Almost 3-Approximate Correlation Clustering
in Constant Rounds

Soheil Behnezhad*   Moses Charikar*   Weiyun Ma*   Li-Yang Tan*

Abstract

We study parallel algorithms for *correlation clustering*. Each pair among n objects is labeled as either "similar" or "dissimilar". The goal is to partition the objects into arbitrarily many clusters while minimizing the number of disagreements with the labels.

Our main result is an algorithm that for any $\varepsilon > 0$ obtains a $(3+\varepsilon)$-approximation in $O(1/\varepsilon)$ rounds (of models such as massively parallel computation, local, and semi-streaming). This is a culminating point for the rich literature on parallel correlation clustering. On the one hand, the approximation (almost) matches a natural barrier of 3 for combinatorial algorithms. On the other hand, the algorithm's round-complexity is essentially constant.

To achieve this result, we introduce a simple $O(1/\varepsilon)$-round parallel algorithm. Our main result is to provide an analysis of this algorithm, showing that it achieves a $(3+\varepsilon)$-approximation. Our analysis draws on new connections to sublinear-time algorithms. Specifically, it builds on the work of Yoshida, Yamamoto, and Ito [27] on bounding the "query complexity" of greedy maximal independent set. To our knowledge, this is the first application of this method in analyzing the approximation ratio of any algorithm.

---

*Department of Computer Science, Stanford University.

1 Introduction

We study parallel algorithms for the following *correlation clustering* problem. The input is a collection of objects and a complete labeling of the object-pairs as "similar" or "dissimilar". It would be convenient to model the labels by a graph $G = (V, E)$ where similar pairs are adjacent and dissimilar pairs are non-adjacent. The goal is to partition the vertex set V into *arbitrarily many* clusters, capturing the labels as closely as possible. Unless G is a collection of vertex-disjoint cliques, a perfect clustering satisfying all the labels does not exist. A natural objective is thus to find the clustering that minimizes *disagreements*[1]: That is, the number of edges that go across clusters plus the number of non-adjacent pairs inside the clusters.

In contrast to some other clustering problems such as k-means, k-median, or k-center, correlation clustering does not require the number of clusters to be pre-specified. This, as well as the fact that correlation clustering uses information about both similarity and dissimilarity of the pairs in its output makes it a desirable clustering method for various tasks. Examples of applications of correlation clustering include image segmentation [22], community detection [26], disambiguation tasks [21], automated labeling [1, 13], and document clustering [8], among others.

In many of these applications, the input tends to be by orders of magnitude larger than what a single machine can handle. This has motivated a long and rich body of work studying efficient parallel algorithms for this problem; see [11, 16, 24, 2, 18, 12, 17, 7] and the references therein.

In this paper, we continue the study of parallel correlation clustering algorithms. Our main result is that an (almost) 3-approximation can be obtained in constant rounds.

> **Result 1** (Informal – see Theorem 1.1)**.** *For any $\varepsilon > 0$, one can obtain a $(3 + \varepsilon)$-approximation of correlation clustering in $O(1/\varepsilon)$ parallel rounds.*

Result 1 is a culminating point for parallel correlation clustering. First, its round complexity is essentially constant. Second, 3-approximation is a natural target; it remains the best achieved by any combinatorial correlation clustering algorithm, even sequential ones, that do not solve an LP.

To put our result into perspective, let us first overview the prior work. Throughout this paper, let $n = |V|$ denote the number of vertices in G, $m = |E|$ denote the number of edges in G, and Δ denote the maximum degree in G.

1.1 State of Affairs on (Parallel) Correlation Clustering

Sequential algorithms: Correlation clustering was first introduced by Bansal, Blum, and Chawla [8, 9] who showed that it admits a (large) constant approximation in polynomial time. Several follow-up works improved the approximation ratio [14, 4, 5, 15]. The current best known is 2.06 by Chawla, Makarychev, Schramm, and Yaroslavtsev [15], which is obtained by rounding the solution to a natural LP. It is also known that the problem is APX-hard [14].

Among known combinatorial algorithms the best known approximation is 3. It is achieved by a surprisingly simple randomized algorithm of Ailon, Charikar, and Newman [4], known as the Pivot algorithm. It is worth noting that the Pivot algorithm is also useful in LP rounding; for instance Chawla et al. [15] first modify the input based on the LP solution, then run Pivot on the resulting graph. The Pivot algorithm iteratively picks a random vertex, clusters it with its

---

[1]Another natural objective is to maximize agreements which is "easier" for approximate algorithms [8, 14].

Approx	Rounds	Sublinear MPC	Semi-Streaming	Local	Reference
3	$O(n)$	✓	✓	✓	Ailon et al. [4, 5]
$3 + \varepsilon$	$O(\log n \cdot \log \Delta/\varepsilon)$	✓	✓	✓	Chierichetti et al. [16]
3	$O(\log^2 n)$	✓	✓	✓	Blelloch et al. [11]
3	$O(\log n)$	✓	✓	✓	Fischer and Noever [18, 19]
3	$O(\log \Delta \cdot \log \log n)$	✓	×	×	Cambus et al. [12]
3	$O(\log \log n)$	×	✓	×	Ahn et al. [2, 3]
701	$O(1)$	✓	✓	✓	Cohen-Addad et al. [17]
10^5	1	×	✓	×	Assadi and Wang [7]
$3 + \varepsilon$	$O(1/\varepsilon)$	✓	✓	✓	**This work.**

Table 1: Prior work on low-depth correlation clustering. Here $\varepsilon > 0$ can be made arbitrarily small. See Section 4 for the formal definition of the models.

remaining neighbors, removes this cluster from the graph, and recurses on the remaining graph. A slightly paraphrased, but still equivalent, variant of this algorithm reads as follows.

