, Finland

**b** University of Freiburg, Germany

**c** Indian Institute of Technology Madras, India

**ABSTRACT.** Grouping together similar elements in datasets is a common task in data mining and machine learning. In this paper, we study streaming algorithms for correlation clustering, where each pair of elements is labeled either similar or dissimilar. The task is to partition the elements and the objective is to minimize disagreements, that is, the number of dissimilar elements grouped together and similar elements that get separated.

Our main contribution is a semi-streaming algorithm that achieves a $(3+\varepsilon)$-approximation to the minimum number of disagreements using a single pass over the stream. In addition, the algorithm also works for dynamic streams. Our approach builds on the analysis of the PIVOT algorithm by Ailon, Charikar, and Newman [JACM'08] that obtains a 3-approximation in the centralized setting. Our design allows us to sparsify the input graph by ignoring a large portion of the nodes and edges without a large extra cost as compared to the analysis of PIVOT. This sparsification makes our technique applicable in models such as semi-streaming, where sparse graphs can typically be handled much more efficiently.

Our work improves on the approximation ratio of the recent single-pass 5-approximation algorithm and on the number of passes of the recent $O(1/\varepsilon)$-pass $(3+\varepsilon)$-approximation algorithm [Behnezhad, Charikar, Ma, Tan FOCS'22, SODA'23]. Our algorithm is also more robust and can

A preliminary version of this article appeared at SODA 24 [**14**].

**Cite as** Mélanie Cambus, Fabian Kuhn, Etna Lindy, Shreyas Pai, Jara Uitto. A $(3+\varepsilon)$-Approximate Correlation Clustering Algorithm in Dynamic Streams. TheoretiCS, Volume 4 (2025), Article 6, 1-19.

https://theoretics.episciences.org
**DOI** 10.46298/theoretics.25.6

be applied in dynamic streams. Furthermore, it is the first single pass $(3+\varepsilon)$-approximation algorithm that uses polynomial post-processing time.

## 1. Introduction

In this paper, we consider the *correlation clustering* problem introduced by [10], where the goal is to group together similar elements and separate dissimilar elements. We model the similarity as a complete signed graph $G = (V, E^+ \cup E^-)$, where a *positive* edge $\{u, v\} = e \in E^+$ indicates that $u$ and $v$ are similar. In case $e \in E^-$, the edge is *negative* and the nodes are dissimilar. The goal is to minimize the *disagreements*, where a disagreement is induced by grouping together dissimilar nodes or separating similar ones. As pointed out by [19], it is typically the case that the set of negative edges is much larger than the set of positive edges. Hence, in this paper, we identify the input graph with the set of positive edges, i.e., $G = (V, E^+)$ and the negative edges are defined implicitly[1]. Correlation clustering is a natural abstraction for central problems in data mining and machine learning such as community and duplicate detection [5, 18], link prediction [30], and image segmentation [27]. A key feature of correlation clustering, as opposed to, for example, the standard $k$-means clustering, is that the number of clusters is not predetermined.

As the volume of data sets is growing fast, there is an increasing demand for *sublinear* solutions to clustering problems. Our main contribution is a novel sparsification technique, where we turn an input graph of $n$ nodes and $m$ edges into a sparse representation of $\widetilde{O}(n)$ bits[2]. We show how to find a $(3+\varepsilon)$-approximate clustering of the *original* graph by only processing the sparsified graph. This approach is appealing for many models of computation tailored for processing massive data sets such as semi-streaming, where the working space is much smaller than the size of the input graph. We measure the space as *words* of $O(\log n)$ bits, which is just enough to store an identifier of an edge. We now state our main result and then, introduce the semi-streaming model and related work.

**THEOREM (Main Theorem, informal version).** *There is a single-pass semi-streaming algorithm that obtains a $(3+\varepsilon)$-approximation to correlation clustering. The algorithm works even for dynamic streams. The approximation guarantee holds in expectation and with high probability[3].*

**State-of-the-Art in Semi-Streaming.** In graph streaming, the input graph is given to the algorithm as an *edge stream* [25, 26, 28]. In the semi-streaming setting, the algorithm has $\widetilde{O}(n)$

1 We can instead identify the input graph with $G = (V, E^+)$ i.e. the set of negative edges. This does not make a difference for dynamic semi-streaming as we can modify the stream to first add edges between all vertex pairs, and then the stream of negative edges can be interpreted as edge deletions.

2 The $\widetilde{O}(f(n))$-notation hides polylogarithmic in $n$ terms.

3 An event holds with high probability, w.h.p., if it holds with probability at least $1 - n^{-c}$ for a desirably large constant $c \geq 1$.

working space to store its state. The goal is to make as few *passes* over the edge-stream as possible, ideally just one. It is well known that there is a strong separation between one and two passes for problems like deterministic coloring [6] and minimum cuts [7]. In the case of many problems, such as matching approximation or correlation clustering, simply storing the output might demand $\Omega(n)$ words.

For correlation clustering, it has already been observed in [1, 13] that by allowing exponential-time computation, one can first run a streaming algorithm that computes an $\widetilde{O}(n)$-sized sketch of the graph that approximately stores the values of all cuts [3] and to then brute-force a solution by iterating over all possible clusterings. In this way, one obtains a $(1+\varepsilon)$-approximation algorithm that uses $\widetilde{O}(n/\varepsilon^2)$ space and a single pass even for dynamic streams. Note that since correlation clustering is APX-hard [17], unless P = NP, exponential-time computation is necessary for obtaining a $(1+\varepsilon)$-approximation. The focus has therefore been on designing polynomial-time algorithms that achieve a constant approximation ratio.

Constant approximation ratios have been reached by using the sparse-dense decomposition [20, 9] in single-pass semi-streaming. On the downside, the approximation ratios, while being constant, are very high. In the case of [20], they obtain an approximation ratio over 700, the ratio of [9] is over 6400. An $O(1/\varepsilon)$-pass semi-streaming algorithm was given that obtains a $(3+\varepsilon)$-approximation to correlation clustering in [12]. A 5-approximation was given using just a single pass [13]. Chakrabarty and Makarychev [16] improve the single-pass 5-approximation algorithm of [13] to obtain a $(3+\varepsilon)$-approximation. For insertion-only streams, the algorithm of [16] only requires $O(n)$ words of space, whereas our algorithm in this case requires $O(n\log^2 n)$ words of space. However, a downside of the recent works by [12, 13, 16] is that they do not work in dynamic streams. Our algorithm and its analysis are more robust in the sense that they can be adapted to dynamic streams by using standard techniques.

**REMARK 1.1.** Subsequent to our work, the approximation ratio has been improved to 1.876 by [23] using sublinear space in the streaming setting.

## 1.1 Related Works on Correlation Clustering

In the centralized setting, finding an optimal clustering that minimizes disagreements is known to be NP-hard [10], which motivates the study of approximation algorithms. We note that there is another variant of the correlation clustering problem where we are interested in *maximizing agreements*. An agreement corresponds to clustering together positive edges and separating negative edges. This variant is also NP-hard since the optimum solutions are the same for the maximization and the minimization problems. However for approximate solutions, the two variants are very different. For maximizing agreements, a trivial algorithm consisting in forming one single cluster or only single node clusters yields a $1/2$-approximation. Furthermore,

0.7664-approximation and 0.7666-approximation algorithms are known, even for weighted graphs [29, 17].