Algorithm PIVOT [4]:

- Draw a permutation π of the vertex set V uniformly at random.
- While G has at least one vertex:
  - Let v be the vertex in G with the lowest rank in π. Mark v as a *pivot*.
  - Put v and its (remaining) neighbors in a cluster C_v and remove the vertices of C_v from G.

Parallel algorithms: The algorithms noted above are all highly sequential. Even the simple PIVOT algorithm, as stated, may take $\Omega(n)$ rounds as it picks only one pivot in each round. This has led to a rich and beautiful line of work on both parallelizing the PIVOT algorithm and also devising new parallel algorithms from scratch [11, 16, 24, 2, 18, 12, 17, 7]. See Table 1 for an overview of these results in three models of *massively parallel computation* (MPC) with sublinear space, the *streaming* model with $\widetilde{O}(n)$ space, and the distributed local model. The formal definition of these models is not relevant for our discussion in this section, and is thus deferred to Section 4.

Parallelizing PIVOT: An early work of [16] was based on the idea of parallelizing PIVOT through picking multiple pivots independently in each round. For the approximation analysis to go through, it is important for all vertices of each round to have nearly the same chance of getting marked as pivots. Additionally, since the pivots must be non-adjacent, not too many vertices can be marked as pivots in parallel, leading to a round-complexity of $O(\log^2 n/\varepsilon)$ for a $(3 + \varepsilon)$-approximation.

Another beautiful line of work [11, 18] on parallelizing PIVOT, which is the closest to this paper in terms of techniques, is based on a parallel implementation of the so-called *randomized greedy maximal independent set* algorithm. This algorithm also picks multiple pivots in each round, but correlates these choices in a way that guarantees exactly the same output as PIVOT. In this algorithm, one first fixes a random permutation π as in PIVOT. Then in each round all vertices that come earlier than their (remaining) neighbors in π are marked as pivots in parallel, and are removed from the graph along with their neighbors. This repeats over the same permutation π until the graph becomes empty. It can be shown that the final set of pivots formed this way is exactly equal

to the set of pivots found by PIVOT over π. But because the decisions across different rounds are not independent, the parallel round complexity of this algorithm is more complicated to analyze. Blelloch, Fineman, and Shun [11] were the first to show that this process w.h.p. terminates in poly $\log n$ rounds. Fischer and Noever [18] later improved this to $O(\log n)$, which they proved is the correct bound for this algorithm by providing a matching lower bound of $\Omega(\log n)$.

Depending on the specific model of computation, the round-complexity of PIVOT can be further improved to sublogarithmic [2, 12], e.g. to $O(\log \log n)$ in the semi-streaming model [2]. But all these algorithms still require $\omega(1)$ rounds. See Table 1.

Constant Round Parallel Algorithms: An alternative line of work on parallel correlation clustering focuses on obtaining constant round algorithms [17, 7]. These algorithms do not attempt to parallelize PIVOT. Rather, they are based on the key new insight that for an $O(1)$-approximation, it suffices to find clusters that are either singletons or near-cliques. This helps getting around the intricacies of finding a maximal independent set (as in PIVOT) which in turn results in a much faster round complexity of $O(1)$. The main downside of solutions with only near-cliques and singleton clusters is that a large approximation is inherent to it (see [17, Remark 3.10] for an example). For instance, the approximations achieved by [17] and [7] are respectively 701 and 10^5.

Compared to the two lines of work discussed above on parallel correlation clustering, Result 1 achieves the best of both worlds. Its approximation ratio comes close to the 3-approximation of the first line of work, and its round-complexity is essentially constant.

1.2 Our Contribution

We introduce a new algorithm r-PIVOT which has a parameter $r \geq 1$ that adjusts the round-complexity. It proceeds in the same way as the parallel PIVOT discussed above for r rounds, then we truncate the process and no longer find pivots. As a result, unlike PIVOT, not every vertex will have a pivot among its neighbors. Thus, we have to be careful about how we form the clusters at the end. For technical reasons, even the vertices that do have pivots among their neighbors may form singleton clusters in our algorithm. We will discuss the intuition behind this perhaps counter-intuitive process of forming the clusters later in Section 2.

Algorithm r-PIVOT: Our r-round algorithm for correlation clustering.

- Draw a permutation π of the vertex set V uniformly at random.
- Initially every vertex is *unsettled*.
- For r rounds:
  - For any unsettled $v \in V$, mark v as a *pivot* if $\pi(v) < \pi(u)$ for all unsettled $u \in N(v)$.
  - Mark all pivots and any vertex adjacent to them as *settled*.
- Every pivot starts a cluster which includes itself. Then for every non-pivot vertex u:
  - If there is no pivot in $N(u)$ or if there exists an unsettled vertex $w \in N(u)$ whose rank is smaller than all the pivots in $N(u)$, then u forms a singleton cluster.
  - Otherwise u joins the cluster of the minimum rank pivot in $N(u)$.

Notation: Let us denote the cost paid by r-PIVOT for parameter r, permutation π, and input graph G by $\text{COST}_{r\text{-PIV}}(G, \pi)$. We write $\text{COST}_{\text{PIV}}(G, \pi)$ to denote the cost paid by PIVOT run on

permutation π. Let us also denote by $\mathrm{OPT}(G)$ the optimal correlation clustering cost of graph G.

Our main result is that our truncated algorithm r-PIVOT achieves almost the same approximation as the full fledged PIVOT algorithm, even if r is a rather small constant:

Theorem 1.1 (**Main Technical Result**)**.** *For any graph G and any $r \geq 1$,*

$$\mathbf{E}_\pi[\mathrm{COST}_{r\text{-}\mathrm{PIV}}(G, \pi)] \leq \mathbf{E}_\pi[\mathrm{COST}_{\mathrm{PIV}}(G, \pi)] + \frac{8}{2r - 1} \cdot \mathrm{OPT}(G).$$

Our proof of Theorem 1.1 is fundamentally different from how the original PIVOT algorithm was analyzed, and is based on a new connection to sublinear time algorithms. We elaborate more on the key intuitions behind our analysis in Section 2.

Combined with the 3-approximation guarantee of [5] for algorithm PIVOT, Theorem 1.1 implies:

Corollary 1.2. *For any graph G and any $r \geq 1$,*

$$\mathbf{E}_\pi[\mathrm{COST}_{r\text{-}\mathrm{PIV}}(G, \pi)] \leq \left(3 + \frac{8}{2r - 1}\right) \cdot \mathrm{OPT}(G).$$

Implications: Corollary 1.2 implies that running r-PIVOT for $r = O(1/\varepsilon)$ suffices for a $(3 + \varepsilon)$-approximation. This is useful because r-PIVOT can be implemented in $O(r)$ rounds of various models. In particular, we obtain the following results; see Section 4 for models/implementations.

Corollary 1.3 (MPC)**.** *For any $\varepsilon > 0$, there is a randomized $O(1/\varepsilon)$-round MPC algorithm that obtains a $(3 + \varepsilon)$-approximation of correlation clustering. The algorithm requires $O(n^\delta)$ space per machine where constant $\delta > 0$ can be made arbitrarily small and requires $O(m)$ total space.*

Corollary 1.4 (Streaming)**.** *For any $\varepsilon > 0$, there is a randomized $O(1/\varepsilon)$-pass streaming algorithm using $O(n \log n)$ bits of space that obtains a $(3 + \varepsilon)$-approximation of correlation clustering.*

Corollary 1.5 (Local)**.** *For any $\varepsilon > 0$, there is a randomized $O(1/\varepsilon)$-round local algorithm that obtains a $(3 + \varepsilon)$-approximation of correlation clustering using $O(\log n)$-bit messages.*

Corollary 1.6 (LCA)**.** *For any $\varepsilon > 0$, there is a randomized local computation algorithm (LCA) that obtains a $(3 + \varepsilon)$-approximation of correlation clustering in $\Delta^{O(1/\varepsilon)} \cdot \mathrm{poly} \log n$ time/space.*

Instance-wise Guarantee: It is worth emphasizing that our approximation guarantee of r-PIVOT in Theorem 1.1 is *instance-wise* close to what PIVOT achieves. That is, if for some input G the PIVOT algorithm obtains an α-approximation where $\alpha < 3$, then the 3-factors of all corollaries above for r-PIVOT also improve to α for this instance G. This is appealing for two main reasons:

- **Practical purposes:** If PIVOT performs better than its worst-case guarantee on certain input distributions, then so does r-PIVOT.

- **Rounding the natural LP in $O(1)$ rounds:** As discussed, [15] obtain a 2.06-approximation by first (locally) modifying the graph based on the LP solution, and then running PIVOT on the modified graph. Thus if we are given this LP solution in any of the models above, we can first modify the graph (this step is simple and local, so can be done efficiently in all these models) and then instead of PIVOT run r-PIVOT on it. Because of the instance-wise guarantee, this also leads to an (almost) 2.06-approximation, but now in only $O(1)$ rounds.

Future Work: Our work leaves several interesting questions especially in big data models where there is no direct notion of rounds. See Section 5 for some of these open problems.

2   A High-Level Overview of Our Techniques

In this section, we give some high-level intuitions behind both our algorithm and its analysis.

It would be useful to first compare the outputs of PIVOT and r-PIVOT when both algorithms are run on the same permutation π. Figure 1 provides such a comparision over an example for $r = 1$. We write $\mathcal{C}_{r\text{-PIV}}$ and $\mathcal{C}_{\text{PIV}}$ to denote the output clusters of r-PIVOT and PIVOT respectively, and use P_{PIV} and $P_{r\text{-PIV}}$ to denote their pivots respectively.
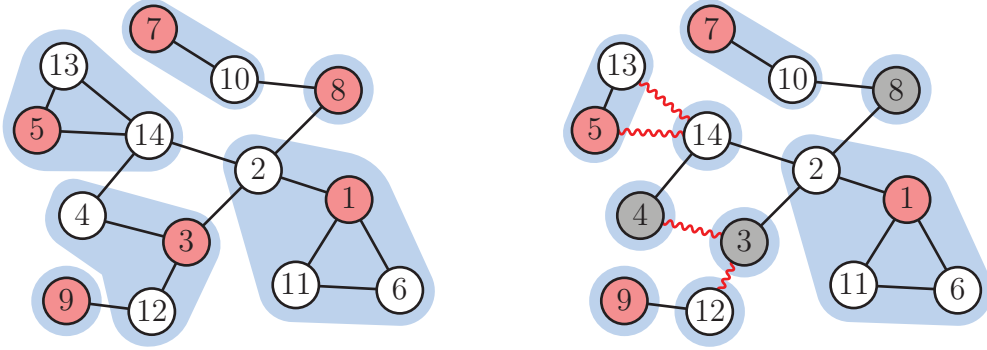

Figure 1: Comparison of PIVOT (left) with r-PIVOT (right) for $r = 1$ over the same permutation π whose ranks are shown on the vertices. The red vertices are pivots, the blue areas are the clusters, and the gray vertices are those marked as unsettled by r-PIVOT. The red zigzagged edges are those that r-PIVOT makes a mistake for but PIVOT does not.