In this paper, we focus on the minimizing disagreements problem. The first work to breach the integrality gap of 2 for the standard LP relaxation of the problem was due to [22, 21], it gives an approximation ratio of $(1.73 + \varepsilon)$ through rounding a solution to the Sherali-Adams relaxation. The current state-of-the-art approximation ratio is 1.437 due to [15], which is obtained by rounding the solution to the cluster LP. The cluster LP is exponentially-sized but it can be approximately solved in polynomial time and it has the advantage that we can do rounding without having to deal with correlated rounding errors.

The simple and well-known PIVOT algorithm, yields a 3-approximation [4] and is not based on solving an LP. The PIVOT algorithm works as follows.

— In each sequential step, pick a node $u$ uniformly at random.
— Create a cluster $C$ that contains $u$ and all of its neighbors in the current graph.
— Remove $C$ from the graph and recurse on the remaining graph.

An equivalent formulation is through a *randomized greedy Maximal Independent Set (MIS)*, where we pick a random permutation of the nodes and iterate over the nodes according to the permutation. In each step, the current node $v$ is selected to the MIS and its neighbors removed from the graph, unless $v$ was removed in an earlier step. Through the randomized greedy MIS, one can obtain an $O(\log\log\Delta)$-pass algorithm for a 3-approximation in semi-streaming [1].

Due to this connection to MIS, implementing the PIVOT algorithm in semi-streaming is also provably hard. There is an $\widetilde{\Omega}(n^2)$ space lower bound for computing an MIS in a single-pass of a stream of edges [24] and any semi-streaming algorithm using $O(n \cdot \operatorname{poly}\log(n))$ space for finding an MIS with constant probability of success requires $\Omega(\log\log n)$ passes [8]. Nevertheless, variants of the PIVOT algorithm have been successfully shown to achieve good approximations. Our algorithm, and the works of [12, 13, 16] discussed earlier are all variants of the PIVOT algorithm.

**Prior Work and Dynamic Streams** We now elaborate on the details of [13, 16] and explain why it does not extend to dynamic streams. To compute the 5-approximation, [13] first picks a random permutation of the nodes and for each node maintains a pointer to the neighbor with the smallest rank in the permutation throughout the stream. A partial clustering is obtained based on these pointers and then unclustered nodes are put into singleton clusters. The algorithm has a linear space requirement for insertion-only streams. The authors of [16] improve on this work by implementing a similar scheme but keep track of the $k$ smallest rank neighbors for each node. They show that this extension gives a $(3 + O(1/k))$-approximation. Their approach requires $O(kn)$ words of space in insertion-only streams. However, finding the smallest rank neighbor for each node seems fundamentally challenging since computing a minimum in dynamic streams

is provably hard. Our algorithm on the other hand can be implemented in dynamic streams with only a poly $\log n$ space overhead, while giving the same $(3+\varepsilon)$-approximation guarantee.

## 1.2 A High Level Technical Overview of Our Contributions

Our main contribution is a graph *sparsification* technique inspired by the approaches that simulate the greedy MIS to approximate correlation clustering. In the previous aforementioned works based on directly simulating the greedy MIS, the progress guarantee is given by a double exponential drop in the maximum degree or the number of nodes in the graph leading to $O(\log\log\Delta)$ and $O(\log\log n)$ pass algorithms. Moreover, the handle used to obtain this degree drop is the following: Consider the random permutation over the nodes. After processing the first $t$ nodes, we can guarantee that the maximum degree is at most $O(n\log n)/t$ w.h.p. (see for example [1]). Furthermore, after the maximum degree is $d$, a prefix of length roughly $n/\sqrt{d}$ of the random permutation contains $O(n)$ edges. Then we can iterate over the permutation and get the guarantee that the maximum degree of the remaining graph is $O(\sqrt{d}\log n)$.

For a single pass semi-streaming algorithm, this approach seems fundamentally insufficient, as it relies on the progress related to the maximum degree of the graph. In our approach, we give a modified process that effectively gives a 3-approximation in expectation (as in the greedy process), but only for *almost* all nodes. For this exposition, suppose that we have a $d$ regular graph for a sufficiently large $d$. After we process the "prefix" containing the first $\Theta(n/d)$ nodes in the permutation, we expect that the degree of each node $u$ has dropped by a significant factor or a neighbor of $u$ has joined the MIS.

The key idea is that if a node $u$ is not part of this prefix, it is unlikely to join the MIS after this prefix is processed. We leverage this idea as follows. Prior to the simulation of the randomized greedy MIS, we "set aside" all nodes whose rank in the permutation is considerably larger than $n/d$. The graph on the nodes with rank at most $n/d$ corresponds roughly to a set of nodes sampled with probability $1/d$, which we show to contain $\widetilde{O}(n)$ edges.

We process the prefix graph (i.e. the graph induced by sampled nodes) by running a greedy MIS algorithm on it. This corresponds to running the PIVOT algorithm on the prefix graph that does not contain any nodes that are set aside. From prior work [4], we almost immediately get that we do not lose more than a factor of 3 from the optimum on the nodes clustered by the MIS on the sampled graph (Lemma 3.6).

For the nodes that are set aside, we need more work. By carefully choosing the prefix length, we show that the degree of each node in the input graph drops by at least a factor of $(1-\varepsilon)$, with high probability, due to the greedy MIS (Lemma 3.7). We can charge each edge $e$ between a PIVOT node and a node set aside to the greedy MIS analysis. We then give a counting argument that shows that only an $\varepsilon$ factor of the edges are between the nodes set aside (Lemma 3.9). Hence, we can charge those to the PIVOT analysis and pay only an additive $\varepsilon$ factor in the approximation.

By setting the sampling probability appropriately, this line of attack works also for the non-regular case. This idea is the basic building block for our results. We note that in this sampling step, we add a $\log n$ and an $\varepsilon$ term into the sampling probability in order to obtain a degree drop large enough for our approximation analysis and to make sure all guarantees hold with high probability.

### 1.3 The Semi-Streaming Model.

In the semi-streaming model, the input graph is not stored centrally, but an algorithm has access to the edges one by one in an input stream. A *single-pass* semi-streaming algorithm has $\widetilde{O}(n)$ working space that it can use to store its state and is allowed to go through the stream only once.

In the dynamic setting, the input stream consists of arbitrary edge insertions and deletions. Formally, the input stream is a sequence $S = \langle s_1, s_2, ... \rangle$ where $s_i = (e_i, \delta_i)$ where $e_i$ encodes an arbitrary undirected edge and $\delta_i \in \{-1, 1\}$. The multiplicity of an edge $e$ is defined as $f_e = \sum_{i:e_i=e} \delta_i$. Since the input graph is simple, we assume that $f_e \in \{0, 1\}$ throughout the stream for all $e$. At the end of the stream, we have $f_e = 1$ if $e$ belongs to the input graph and 0 otherwise. In the insertion-only setting, the input stream consists only of edge insertions, i.e. $\delta_i = 1$ for all $i$.

For the sake of clarity, we describe here *how* the stream of edges is chosen. We assume that the graph is fixed before the algorithm executes, but the edges updates arrive in an adversarial order. The edge updates are revealed by the adversary, depending on the choices made by the algorithm so far. Although it is not explicitly stated, [13, 16] assume this setting for insertion-only streams. Our algorithm also works in this setting even for dynamic streams.

#### Organization of the Paper

The paper is organized as follows. In section 2, we introduce the Truncated-Pivot algorithm (Algorithm 1) for correlation clustering and show how it can be implemented in a single-pass in the dynamic and insertion-only semi-streaming models. In section 3, we show that the Truncated-Pivot algorithm returns a $(3 + \varepsilon)$-approximation of an optimum clustering.