Some Intuition Behind r-PIVOT: The final step of r-PIVOT, where we form the clusters, is specifically designed to ensure that $\mathcal{C}_{r\text{-PIV}}$ is a *refinement* of $\mathcal{C}_{\text{PIV}}$. That is, each cluster in $\mathcal{C}_{r\text{-PIV}}$ is completely inside a cluster in $\mathcal{C}_{\text{PIV}}$ (see Figure 1). To achieve this, if some vertex u joins the cluster of a pivot $v \in P_{r\text{-PIV}}$ in r-PIVOT, we make sure that u joins the cluster of v in PIVOT too. The unsettled vertices in r-PIVOT are precisely defined to guarantee this property. Intuitively, while any settled vertex has the same pivot status in both PIVOT and r-PIVOT, unsettled vertices in r-PIVOT may or may not be pivots in PIVOT. Now if a vertex u joins the cluster of a neighboring pivot v in r-PIVOT, we make sure that not only v is the lowest rank pivot neighboring u, but that v's rank is smaller than all unsettled neighbors of u too. In Figure 1, e.g., even though vertex 12 has an adjacent pivot 9, it decides not to join 9's cluster because of its unsettled neighbor 3. Note that 3 indeed ends up being a pivot in PIVOT, so this decision was crucial for $\mathcal{C}_{r\text{-PIV}}$ to be a refinement of $\mathcal{C}_{\text{PIV}}$. On the other hand, vertex 14 does not join the cluster of 5 because of its unsettled neighbor 4, but this time 4 is not a pivot in PIVOT which clusters 14 and 5 together. In our analysis, we have to make sure that our criteria which rather aggressively puts the vertices into singleton clusters does not hurt the approximation ratio of r-PIVOT much, compared to PIVOT.

Analysis of r-PIVOT: To analyze r-PIVOT, we couple it with PIVOT (over the same π) and bound the number of vertex pairs that are mistakenly clustered by r-PIVOT but correctly clustered by PIVOT. Let X be the set of such pairs. Our key contribution is to show that $\mathbf{E}\,|X| = O(\mathrm{OPT}(G)/r)$ (stated as Lemma 3.1). This is useful because if we run r-PIVOT for only $r = O(1/\varepsilon)$ steps, then we only pay an expected extra cost of $\mathbf{E}\,|X| \leq \varepsilon \cdot \mathrm{OPT}(G)$ compared to what PIVOT pays, which is essentially the guarantee of our main Theorem 1.1. Below, we present the key ideas behind how we prove this upper bound on $\mathbf{E}\,|X|$.

5

Our first insight is that no pair in X can be a non-edge. Note that the endpoints of any non-edge in X, by definition of X, should be clustered together in $\mathcal{C}_{r\text{-PIV}}$ but separated in $\mathcal{C}_{\text{PIV}}$. However, this would contradict our earlier discussion that $\mathcal{C}_{r\text{-PIV}}$ is a refinement of $\mathcal{C}_{\text{PIV}}$. Therefore, X is essentially the set of edges in E whose endpoints are separated by $\mathcal{C}_{r\text{-PIV}}$, but clustered together in $\mathcal{C}_{\text{PIV}}$. In Figure 1, the set X is illustarted by zigzagged red edges.

To show how we relate X to $\text{OPT}(G)$, let us first recall a standard framework of the literature in *charging bad triangles*. A bad triangle is a triplet of vertices $\{u, v, w\}$ such that two of the pairs $\{u,v\}, \{u,w\}, \{v,w\}$ are edges and one is a non-edge. Observe that no matter how the vertices of a bad triangle are clustered, at least one pair must be clustered incorrectly. Thus, if one finds β edge disjoint bad triangles in G, then $\text{OPT}(G) \geq \beta$. This implies that to show $|X| = O(\text{OPT}(G)/r)$, it suffices to charge $\alpha = \Omega(r)$ bad triangles for every edge in X, and guarantee that these $|X| \cdot \alpha$ bad triangles are all edge disjoint. Instead of a deterministic charging scheme, it would be more convenient to pick the charged bad triangles randomly. Doing so and by generalizing the same argument, one can show that instead of full edge disjointness of bad triangles, it suffices to prove that each vertex pair belongs, in expectation, to at most one charged bad triangle.

Key Idea I – Charging Triangles Far Away: Our charging scheme differs from those in the literature in that a mistake and the triangle that we charge it to may be far (at distance $\Omega(r)$) from each other. Previous charging schemes were all *highly local*, in that they charge any mistake to a bad triangle involving it. For instance, the 3-approximate analysis [5] of PIVOT charges any mistake $\{u, v\}$ to the bad triangle $\{u, v, w\}$ where w is the lowest rank pivot in $N(u) \cup N(v)$. It is impossible to analyze r-PIVOT with a local charging schemes. The reason is that a local charging scheme can be shown to imply a deterministic upper bound of $O(n \cdot \text{OPT}(G))$ on the clustering cost which, for instance, holds for PIVOT and *every* π. However, this deterministic upper bound does not hold for r-PIVOT which for some pathological permutation may have cost $\Omega(n^2/r)$ times $\text{OPT}(G)$. For more details about this, see Appendix B.

Key Idea II – Connections to Sublinear Algorithms: As discussed above, in our analysis we charge the mistakes $\{u, v\} \in X$ to triangles that are far away from u, v. To pick these triangles, we build on a query process developed originally for sublinear time algorithms [23, 27]. It is not hard to see that algorithm PIVOT marks a vertex v as a pivot iff there is no vertex u adjacent to v such that $\pi(u) < \pi(v)$ and u is a pivot. Therefore, to determine whether a vertex v is a pivot, it suffices to recursively query whether any of its lower rank neighbors are pivots. The total number of vertices that are (recursively) explored to answer such queries is known as the *query complexity* of random greedy maximal independent set (RGMIS), which is equivalent to the PIVOT algorithm. In a beautiful result, Yoshida, Yamamoto, and Ito [27] showed that in any n-vertex m-edge graph, an average vertex v has expected query complexity $O(m/n)$. This has been an influential work in the area of sublinear-time algorithms, where the goal is to explore a small part of the graph and estimate various global properties of it. In this work, we apply this method to a completely different context, and use it to analyze the approximation ratio of r-PIVOT.

We first propose a natural analog of the vertex query process for pairs of vertices (Section 3.3). Then to charge a mistake $\{u, v\} \in X$ to a bad triangle, we run this pair query process on $\{u, v\}$. We use $\{u, v\} \in X$ to show that there will be a moment when the stack of recursive calls by the query process, which will be a path in the graph, has size $\Theta(r)$. We then take the *first* moment that the stack gets to this size, and charge its last three vertices, which we prove must form a bad triangle. We then generalize the analysis of [27] in a non-trivial way (particularly by using several structural properties of the mistakes in X) to bound the number of times each vertex pair is charged,

arriving at our final approximation guarantee of Theorem 1.1. We give a more detailed high-level comparison between our analysis and [27] later in Section 3.5 after formalizing our charging scheme.

Correlation Clustering vs MIS: We finish this section with a brief comparison of correlation clustering with maximal independent set (MIS). Recall that the set P_{PIV} of the pivots formed by the PIVOT algorithm is an MIS of G. When we truncate parallel PIVOT after r rounds, we still have an independent set $P_{r\text{-PIV}}$ as our pivot-set but it is not necessarily maximal. In light of Theorem 1.1 which shows the clusters started by $P_{r\text{-PIV}}$, for some $r = O(1/\varepsilon)$, are almost as good as the clusters started by P_{PIV}, it could be interesting to see how $P_{r\text{-PIV}}$ and P_{PIV} compare in size. It might be natural to go as far as conjecturing that $\mathbf{E}\,|P_{r\text{-PIV}}| \geq (1-\varepsilon)\,\mathbf{E}\,|P_{\text{PIV}}|$ for $r = O(1/\varepsilon)$. But this is far from the truth. In fact, there are graphs for which $\mathbf{E}\,|P_{r\text{-PIV}}|$ is $n^{\Theta(1/r)}/r$ times smaller than $\mathbf{E}\,|P_{\text{PIV}}|$; see Appendix A. Therefore, it is perhaps surprising that $P_{r\text{-PIV}}$, while being significantly smaller than P_{PIV}, is almost as good for starting clusters and approximating correlation clustering.

3 The Analysis

In this section we prove Theorem 1.1 by analyzing the approximation ratio of r-PIVOT.

We use $\mathcal{C}_{\text{PIV}}$ and $\mathcal{C}_{r\text{-PIV}}$ to respectively denote the set of clusters returned by PIVOT and r-PIVOT. Let X be the set of pairs of vertices $\{u, v\}$ such that $\mathcal{C}_{r\text{-PIV}}$ disagrees with the label of $\{u, v\}$ but $\mathcal{C}_{\text{PIV}}$ agrees with it. In other words, X is the set of pairs for which $\mathcal{C}_{r\text{-PIV}}$ pays an extra cost for, compared to $\mathcal{C}_{\text{PIV}}$. We have

$$\text{COST}_{r\text{-PIV}}(G, \pi) \leq \text{COST}_{\text{PIV}}(G, \pi) + |X|. \tag{1}$$

Our plan for proving Theorem 1.1 is to show that the set X has a small expected size. In particular, the core of our proof is the following bound on X which immediately proves Theorem 1.1.

Lemma 3.1. *For any $r \geq 1$, $\mathbf{E}_\pi\,|X| \leq \frac{8}{2r-1} \cdot \text{OPT}(G)$.*

Proof of Theorem 1.1 via Lemma 3.1. By taking expectation over π on both sides of Eq (1) and applying Lemma 3.1, we get

$$\mathbf{E}_\pi[\text{COST}_{r\text{-PIV}}(G, \pi)] \leq \mathbf{E}_\pi[\text{COST}_{\text{PIV}}(G, \pi)] + \mathbf{E}_\pi\,|X|$$
$$\leq \mathbf{E}_\pi[\text{COST}_{\text{PIV}}(G, \pi)] + \frac{8}{2r-1} \cdot \text{OPT}(G). \qquad \square$$

We prove Lemma 3.1 in the rest of this section.

3.1 Charging Schemes

We first recall a standard framework of the literature in *charging bad triangles* (see e.g. [5]). We say three distinct vertices $\{a, b, c\}$ in V form a *bad triangle* if exactly two of the pairs $\{a, b\}, \{a, c\}, \{b, c\}$ belongs to E.

We say an algorithm $\mathcal{S}$ is a *charging scheme* for X, if $\mathcal{S}$ charges every pair in X to a bad triangle of the input. We say $\mathcal{S}$ has *width* w if for every pair of distinct vertices $a, b \in V$, the expected number of charges to bad triangles involving both a and b is upper bounded by w, where the

expectation is taken over π and the randomization of $\mathcal{S}$. We note that a triangle can be charged multiple times for different mistakes, but all of these charges must be counted in analyzing the width of the charging scheme. The following lemma shows that one can bound the expected size of X in terms of the width of a charging scheme:

Lemma 3.2. *If there exists a charging scheme $\mathcal{S}$ for X that has width w, then*

$$\mathbf{E}_\pi[|X|] \leq w \cdot \mathrm{OPT}(G).$$

Lemma 3.2 is a standard result in the framework of charging costs to bad triangles. For completeness, we provide a proof in Appendix C.

Therefore, to prove Lemma 3.1, our plan is to design a charging scheme for X that has width $\frac{8}{2r-1}$ for any $r \geq 1$. We define and analyze our charging scheme in Section 3.4. To that end, we first give a characterization of the pairs in X by comparing the clusterings $\mathcal{C}_{\mathrm{PIV}}$ and $\mathcal{C}_{r\text{-}\mathrm{PIV}}$ in Section 3.2. Then in Section 3.3, we associate bad triangles to the pairs in X to be charged.

3.2 Characterization of Pairs in X

In this subsection, we introduce some notations for the clusterings $\mathcal{C}_{\mathrm{PIV}}$ and $\mathcal{C}_{r\text{-}\mathrm{PIV}}$ returned by PIVOT and r-PIVOT respectively. We show that $\mathcal{C}_{r\text{-}\mathrm{PIV}}$ is a certain refinement of $\mathcal{C}_{\mathrm{PIV}}$ (Claim 3.5), and that all pairs in X are edges in the graph (Claim 3.6). This leads to our characterization of the pairs in X (Lemma 3.8), which singles out an unsettled vertex with small rank in the neighborhood of each pair in X.