## 2. The Truncated-Pivot Correlation Clustering Algorithm

In this section, we give the Truncated-Pivot algorithm for correlation clustering, which forms the basis for the semi-streaming implementation. The high-level idea of our algorithm is to compute a randomized greedy MIS with a small twist. Informally, we exclude nodes whose degree is likely to drop significantly before they are processed in the greedy MIS algorithm, where the MIS nodes will correspond to the PIVOT nodes, or simply *pivots*. This then allows us to effectively *ignore* a large fraction of the nodes that will never be chosen as pivots.

**Input:** Graph $G = (V, E^+)$, each node $v \in V$ knows its degree $\deg(v)$ in $G$

1: Fix a random permutation $\pi$ over the nodes.
2: Initially, all nodes are unclustered and *interesting*.
3: A node $u$ marks itself *uninteresting* if $\pi_u \geq \tau_u$ where $\tau_u = \frac{c}{\varepsilon} \cdot \frac{n \log n}{\deg(u)}$
4: Let $G_{\text{store}}$ be the graph induced by the interesting nodes.
5: Let $\mathcal{I}$ be the output of running greedy MIS on $G_{\text{store}}$ with ordering $\pi$.

6: Nodes in $\mathcal{I}$ become cluster centers (pivots).
7: Each node $u \in V \setminus \mathcal{I}$ joins the cluster of the smallest rank pivot neighbor $v$, if $\pi_v < \tau_u$.
8: Each unclustered node forms a singleton cluster.

**Algorithm 1.** Truncated-Pivot

In Section 3 we will prove the following theorem that gives a guarantee on the cost of the clustering returned by Algorithm 1.

**THEOREM 2.1 (Main Theorem, formal version).** *For any $\varepsilon \in (0, 1/4)$, the Truncated-Pivot algorithm (Algorithm 1) is a $(3+\varepsilon)$-approximation algorithm to the Correlation Clustering problem. The approximation guarantee is in expectation.*

## 2.1 Implementation in Dynamic Streams

Here we describe and analyze Algorithm 2, which implements Truncated-Pivot in the dynamic semi-streaming model. We begin with the observation that in order to simulate Algorithm 1, we only need to store the edges incident to interesting nodes. This is because we run a greedy MIS on the graph induced by the interesting nodes, and in Line 7, we only cluster vertices that are neighbors of pivot (i.e. interesting) nodes.

According to Line 3 of Algorithm 1, a node $u$ marks itself uninteresting if $\pi_u \geq \tau_u$ where $\tau_u = cn \log n/\varepsilon \deg(u)$. This is equivalent to saying that $u$ marks itself uninteresting if $\deg(u) \geq \sigma_u$ where $\sigma_u = cn \log n/\varepsilon\pi_u$. Therefore, if the stream was insertion-only, we could only store the edges of $u$ as long as $\deg(u) < \sigma_u$.

The main challenge with dynamic streams is that we need to keep track of the incident edges of a node even if $\deg(u) \geq \sigma_u$, because its degree could go down later in the stream, and it could become interesting again. To overcome this, we will maintain a *k-sparse recovery* data

structure for the incident edges of each node, that allow us to recover the ($< \sigma_u$) incident edges of each interesting node $u$ at the end of the stream deterministically. This strategy is described more formally in Algorithm 2.

The following lemma describes a deterministic $k$-sparse recovery data structure, which follows from Lemma 9 in [11] (by substituting $n = k$, $u = n$, and $r = 1$ for our use case).

**LEMMA 2.2 (Lemma 9, [11]).** *There exists a deterministic data structure, $k$-sparse recovery with parameter $k$, that that maintains a sketch of stream $I$ (involving insertions and deletions of elements from $[n]$) and can recover all of $I$'s elements if $I$ contains at most $k$ distinct elements. It uses $O(k \log n)$ bits of space and can be updated in $O(\log^2 k)$ amortized operations.*

Note that Lemma 2.2 does not give any guarantees if the stream contains more than $k$ distinct elements. In this case, the output might be something completely meaningless. But in our use case, this only happens for uninteresting nodes, and we don't want to recover their incident edges anyway. Therefore, we are able to deterministically recover all the edges incident on interesting nodes at the end of the stream.

**Input:** Graph $G = (V, E)$ as a dynamic stream of edge insertions and deletions

1: Fix a random permutation $\pi$ over the nodes.

2: Initially, all nodes $u$ are unclustered and *interesting*, $\deg(u) = 0$, and $\sigma_u = \frac{c}{\varepsilon} \cdot \frac{n \log n}{\pi_u}$.

3: For each node $u$, we initialize a $\sigma_u$-sparse recovery data structure for the adjacency vector of $u$ (the row of the adjacency matrix of $G$ that corresponds to $u$).

4: Upon receiving the $i^{th}$ element of the stream, $s_i = (e_i, \delta_i)$ where $e_i = \{u, v\}$, we update $\deg(u)$, $\deg(v)$, and the sparse recovery data structures associated with $u$ and $v$.

**At the end of the stream:**

5: A node $u$ marks itself *uninteresting* if $\deg(u) \geq \sigma_u$.

6: We retrieve all incident edges of interesting nodes using the $\sigma_u$-sparse recovery structures for all $u$.

7: Simulate Lines 4 to 8 of Algorithm 1.

**Algorithm 2.** Dynamic Semi-Streaming Truncated-Pivot

We now prove a bound on the space requirement of Algorithm 2. Note that for insertion-only streams we can get the same space guarantee by simply storing the $(< \sigma_u)$ incident edges of all interesting nodes $u$.

**LEMMA 2.3.** *Algorithm 2 requires $O(n\log^2(n)/\varepsilon)$ words of space.*

**PROOF.** For node $u$, $\sigma_u$-sparse recovery requires $O(\sigma_u \cdot \log n)$ bits of space, where $\sigma_u = cn\log n/\varepsilon\pi_u$. Since, each node requires one single $\sigma_u$-sparse recovery structure, the total amount of memory required to store and maintain all $\sigma_u$-sparse recovery structures throughout the algorithm is:

$$\sum_{u\in V}\sigma_u\cdot\log n=\sum_{i=1}^{n}\frac{cn\log^2 n}{\varepsilon\cdot i}=\frac{cn}{\varepsilon}\log^2 n\cdot\sum_{i=1}^{n}\frac{1}{i}=O(n\log^3 n/\varepsilon)$$

because the $n^{th}$ harmonic number is $H_n = O(\log n)$. For each node, storing the current degree and the node identifier only requires $O(\log n)$ bits of memory, which makes the memory necessary for the algorithm $O(n\log^3 n/\varepsilon)$ bits. Since each word contains $O(\log n)$ bits, the lemma follows. ■

**THEOREM 2.4.** *Algorithm 2 computes a $(3+\varepsilon)$-approximation in expectation of an optimum clustering in a single pass of the dynamic semi-streaming model, and it requires $O(n\log^2(n)/\varepsilon)$ words of space.*

**PROOF.** By Lemma 2.2, in Algorithm 2, all edges with an interesting endpoint can be recovered. Hence, Algorithm 2 works with the same set of edges as Algorithm 1 when computing the clustering, thus implying that both algorithms return the same clustering.