Throughout this subsection, we fix a permutation π over V.

Definition 3.3 (Pivot sets). *Let P_{PIV} and $P_{r\text{-}PIV}$ to respectively denote the set of pivots marked by PIVOT and r-PIVOT both run on the same permutation π.*

Clearly, $P_{r\text{-}\mathrm{PIV}} \subseteq P_{\mathrm{PIV}}$. Moreover, every cluster in $\mathcal{C}_{\mathrm{PIV}}$ contains a unique pivot, and every cluster in $\mathcal{C}_{r\text{-}\mathrm{PIV}}$ that is not singleton contains a unique pivot.

Definition 3.4 (Pivot of a vertex). *For every $v \in V$, let $p_v \in P_{PIV}$ denote the pivot of the cluster in $\mathcal{C}_{PIV}$ that contains v. We say that p_v is the pivot of v in PIVOT.*

Note that $p_v = v$ if and only if $v \in P_{\mathrm{PIV}}$. Moreover, $\pi(p_v) \leq \pi(v)$ and p_v is the vertex in $(N(v) \cup \{v\}) \cap P_{\mathrm{PIV}}$ with minimum rank under π. Our next claim shows that the clustering $\mathcal{C}_{r\text{-}\mathrm{PIV}}$ refines $\mathcal{C}_{\mathrm{PIV}}$ in a particular way. See Figure 1.

Claim 3.5. *For any cluster C in $\mathcal{C}_{PIV}$, if v is the pivot of C, then C is partitioned by distinct clusters $C'_1, \ldots, C'_k$ in $\mathcal{C}_{r\text{-}PIV}$, i.e.*
$$C = C'_1 \cup C'_2 \cup \cdots C'_k,$$
such that $v \in C'_1$ and the remaining clusters $C'_2, \ldots, C'_k$ are all singletons. In particular, any cluster C' in $\mathcal{C}_{r\text{-}PIV}$ is contained in a cluster in $\mathcal{C}_{PIV}$.

Proof. We first show that any cluster $C' \in \mathcal{C}_{r\text{-}\mathrm{PIV}}$ that intersects C but does not contain the pivot v of C must be singleton. Suppose otherwise that C' is not singleton. Then C' contains a unique pivot $v' \in P_{r\text{-}\mathrm{PIV}}$, which must be different from v since $v \notin C'$. In particular, v' cannot be contained in C since otherwise C would contain two different pivots v and v' in P_{PIV}. Now take a vertex

8

$u \in C \cap C'$, which must be different from both v and v'. Note that $v, v' \in N(u)$ and $v = p_u$. Thus we have $\pi(v) < \pi(v')$. Then, in r-PIVOT, either v is identified as a pivot, in which case u would prefer to join the cluster of v in the last step, or v is unsettled, in which case u would form a singleton cluster. Either contradicts that u joins the cluster of v' in $\mathcal{C}_{r\text{-PIV}}$.

It remains to show that the cluster $C'_1 \in \mathcal{C}_{r\text{-PIV}}$ that contains v is contained in C. Suppose otherwise that there exists a vertex $u \in C'_1 \setminus C$. Then u is contained in a cluster $C'' \in \mathcal{C}_{\text{PIV}}$ other than C, and the pivot v'' of C'' is different from v. Since C'_1 already contains the pivot v, we have $v'' \notin C'_1$. By the first part of the proof applied to C'', we see that C'_1 must be a singleton cluster that only contains u, which is a contradiction. Therefore, no such u exists and $C'_1 \subseteq C$. $\square$

As a consequence, we show that all pairs in X are edges in the graph:

Claim 3.6. $X \subseteq E$.

Proof. Suppose otherwise that there exists a pair $\{u, v\} \in X \setminus E$. Since $\mathcal{C}_{r\text{-PIV}}$ disagrees with the label of $\{u, v\}$, u and v must belong to the same cluster $C' \in \mathcal{C}_{r\text{-PIV}}$. But by Claim 3.5, C' must be contained in a cluster $C \in \mathcal{C}_{\text{PIV}}$, which implies that $\mathcal{C}_{\text{PIV}}$ also disagrees with the minus label of $\{u, v\}$. This is a contradiction. $\square$

Therefore, for $\{u, v\} \in X$, since $\mathcal{C}_{\text{PIV}}$ agrees with the plus label of $\{u, v\}$, we have $p_u = p_v$.

Definition 3.7 (Common pivot of a pair in X). *For $\{u, v\} \in X$, let $p_{\{u,v\}} = p_u = p_v \in P_{PIV}$ denote the common pivot of u and v in PIVOT.*

Note that for every $\{u, v\} \in X$, $\pi(p_{\{u,v\}}) \leq \min\{\pi(u), \pi(v)\}$ and $p_{\{u,v\}}$ is the vertex in $(N(u) \cup N(v)) \cap P_{\text{PIV}}$ with minimum rank under π. Based on the last step of r-PIVOT, we can characterize the pairs in X as follows:

Lemma 3.8. *For every $\{u, v\} \in X$, one of the following holds after r rounds in r-PIVOT:*

(i) $p_{\{u,v\}}$ *is unsettled.*

(ii) $p_{\{u,v\}}$ *is settled. Moreover, at least one of u and v is singleton and has an unsettled, non-pivot neighbor $w_{\{u,v\}} \notin P_{PIV}$ with $\pi(w_{\{u,v\}}) < \pi(p_{\{u,v\}})$.*

Proof. It suffices to show that in Case (ii), at least one of u and v is singleton and has an unsettled neighbor $w_{\{u,v\}} \notin P_{\text{PIV}}$ with $\pi(w_{\{u,v\}}) < \pi(p_{\{u,v\}})$. Let C'_u (resp. C'_v) be the cluster in $\mathcal{C}_{r\text{-PIV}}$ that contains u (resp. v). Note that $C'_u \neq C'_v$. By Claim 3.5, both C'_u and C'_v are contained in the cluster C in $\mathcal{C}_{\text{PIV}}$ started by the pivot $p_{\{u,v\}}$. Then at least one of C'_u, C'_v cannot contain $p_{\{u,v\}}$, which must thus be singleton again by Claim 3.5. Say $C'_v = \{v\}$ is singleton. Since $p_{\{u,v\}}$ is settled, by the last step of r-PIVOT, v must have an unsettled neighbor $w_{\{u,v\}} \in N(v)$ with $\pi(w_{\{u,v\}}) < \pi(p_{\{u,v\}})$. In addition, $p_{\{u,v\}} = p_v$ is the pivot that v joins in r-PIVOT, which is the pivot neighbor of v with minimum rank under π. It follows that $w_{\{u,v\}} \notin P_{\text{PIV}}$. $\square$

3.3 Vertex and Pair Oracles

In this subsection, we define the following vertex and pair oracles. The vertex oracle locally determines whether a given vertex v is part of the greedy MIS over permutation π, or equivalently in our context, whether v is a pivot in the PIVOT algorithm. The vertex oracle was first defined by

Nguyen and Onak [23] and was further analyzed by Yoshida, Yamamoto, and Ito [27]. To analyze the pairs in X, we introduce a counterpart of the vertex oracle for pairs.

Throughout this subsection, we again fix a permutation π over V.

---

1 Function Vertex(v):
2   Let $w_1, \ldots, w_d$ be vertices in $N(v)$ s.t. $\pi(w_1) < \ldots < \pi(w_d) < \pi(v)$.
3   **for** i *in* $1 \ldots d$ **do**
4   **if** Vertex(w_i) $= 1$ **then return** 0
5   **return** 1
6 Function Pair(u, v):
7   Let $w_1, \ldots, w_d$ be vertices in $N(u) \cup N(v)$ s.t. $\pi(w_1) < \ldots < \pi(w_d) < \min\{\pi(u), \pi(v)\}$.
8   **for** i *in* $1 \ldots d$ **do**
9   **if** Vertex(w_i) $= 1$ **then return** 0
10   **return** 1

For a vertex $v \in V$, Vertex(v) returns 1 if and only if v is identified as a pivot by PIVOT. As for a pair $\{u, v\}$ in X, it is straightforward to see that Pair(u, v) returns 1 if and only if the common pivot $p_{\{u,v\}}$ of u and v in PIVOT turns out to be one of u, v.

Definition 3.9. *We say that a vertex $v \in V$ (resp. pair $\{u, v\} \in X$) directly queries a vertex $z \in V$ if* Vertex(v) *(resp.* Pair(u, v)*) directly calls* Vertex(z).

We will be interested in the set of vertices directly queried by a vertex or a pair in X:

Observation 3.10. *It holds that:*

(a) *For a pivot $v \in P_{PIV}$, it directly queries all neighbors $z \in N(v)$ with rank $\pi(z) < \pi(v)$ and no other vertex. In particular, v does not directly query any pivots in P_{PIV}.*

(b) *For a non-pivot $v \notin P_{PIV}$, it directly queries all neighbors $z \in N(v)$ with rank $\pi(z) \leq \pi(p_v)$ and no other vertex. In particular, p_v is the only pivot in P_{PIV} that v directly queries.*

(c) *For a pair $\{u, v\} \in X$ such that $p_{\{u,v\}} \in \{u, v\}$, it directly queries all neighbors $z \in N(u) \cup N(v)$ with rank $\pi(z) < \pi(p_{\{u,v\}})$ and no other vertex. In particular, $\{u, v\}$ does not directly query any pivots in P_{PIV}.*

(d) *For a pair $\{u, v\} \in X$ such that $p_{\{u,v\}} \notin \{u, v\}$, it directly queries all neighbors $z \in N(u) \cup N(v)$ with rank $\pi(z) \leq \pi(p_{\{u,v\}})$ and no other vertex. In particular, $p_{\{u,v\}}$ is the only pivot in P_{PIV} that $\{u, v\}$ directly queries.*

In our analysis, we will focus on the stack of *recursive* calls to function Vertex when we call Pair(u, v) for a pair $\{u, v\} \in X$. Our first insight is that, there is a moment when this stack includes $\Theta(r)$ elements.

Claim 3.11. *Let $\{u, v\} \in X$ and $\ell \leq 2r$. When we call* Pair(u, v)*, at some point the stack of recursive calls to* Vertex *includes exactly ℓ elements.*

To prevent interruptions to the flow of this part, we defer the proof of Claim 3.11 to Section 3.7.

Of particular importance to our analysis, is the *first* moment that the stack of recursive calls to Vertex when we call Pair(u, v) for $\{u, v\} \in X$ reaches a certain size.

Definition 3.12. *Let $\{u, v\} \in X$ and consider the stack of recursive calls to* Vertex *when we call* Pair(u, v)*. For each $\ell \in [2, 2r]$, we denote by $S_\ell(u, v)$ the ordered list of the elements in the stack the first time that it includes ℓ elements.*

For example $S_4(u, v) = (w_1, w_2, w_3, w_4)$ implies that $\{u, v\}$ directly queries w_1, w_1 directly queries w_2, w_2 directly queries w_3, w_3 directly queries w_4, and the first time after calling Pair(u, v) that the stack has 4 elements, only $w_1, w_2, w_3,$ and w_4 are in it. Claim 3.11 ensures that $S_\ell(u, v)$ is defined for any $\{u, v\} \in X$ and $\ell \le 2r$.