Since Theorem 2.1 implies that Algorithm 1 outputs a clustering with expected cost that is a $(3+\varepsilon)$-approximation, then Algorithm 2 does as well. Additionally, Lemma 2.3 states that Algorithm 2 requires $O(n\log^2(n)/\varepsilon)$ words of space throughout the stream. ■

**REMARK 2.5.** We can run Algorithm 1 independently $O(\log n)$ times and return the best solution to get a w.h.p. approximation guarantee using standard probability amplification arguments. This adds an additional $\log n$ factor to the space requirement. The lowest cost clustering can be found in the same pass by (approximately) evaluating the cost of each clustering on a cut-sparsifier [2] (see Appendix A of [13] for more details).

## 3. Approximation Analysis of Truncated-Pivot

The goal of this section is to prove the approximation guarantee of the Truncated-Pivot algorithm. For a more comfortable analysis, we prove the approximation guarantee for a sequential version that produces the same output as Algorithm 1 for each permutation. Following is the main result of this section.

**THEOREM 3.1.** *For any $\varepsilon \in (0, 1/4)$, Sequential Truncated-Pivot (Algorithm 3) is a $(3+\varepsilon)$-approximation algorithm to the correlation clustering problem. The approximation guarantee is in expectation.*

**A Sequential Process.** Consider the following (sequential) algorithm and refer to Algorithm 3 for a pseudocode representation. Initially, each node is considered *active*. For each node $u$, we store the degree $\deg(u)$ of $u$ in the input graph. We pick a random permutation $\pi$ on the nodes and in each iteration, we pick a node following the permutation. If this node is still active, it is chosen as a pivot and we create a *pivot cluster* consisting of the pivot node and its active neighbors (Line 7 of Algorithm 3). The clustered nodes then become inactive and will not be chosen as pivots later.

**Input:** Graph $G = (V, E^+)$, each node is active in the beginning. Let $\deg(u) = |N(u)|$ be the *initial* degree of node $u$

1: Pick a random permutation $\pi$ over the nodes.
2: **for** iteration $i = 1, 2, \ldots$ ▷ Iterate over $\pi$
3: **do**
4: Let $\ell := \frac{c}{\varepsilon} \cdot \frac{n \log n}{i}$ ▷ $c$ is a well-chosen constant.
5: Let $u \in V$ be the $i^{th}$ node in $\pi$.
6: Each active node $v$ with $\deg(v) \geq \ell$ becomes inactive and creates a singleton cluster
7: If $u$ is active, create a pivot cluster $C$ consisting of $u$ and its active neighbors.
8: Each node in $C$ becomes inactive.

**Algorithm 3.** Sequential Truncated-Pivot

Additionally, in iteration $i$, we check whether each active node $v$ has a degree significantly larger than $(n \log n)/i$. If so, we expect that the previous pivot choices have removed a large fraction of the neighbors of $v$ from the graph. In this case, $v$ becomes a singleton cluster (Line 6 in Algorithm 3) and we charge the remaining edges of $v$ to the edges incident on neighbors that joined some pivot clusters in previous iterations. Notice that the edges of $v$ that got removed before iteration $i$ can be due to a neighbor joining a pivot cluster or due to creating a singleton cluster. As a technical challenge, we must show that most of the neighbors joined pivot clusters. Before the approximation analysis, we show in Lemma 3.2 that Algorithm 1 and Algorithm 3 produce the same clustering if they sample the same random permutation.

### 3.1 Equivalence with Truncated-Pivot

**LEMMA 3.2.** *Fix a (random) permutation $\pi$ over the nodes of $G = (V, E)$. Running the Sequential Truncated-Pivot (Algorithm 3) with $\pi$ outputs the same clustering as running the Truncated-Pivot (Algorithm 1) with $\pi$.*

**PROOF.** Our goal is to show that both algorithms output the same clustering. First, we show that in both cases, the singleton clusters are the same. Then, we show that in both cases the greedy MIS runs on the same subgraph, hence outputting the same pivot clusters.

Consider a node $u$ that is active in the beginning of iteration $i$ ($i \leq \pi_u$), and becomes a singleton cluster due to Line 6 of Algorithm 3. By definition, $i$ is the smallest integer such that $\deg(u) \geq \frac{c}{\varepsilon} \cdot \frac{n}{i}$ and therefore, $i = \lceil \tau_u \rceil$. Since $i \leq \pi_u$, we have $\deg(u) \geq \frac{c}{\varepsilon} \cdot \frac{n}{\pi_u}$, which corresponds to $u$ being uninteresting in Algorithm 1. Since $u$ is in a singleton cluster, it did not join any pivot cluster, implying that no neighbor of $u$ was picked as a pivot before $u$ became a singleton cluster (i.e. $\forall v \in N(u), \pi_v > i$ or $v$ was clustered before iteration $\pi_v$). Hence, no neighbor $v$ of $u$ s.t. $\pi_v < \lceil \tau_u \rceil$ becomes a pivot. Since $\pi_v$ is an integer, this is equivalent to saying no neighbor $v$ of $u$ s.t. $\pi_v < \tau_u$ becomes a pivot, so by Line 7 of Algorithm 1, $u$ creates a singleton cluster in Algorithm 1 as well.

Now consider a node $u$ that creates a singleton cluster in Algorithm 1. Node $u$ must have been labeled uninteresting (implying $\pi_u \geq \tau_u$), and $u$ can neither be a pivot nor have a neighboring pivot $v$ satisfying $\pi_v < \tau_u$. By definition of $\tau_u$, iteration $\lceil \tau_u \rceil$ is the smallest iteration such that $\deg(u) \geq \frac{c}{\varepsilon} \cdot \frac{n}{\lceil \tau_u \rceil}$. This implies that $u$ must be active at the beginning of iteration $\lceil \tau_u \rceil$ in Algorithm 3, and forms a singleton cluster in that iteration.

Since the nodes forming singleton clusters in both algorithms are the same, the subgraph induced by nodes not forming singleton clusters $G\left[V \setminus V^{\text{sin}}\right]$ is the same in both cases. Both algorithms find a greedy MIS on $G\left[V \setminus V^{\text{sin}}\right]$, which implies that the pivot nodes will be the same in both cases. Finally, we observe that in both algorithms, a non-pivot node $u$ joins the cluster of the first neighbor $v$ s.t. $\pi_v < \tau_u$. Hence, the pivot clusters are the same for both Algorithm 3 and Algorithm 1. ■

### 3.2 Analyzing the Pivot Clusters

As the first step of our approximation analysis, we bound the number of disagreements caused by the pivot nodes and their respective clusters. The analysis is an adaptation of the approach by [4], where we only focus on a subset of the nodes.

Recall the PIVOT algorithm [4] that computes a greedy MIS. Initially, each node is considered *active*. The PIVOT algorithm picks a random permutation of the nodes and iteratively considers each node in the permutation. For each active node $u$ (iterating over the permutation), PIVOT forms a cluster with the active neighbors of $u$. The cluster is then deleted from the graph by marking the nodes in the new cluster *inactive*. This is repeated until the graph is empty, i.e.,

all the nodes are clustered. The PIVOT algorithm gives a solution with the expected cost being a 3-approximation of the optimum solution.

The 3-approximation given by the PIVOT algorithm is due to the nature of the mistakes that can be made through the clustering process. Consider $u, v, w \in V$: if $e_1 := \{u, v\}$ and $e_2 := \{v, w\}$ are in $E^+$ but $e_3 := \{w, u\} \in E^-$, then clustering those nodes has to produce at least one mistake. The triplet $(e_1, e_2, e_3)$ is called a *bad triangle*. Because a bad triangle induces at least one mistake in any clustering, even an optimum one, the number of disjoint bad triangles gives a lower bound on the disagreement produced by an optimum clustering. In the case of the pivot algorithm, since only direct neighbors of a pivot are added to a cluster, then the following mistakes can happen. Either two neighbors are included in the same cluster being dissimilar, which includes a negative edge in the cluster (the pivot was the endpoint of two positive edges in a bad triangle), or the pivot was an endpoint of the negative edge in a bad triangle which implies that only one positive edge of this bad triangle is included in the cluster and the second positive edge is cut. The authors of [4] show that the expected number of mistakes produced by the PIVOT algorithm is the sum of the probability that we make a mistake on every single bad triangle (not necessarily disjoint) in the graph. The 3-approximation is obtained by comparing this expected cost to the cost of a *packing LP* which is a lower bound on the cost of an optimum clustering. Our analysis for the mistakes caused by the pivot clusters (Lemmas 3.5 and 3.6) is almost the same as in the previous work [4]. Our analysis of the singleton clusters requires us to have an explicit handle on the positive disagreements between the pivot clusters and the singleton clusters, provided by the analysis of the pivot clusters.

**The Cost of Pivot Clusters in Sequential Truncated-Pivot.** Let us phrase the expected cost of pivot clusters of Sequential Truncated-Pivot (Line 7 of Algorithm 3). Recall that a bad triangle refers to a 3-cycle with two positive and one negative edge.

**DEFINITION 3.3.** Consider the set of all bad triangles $T$, and let $t \in T$ be a bad triangle on nodes $u, v$ and $w$. Define $A_t$ to be the event that, in some iteration, all three nodes are active and one of $\{u, v, w\}$ is chosen as a pivot (Line 7 in Algorithm 3). Let $p_t = \Pr[A_t]$.

**DEFINITION 3.4.** Let $C^{\text{pivot}}$ be the cost, i.e., the number of disagreements induced by the pivot clusters (Line 7 in Algorithm 3). For a pivot cluster $C$ created in iteration $i$, the disagreements include (1) the negative edges inside $C$ and (2) the positive edges from nodes in $C$ to nodes that are active in iteration $i$ and not contained in $C$. The edges that correspond to positive disagreements caused by the pivot clusters are said to be *cut* by the pivot clusters.

**LEMMA 3.5.** *Let $T$ be the set of bad triangles in the input graph. Then, $\mathbb{E}[C^{pivot}] \leq \sum_{t \in T} p_t$.*

**PROOF.** Consider a bad triangle $t \in T$ and suppose that in some iteration $i$ all nodes in $t$ are active and one of them is chosen as a pivot node (Line 7 in Algorithm 3), i.e. the event $A_t$ happens

at iteration $i$. Then, our algorithm creates one disagreement on this triangle on one of its edges $e \in t$. We charge this disagreement on edge $e$.

We observe that each triangle $t$ can be charged at most once: An edge $e \in t$ is charged only if it is not incident on the pivot node and hence, cannot be charged twice in the same iteration. Hence, at most one edge of $t$ can be charged in one iteration. Furthermore, if $e \in t$ gets charged in iteration $i$, its endpoints will not be both active in any later iteration $j > i$. This implies that $t$ cannot be charged again in another iteration.

Also, creating clusters with neighbors can only create disagreements on bad triangles. Since dropping certain nodes of the graph cannot create bad triangles, the number of disagreements created on a subgraph by this process cannot be higher than the number of disagreements created on the whole graph. Therefore, $\mathbb{E}[C^{\text{pivot}}] \leq \sum_{t \in T} p_t$. ■

**Bounding OPT.** In order to give an approximation guarantee to the clustered nodes, we first define the following fractional LP. It was argued by [4] that the cost of the optimal solution $\text{LP}_{OPT}$ to this LP is a lower bound for the cost OPT of the optimal solution for correlation clustering. Following are the primal and dual forms of this LP, respectively:

$$\min \sum_{e \in E^- \cup E^+} x_e, \quad \text{s.t.} \sum_{e \in t} x_e \geq 1, \forall t \in T \qquad \max \sum_{t \in T} y_t, \quad \text{s.t.} \sum_{t \ni e} y_t \leq 1, \forall e \in E^- \cup E^+, \quad (1)$$

where $T$ is the set of all bad triangles (non-necessarily disjoint) of the graph. By weak duality we have, $\sum_{t \in T} y_t \leq \text{LP}_{OPT} \leq OPT$ for all dual feasible solutions $\{y_t\}_{t \in T}$. Therefore, in order to get an approximation guarantee, it suffices to compare the cost $C^{\text{pivot}}$ with a carefully constructed dual feasible solution.

**LEMMA 3.6.** *Let $C^{pivot}$ be the number of disagreements incurred by the pivot clusters (Definition 3.4). We have that $\mathbb{E}[C^{pivot}] \leq 3 \cdot OPT$.*

**PROOF.** Let $T$ be the set of bad triangles. Recall the event $A_t$ that all nodes in $t \in T$ are active and one of the nodes in $t$ is chosen as a pivot (Line 7 of Algorithm 3) and let $\Pr[A_t] = p_t$. Our goal is to use the probabilities $p_t$ to find a feasible solution to the packing LP defined above.

Let $D_e$ be the event that Algorithm 3 creates a disagreement on $e$ and notice that $D_e \wedge A_t$ denotes the event that the disagreement caused by $A_t$ was charged on $e$. By the definition of $A_t$, this disagreement cannot be due to creating singleton clusters in Line 6 of Algorithm 3. Consider now the event $A_t$ and observe that, as we are iterating over a random permutation of the nodes, each node in $t$ has the same probability to be chosen as the pivot (recall that the nodes of $t$ are all active by definition of $A_t$). Furthermore, exactly one choice of pivot can cause $D_e$ for each $e \in t$. Hence, we have that $\Pr[D_e \mid A_t] = 1/3$ and therefore, $\Pr[D_e \wedge A_t] = \Pr[D_e \mid A_t] \cdot \Pr[A_t] = p_t/3$.

Consider the assignment $y_t = p_t/3$. We now show that this is a feasible solution for the dual LP in equation (1). This is because for all edges $e \in E^+ \cup E^-$ the events $\{D_e \wedge A_t\}_{t \ni e}$ are

disjoint from each other, and hence we have

$$\sum_{t \ni e} y_t = \sum_{t \ni e} \frac{p_t}{3} = \sum_{t \ni e} \Pr[D_e \wedge A_t] = \Pr[\cup_{t \ni e} D_e \wedge A_t] \leq 1 .$$

As this is a feasible packing, we have that $\sum_{t \ni e} p_t/3 \leq \mathrm{OPT}$. Finally, by Lemma 3.5

$$\mathbb{E}[C^{\mathrm{pivot}}] \leq \sum_t p_t = 3 \cdot \sum_{t \in T} \frac{p_t}{3} \leq 3 \cdot \mathrm{OPT} .$$ ■

## 3.3 Analyzing the Singleton Clusters

The goal of this section is to bound the number of disagreements caused by the singleton clusters created in Line 6 of Algorithm 3. The high-level idea is to show that for a node $u$ of degree $\deg(u)$, either $u$ is clustered by some pivot node after $O(n/\deg(u))$ iterations or most of its edges have been cut by pivot clusters. In the latter case, we relate the cost of the remaining edges of $u$ to the ones cut by the pivot clusters, and show that the remaining edges do not incur a large additional cost. We also need to account for singleton clusters where most of the edges are incident on other singleton clusters. For this, we will do a counting argument that shows that there cannot be many singleton clusters that have many edges to other singleton clusters.

**Charging the Edges Incident on the Singleton Clusters.** Now, our goal is to bound the number of edges cut by the singleton clusters created in Line 6 of Algorithm 3. For intuition, consider a node $u$ and its neighbors with a *smaller* degree, and suppose that $u$ will not be included in a pivot cluster. Furthermore, suppose that roughly half of its neighbors have a smaller degree. If any smaller degree neighbor $v$ is chosen according to the random permutation in the first (roughly) $n/\deg(u)$ iterations, then $v$ will be chosen as a pivot. As we will show, this implies that, in expectation, almost all (roughly a $(1-\varepsilon)$-fraction) of the smaller degree neighbors either join a pivot cluster or at least one of them will be chosen as a pivot which would include $u$ in a pivot cluster (Lemma 3.7). Once we have this, we can spread the disagreements on the remaining $\varepsilon$-fraction of the edges to smaller degree nodes to the edges cut by pivot clusters. As a technical challenge, we also need to account for nodes who have a few smaller degree neighbors to begin with. We use a counting argument (Lemma 3.9) to show that a large fraction of nodes must have many neighbors in pivot clusters, which allows us to also spread the cost of the nodes with few smaller degree neighbors.

Consider a node $u$ and let $N_i(u)$ be the set of nodes at the beginning of iteration $i$ such that for each $v \in N_i(u)$, we have that $\deg(v) \leq \deg(u)$ and $v$ is not in a pivot cluster. Let $\deg_i(u) = |N_i(u)|$.

**LEMMA 3.7.** *For each node u, at the beginning of iteration* $i = \lceil \tau_u \rceil$ *(recall,* $\tau_u = \frac{c}{\varepsilon} \cdot \frac{n \log n}{\deg(u)}$ *from Algorithm 1), the probability that u is active and* $\deg_i(u) > \varepsilon \cdot \deg(u)$ *is upper bounded by* $1/n^{c/2}$.

**PROOF.** We define $A_k$ the events that $u$ is active at the beginning of iteration $k$, and the events $B_k := \{\{\deg_k(u) > \varepsilon \deg(u)\} \cap A_k\}, \forall\, 1 \le k \le i$. We want to show that $\Pr[B_i] \le 1/n^{c/2}$. A useful property of these events is that $B_k \subseteq B_{k-1}, \forall\, 1 < k \le i$.

Using conditional probabilities we get that,

$$\Pr[B_i] = \Pr[B_i \cap B_{i-1}] = \Pr[B_i \mid B_{i-1}] \cdot \Pr[B_{i-1}] = \left(\prod_{k=2}^{i} \Pr[B_k \mid B_{k-1}]\right) \cdot \Pr[B_1].$$

In the following, we use the fact that if two events $\mathcal{E}_1$ and $\mathcal{E}_2$ are such that $\mathcal{E}_1 \subseteq \mathcal{E}_2$, then $\Pr[\mathcal{E}_1] \le \Pr[\mathcal{E}_2]$. We also use the fact that, conditioning on $B_{k-1}$ implies that at the beginning of iteration $k-1$, there are at least $\varepsilon \cdot \deg(u)$ nodes in $N_{k-1}(u)$.

$$\begin{aligned}
\Pr[B_k^c \mid B_{k-1}] &= \Pr\left[\left\{\deg_k(u) \le \varepsilon \cdot \deg(u)\right\} \cup A_k^c \mid B_{k-1}\right] \\
&\ge \Pr\left[u \text{ becomes inactive during iteration } k-1 \mid B_{k-1}\right] \\
&\ge \Pr\left[\text{a node in } N_{k-1}(u) \text{ becomes a pivot during iteration } k-1 \mid B_{k-1}\right] \\
&\ge \frac{\varepsilon \cdot \deg(u)}{n-k+1} \ge \frac{\varepsilon \cdot \deg(u)}{n}.
\end{aligned}$$

Hence, $\Pr[B_k | B_{k-1}] \le 1 - \varepsilon \deg(u)/n$. We finally get that

$$\Pr[B_i] \le \left(1 - \frac{\varepsilon \deg(u)}{n}\right)^{i-1} \le \left(1 - \frac{\varepsilon \deg(u)}{n}\right)^{i/2} \le \exp\left(-\frac{\varepsilon \deg(u)}{2n} \cdot \frac{c}{\varepsilon} \cdot \frac{n \log n}{\deg(u)}\right) \le \frac{1}{n^{c/2}}. \quad \blacksquare$$

**LEMMA 3.8.** *In all iterations $i$, all nodes $u$ that are put into singleton clusters in iteration $i$ (Line 6 of Algorithm 3) satisfy $\deg_i(u) \le \varepsilon \cdot \deg(u)$ with probability $1 - 1/n^{\alpha}$ where $\alpha := c/2 - 1 \gg 2$.*

**PROOF.** By Lemma 3.7 and union bound over all nodes, we can say that with probability at most $1/n^{\alpha}$, there exists a node $u$ such that at the beginning of iteration $i = \lceil \tau_u \rceil$, $u$ is active and $\deg_i(u) > \varepsilon \cdot \deg(u)$. Therefore, with high probability, for all nodes $u$, at the beginning of iteration $i = \lceil \tau_u \rceil$, either $u$ is already inactive or $\deg_i(u) \le \varepsilon \cdot \deg(u)$. This implies that, with high probability, if $u$ is put in a singleton cluster (Line 6 of Algorithm 3), which can happen only in iteration $i = \lceil \tau_u \rceil$, we have $\deg_i(u) \le \varepsilon \cdot \deg(u)$. $\blacksquare$

**Good Edges and Counting.** Consider a positive edge incident on a singleton cluster that contains a node $u$. Suppose that the singleton cluster was created in iteration $i$. We define an edge $e = \{u, v\}$ to be *good* if the other endpoint, node $v$, was included in a pivot cluster (Line 7 of Algorithm 3) in some iteration $j < i$. Otherwise, edge $e$ is *bad*. The sets $E^{\text{good}}$ and $E^{\text{bad}}$ give a partition of $E^{\text{sin}}$, the set of edges incident to singleton clusters. Intuitively, if an edge is good, we can charge it to the set $C^{\text{pivot}}$ which we know how to bound through Lemma 3.6. Furthermore, if we can show that most edges incident on singleton clusters are good, we can bound the cost of the bad edges.

**LEMMA 3.9.** *Conditioned on the high probability event of Lemma 3.8, $|E^{bad}| \le 2\varepsilon \cdot |E^{sin}|$.*

**PROOF.** For every node $u$, we define $\deg_i(u) = |N_i(u)|$ where $N_i(u)$ is the set of neighbors of $u$ such that for $v \in N_i(u)$, $\deg(v) \leq \deg(u)$ and $v$ is not in a pivot cluster.

For the analysis, let us consider the following orientation on the bad edges. Consider an iteration $i$, where a node $u$ is put into a singleton cluster in Line 6 of Algorithm 3. Notice that this implies that $i = \lceil \tau_u \rceil$. Then, we orient each unoriented edge from $u$ to $v$ for each neighbor $v$ such that $\deg(v) \leq \deg(u)$ and $v$ is not in a pivot cluster, i.e. we orient all edges between $u$ and $N_i(u)$ from $u$ to $N_i(u)$. Denote the out-degree of a node $u$ by $\deg_{\text{out}}(u)$, and notice that $\deg_{\text{out}}(u) = \deg_i(u)$. Notice that $\deg_{\text{out}}(u)$ is a random variable.

Our conditioning on the high probability event of Lemma 3.8 gives $\deg_i(u) \leq \varepsilon \cdot \deg(u)$.

Hence, the out-degree of each singleton node $u$ verifies $\deg_{\text{out}}(u) \leq \varepsilon \cdot \deg(u)$. Also, by definition, the out-degree of each non-singleton node is 0.

Let $V^{\text{sin}}$ be the set of nodes that are put in singleton clusters in Line 6 of Algorithm 3, and let $\mathbb{1}_{u \in V^{\text{sin}}}$ be the corresponding indicator random variable. Notice that $|V^{\text{sin}}| = \sum_{u \in V} \mathbb{1}_{u \in V^{\text{sin}}}$, $|E^{\text{bad}}|$ and $|E^{\text{sin}}|$ are random variables. By definition of the orientation,

$$|E^{\text{bad}}| = \sum_{u \in V} \deg_{\text{out}}(u) = \sum_{u \in V} \mathbb{1}_{u \in V^{\text{sin}}} \cdot \deg_{\text{out}}(u),$$

since $\mathbb{1}_{u \in V^{\text{sin}}} = 0$ implies $\deg_{\text{out}}(u) = 0$. By using Lemma 3.7, we have that

$$|E^{\text{bad}}| = \sum_{u \in V} \mathbb{1}_{u \in V^{\text{sin}}} \cdot \deg_{\text{out}}(u) \leq \sum_{u \in V} \mathbb{1}_{u \in V^{\text{sin}}} \cdot \varepsilon \cdot \deg(u).$$

By using the handshake lemma, we have that

$$\sum_{u \in V} \mathbb{1}_{u \in V^{\text{sin}}} \cdot \varepsilon \cdot \deg(u) \leq 2\varepsilon \cdot |E^{\text{sin}}| \,. \qquad \blacksquare$$

We now have in hand all the necessary results to be able to prove our main theorem, which was the following.

**THEOREM 3.1. (Restated)** *For any $\varepsilon \in (0, 1/4)$, Sequential Truncated-Pivot (Algorithm 3) is a $(3+\varepsilon)$-approximation algorithm to the correlation clustering problem. The approximation guarantee is in expectation.*

**PROOF.** Recall the following definitions.

— We denote the cost of the pivot clusters by $C^{\text{pivot}}$ (see Definition 3.4). This cost also covers the cost of the positive edges between pivot clusters and singleton clusters that were cut by the pivot clusters. These edges are called *good*, and the set of those edges is denoted by $E^{\text{good}}$.

— Bad edges are the positive edges incident on singleton clusters that were not cut by the pivot cluster. Either they are between singletons or the singleton was created before the pivot cluster. Denote those edges by $E^{\text{bad}}$.

— $E^{\text{sin}} = E^{\text{good}} \cup E^{\text{bad}}$

We can split the cost of Algorithm 3 into two parts. By Lemma 3.6, we have that $\mathbb{E}[C^{\text{pivot}}] \leq 3 \cdot \text{OPT}$. Let us define $D$ to be the event that, for all iterations $i$, all nodes $u$ that are put in singleton clusters in iteration $i$ satisfy $\deg_i(u) < \varepsilon \cdot \deg(u)$. By Lemma 3.8, $D$ is a high probability event. Then, by Lemma 3.9, we have that, conditioning on the high probability event $D$,

$$|E^{\text{bad}}| \leq 2\varepsilon \cdot |E^{\text{sin}}| \leq \frac{2\varepsilon}{1-2\varepsilon} \cdot |E^{\text{good}}| \leq 4\varepsilon \cdot C^{\text{pivot}} ,$$

where the last inequality holds because $\varepsilon < 1/4$. This inequality implies that $\mathbb{E}[|E^{\text{bad}}| \mid D] \leq 4\varepsilon \cdot \mathbb{E}[C^{\text{pivot}} \mid D]$. And therefore,

$$\begin{aligned}
\mathbb{E}[|E^{\text{bad}}|] &= \mathbb{E}[|E^{\text{bad}}| \mid D] \Pr[D] + \mathbb{E}[|E^{\text{bad}}| \mid \bar{D}] \cdot \Pr[\bar{D}] \\
&\leq 4\varepsilon \cdot \mathbb{E}[C^{\text{pivot}} \mid D] \cdot \left(1 - \frac{1}{n^c}\right) + n^2 \cdot \frac{1}{n^\alpha} \\
&\leq 4\varepsilon \cdot \mathbb{E}[C^{\text{pivot}} \mid D] + \frac{1}{n^{\alpha-2}} .
\end{aligned}$$

Also, notice that,

$$\begin{aligned}
\mathbb{E}\left[C^{\text{pivot}}\right] &= \mathbb{E}\left[C^{\text{pivot}} \mid D\right] \cdot \Pr[D] + \mathbb{E}\left[C^{\text{pivot}} \mid \bar{D}\right] \cdot \Pr[\bar{D}] \\
&\geq \mathbb{E}\left[C^{\text{pivot}} \mid D\right] \cdot \Pr[D] \\
&\geq \mathbb{E}\left[C^{\text{pivot}} \mid D\right] \cdot \left(1 - \frac{1}{n^\alpha}\right) \\
&\geq \mathbb{E}\left[C^{\text{pivot}} \mid D\right] - \frac{1}{n^{\alpha-2}} , \text{ since } \mathbb{E}\left[C^{\text{pivot}} \mid D\right] \leq n^2.
\end{aligned}$$

Which implies that

$$\mathbb{E}\left[C^{\text{pivot}} \mid D\right] \leq \mathbb{E}\left[C^{\text{pivot}}\right] + \frac{1}{n^{\alpha-2}} .$$

By combining the above observations, we have that the expected cost of Algorithm 3 is at most

$$\begin{aligned}
\mathbb{E}\left[C^{\text{pivot}} + |E^{\text{bad}}|\right] &= \mathbb{E}\left[C^{\text{pivot}}\right] + \mathbb{E}\left[|E^{\text{bad}}|\right] \\
&\leq \mathbb{E}\left[C^{\text{pivot}}\right] + 4\varepsilon \cdot \mathbb{E}[C^{\text{pivot}} \mid D] + \frac{1}{n^{\alpha-2}} \\
&\leq (1+4\varepsilon) \cdot \mathbb{E}\left[C^{\text{pivot}}\right] + \frac{1+4\varepsilon}{n^{\alpha-2}} \\
&\leq (3+12\varepsilon) \cdot \text{OPT} + \frac{1+4\varepsilon}{n^{\alpha-2}}.
\end{aligned}$$

We can substitute $\varepsilon' := 12\varepsilon$, where $\varepsilon$ can be arbitrarily small. Notice that if $\text{OPT} \geq 1$, then we have that $\mathbb{E}\left[C^{\text{pivot}} + |E^{\text{bad}}|\right] \leq (3+12\varepsilon) \cdot \text{OPT}$, which gives us a $(3+\varepsilon')$-approximation in expectation. ∎

**REMARK 3.10.** If $\text{OPT} = 0$, then the expected cost of our solution is $1/\text{poly}(n)$ according to the proof above, or equivalently, the expected cost of our solution is 0 with high probability.

**PROOF OF THEOREM 2.1.** Theorem 2.1 follows from Theorem 3.1 and Lemma 3.2. ■

## Acknowledgements

We would like to thank Moses Charikar, Soheil Behnezhad, Weiyun Ma, and Li-Yang Tan for pointing out an error in an earlier analysis of our correlation clustering algorithm. We would also like to thank Vihan Shah and Sepehr Assadi for pointing out that our algorithm works even in dynamic streaming. Finally, we thank Dennis Olivetti and Alkida Balliu for fruitful discussions.

Mélanie Cambus is supported by Research Council of Finland Grant 334238. Part of this work was done when Shreyas Pai was a postdoctoral fellow at Aalto University, supported by Research Council of Finland Grant 334238 and Helsinki Institute for Information Technology HIIT.

## References

[1] **Kook Jin Ahn**, **Graham Cormode**, **Sudipto Guha**, **Andrew McGregor**, **and Anthony Wirth**. Correlation clustering in data streams. *Algorithmica*, 83(7):1980–2017, 2021. DOI (3–5)

[2] **Kook Jin Ahn and Sudipto Guha**. Graph sparsification in the semi-streaming model. *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009*, pages 328–338, Berlin, Heidelberg. Springer Berlin Heidelberg, 2009. DOI (9)

[3] **Kook Jin Ahn**, **Sudipto Guha**, **and Andrew McGregor**. Analyzing graph structure via linear measurements. *Proceedings of the 2012 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 459–467, 2012. DOI (3)

[4] **Nir Ailon**, **Moses Charikar**, **and Alantha Newman**. Aggregating Inconsistent Information: Ranking and Clustering. *Journal of the ACM (JACM)*, 55(5):1–27, 2008. DOI (4, 5, 11–13)

[5] **Arvind Arasu**, **Christopher Ré**, **and Dan Suciu**. Large-Scale Deduplication with Constraints Using Dedupalog. *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009*, pages 952–963, 2009. DOI (2)

[6] **Sepehr Assadi**, **Andrew Chen**, **and Glenn Sun**. Deterministic graph coloring in the streaming model. *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 261–274, 2022. DOI (3)

[7] **Sepehr Assadi and Aditi Dudeja**. A simple semi-streaming algorithm for global minimum cuts. *2021 Symposium on Simplicity in Algorithms (SOSA)*. Society for Industrial and Applied Mathematics, 2021., pages 172–180. DOI (3)

[8] **Sepehr Assadi**, **Christian Konrad**, **Kheeran K. Naidu**, **and Janani Sundaresan**. O(log log n) passes is optimal for semi-streaming maximal independent set. *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, pages 847–858, Vancouver, BC, Canada. Association for Computing Machinery, 2024. DOI (4)

[9] **Sepehr Assadi and Chen Wang**. Sublinear Time and Space Algorithms for Correlation Clustering via Sparse-Dense Decompositions. *the Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, 10:1–10:20, 2022. DOI (3)

[10] **Nikhil Bansal**, **Avrim Blum**, **and Shuchi Chawla**. Correlation Clustering. *Machine learning*, 56(1):89–113, 2004. DOI (2, 3)

[11] **Neta Barkay**, **Ely Porat**, **and Bar Shalem**. Efficient sampling of non-strict turnstile data streams. *Theor. Comput. Sci.* 590:106–117, 2015. DOI (8)

[12] **Soheil Behnezhad**, **Moses Charikar**, **Weiyun Ma**, **and Li-Yang Tan**. Almost 3-Approximate Correlation Clustering in Constant Rounds. *the Proceeding of the Symp. on Foundations of Computer Science (FOCS)*, pages 720–731, 2022. DOI (3, 4)

[13] **Soheil Behnezhad**, **Moses Charikar**, **Weiyun Ma**, **and Li-Yang Tan**. Single-pass streaming algorithms for correlation clustering. *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 819–849. SIAM, 2023. DOI (3, 4, 6, 9)

[14] **Mélanie Cambus**, **Fabian Kuhn**, **Etna Lindy**, **Shreyas Pai**, **and Jara Uitto**. A (3 + $\varepsilon$)-approximate correlation clustering algorithm in dynamic streams. *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024*, pages 2861–2880. SIAM, 2024. DOI (1)

[15] **Nairen Cao**, **Vincent Cohen-Addad**, **Euiwoong Lee**, **Shi Li**, **Alantha Newman**, **and Lukas Vogl**. Understanding the cluster linear program for correlation clustering. *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC*, pages 1605–1616. ACM, 2024. DOI (4)

[16] **Sayak Chakrabarty and Konstantin Makarychev**. Single-pass pivot algorithm for correlation clustering. keep it simple! *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2023. DOI (3, 4, 6)

[17] **Moses Charikar**, **Venkatesan Guruswami**, **and Anthony Wirth**. Clustering with Qualitative Information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005. DOI (3, 4)

[18] **Yudong Chen**, **Sujay Sanghavi**, **and Huan Xu**. Improved graph clustering. *IEEE Trans. Inf. Theory*, 60(10):6440–6455, 2014. DOI (2)

[19] **Flavio Chierichetti**, **Nilesh Dalvi**, **and Ravi Kumar**. Correlation Clustering in Mapreduce. *the Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 641–650, 2014. DOI (2)

[20] **Vincent Cohen-Addad**, **Silvio Lattanzi**, **Slobodan Mitrović**, **Ashkan Norouzi-Fard**, **Nikos Parotsidis**, **and Jakub Tarnawski**. Correlation Clustering in Constant Many Parallel Rounds. *International Conference on Machine Learning (ICML)*, pages 2069–2078, 2021. DOI (3)

[21] **Vincent Cohen-Addad**, **Euiwoong Lee**, **Shi Li**, **and Alantha Newman**. Handling Correlated Rounding Error via Preclustering: A 1.73-approximation for Correlation Clustering. *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1082–1104, Los Alamitos, CA, USA. IEEE Computer Society, November 2023. DOI (4)

[22] **Vincent Cohen-Addad**, **Euiwoong Lee**, **and Alantha Newman**. Correlation clustering with sherali-adams. *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 651–661. IEEE, 2022. DOI (4)

[23] **Vincent Cohen-Addad**, **David Rasmussen Lolck**, **Marcin Pilipczuk**, **Mikkel Thorup**, **Shuyi Yan**, **and Hanwen Zhang**. Combinatorial correlation clustering. *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC*, pages 1617–1628, 2024. DOI (3)

[24] **Graham Cormode**, **Jacques Dark**, **and Christian Konrad**. Independent Sets in Vertex-Arrival Streams. *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 45:1–45:14, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. DOI (4)

[25] **Joan Feigenbaum**, **Sampath Kannan**, **Andrew McGregor**, **Siddharth Suri**, **and Jian Zhang**. Graph distances in the data-stream model. *SIAM J. Comput.* 38(5):1709–1727, 2008. DOI (2)

[26] **Joan Feigenbaum**, **Sampath Kannan**, **Andrew McGregor**, **Siddharth Suri**, **and Jian Zhang**. On Graph Problems in a Semi-Streaming Model. *Theor. Comput. Sci.* 348(2):207–216, 2005. DOI (2)

[27] **Sungwoong Kim**, **Sebastian Nowozin**, **Pushmeet Kohli**, **and Chang Yoo**. Image Segmentation Using Higher-Order Correlation Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36, 2011. DOI (2)

[28] **S. Muthukrishnan**. Data Streams: Algorithms and Applications. *Foundations and Trends® in Theoretical Computer Science*, 1(2):117–236, 2005. DOI (2)

[29] **Chaitanya Swamy**. Correlation Clustering: Maximizing Agreements via Semidefinite Programming. *the Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 526–527, 2004. DOI (4)

[30] **Grigory Yaroslavtsev and Adithya Vadapalli**. Massively Parallel Algorithms and Hardness for Single-linkage Clustering under $\ell_p$-distances. *the Proceedings of the International Conference on Machine Learning (ICML)*, pages 5596–5605, 2018. DOI (2)