We now construct a path based on $S_\ell(u, v)$ by attaching u and v to the front in a particular order:

Definition 3.13. *Let $\{u, v\} \in X$, $\ell \in [2, 2r]$, and $S_\ell(u, v) = (w_1, \ldots, w_\ell)$. We define $P_\ell(u, v)$ to be the ordered list of $\ell + 2$ vertices defined as:*

$$P_\ell(u, v) = \begin{cases} (v, u, w_1, \ldots, w_\ell) & \text{if } w_1 \text{ is directly queried by } u \text{ but not } v, \\ (u, v, w_1, \ldots, w_\ell) & \text{if } w_1 \text{ is directly queried by } v \text{ but not } u, \\ (v, u, w_1, \ldots, w_\ell) & \text{if } w_1 \text{ is directly queried by both } u \text{ and } v, \pi(u) < \pi(v). \end{cases}$$

In other words, we choose the order of u, v to ensure that w_1 is directly queried by the vertex that precedes it. If both u and v directly query w_1, we choose the order in a way that the ranks of the vertices in $P_\ell(u, v)$ are in descending order. Note that the first vertex w_1 in the stack $S_\ell(u, v)$ is directly queried by $\{u, v\}$, and it follows from Observation 3.10 that w_1 is directly queried by either u or v. Thus $P_\ell(u, v)$ is always defined. Moreover, any two consecutive vertices in $P_\ell(u, v)$ form an edge in E, so it indeed specifies a path in G. The following observation is immediate from the construction of $S_\ell(u, v)$ and $P_\ell(u, v)$:

Observation 3.14. *Let $\{u, v\} \in X$, $\ell \in [2, 2r]$, and write*

$$P_\ell(u, v) = (w_1, \ldots, w_{\ell+2}).$$

Then for each $i \in [3, \ell + 2]$, we have $\pi(w_i) < \pi(w_{i-1})$, and w_{i-1} directly queries w_i.

Our charging scheme for X relies crucially on the following claim:

Claim 3.15. *For $\{u, v\} \in X$ and $\ell \in [2, 2r]$, the last three vertices in $P_\ell(u, v)$ form a bad triangle.*

Proof. We write
$$P_\ell(u, v) = (w_1, w_2, \ldots, w_{\ell+2}).$$
To show that $\{w_\ell, w_{\ell+1}, w_{\ell+2}\}$ form a bad triangle, it suffices to show that $(w_\ell, w_{\ell+2}) \notin E$. We assume otherwise that $(w_\ell, w_{\ell+2}) \in E$ and argue by contradiction. Note by Observation 3.14 that

$$\pi(w_\ell) > \pi(w_{\ell+1}) > \pi(w_{\ell+2}),$$

w_ℓ directly queries $w_{\ell+1}$, and $w_{\ell+1}$ directly queries $w_{\ell+2}$.

11

Suppose first that $w_{\ell+2} \in P_{\mathrm{PIV}}$ is a pivot. Since $w_{\ell+2}$ is also a neighbor of w_ℓ, its rank under π must be at least that of p_{w_ℓ}. But then $\pi(w_{\ell+1}) > \pi(w_{\ell+2}) \geq \pi(p_{w_\ell})$, which by Observation 3.10 (b) implies w_ℓ does not directly query $w_{\ell+1}$. This is a contradiction.

Suppose on the other hand that $w_{\ell+2} \notin P_{\mathrm{PIV}}$. Then by Observation 3.10 (b), $w_{\ell+2}$ directly queries $p_{w_{\ell+2}}$. Moreover, since w_ℓ directly queries $w_{\ell+1}$ and $w_{\ell+2}$ is a neighbor of w_ℓ with rank smaller than $w_{\ell+1}$, w_ℓ also directly queries $w_{\ell+2}$. Therefore, when we call $\mathtt{Pair}(u,v)$, at some point the stack of recursive calls to $\mathtt{Vertex}$ consists of the following ℓ elements:

$$(w_3, \ldots, w_\ell, w_{\ell+2}, p_{w_{\ell+2}}),$$

which happens before the stack consists of $S_\ell(u,v) = (w_3, \ldots, w_\ell, w_{\ell+1}, w_{\ell+2})$. This contradicts that $S_\ell(u,v)$ is the list of elements in the stack when it *first* reaches ℓ elements. $\qquad\square$

3.4 Our Charging Scheme

We can now define our charging scheme for X.

The Charging Scheme:

- Pick ℓ from $[2, 2r]$ uniformly at random.

- For any $(u,v) \in X$ charge the bad triangle formed by the last three vertices of $P_\ell(u,v)$.

  (Here the existence of $P_\ell(u,v)$ (or $S_\ell(u,v)$) follows from Claim 3.11 and its last three vertices form a bad triangle by Claim 3.15.)

Our main result is the following bound on the width of our charging scheme:

Lemma 3.16. *Our charging scheme for X has width $\frac{8}{2r-1}$ for any $r \geq 1$.*

Note that by Lemma 3.2, Lemma 3.16 immediately proves Lemma 3.1.

Next, we focus on proving Lemma 3.16.

3.5 The High Level Approach and Relation to [27]

We first give a high-level summary of our plan. To bound the number of charges to bad triangles involving two fixed vertices $a, b \in V$, we are led to bound the number of pairs of a permutation π of V and a path $P_\ell(u,v)$, initiated by the pair oracle $\mathtt{Pair}$ called on a mistake $\{u,v\}$ in X under π, that ends at a bad triangle involving both a and b. To do this, we construct a new permutation $\tilde{\pi}$ from π by rotating the ranks of vertices on the path $P_\ell(u,v)$ in a certain way while fixing the ranks of the other vertices. Let us temporarily denote $\tilde{\pi} = \phi(\pi, P_\ell(u,v))$.

In the case $\{a, b\}$ is an edge in E, the map ϕ (illustrated as Figure 2) is the same as a construction by Yoshida, Yamamoto, and Ito [27], which they used to bound the number of times the recursive query process of the vertex oracle $\mathtt{Vertex}$ passes through a fixed edge $e \in E$, and thereby bound the query complexity of the vertex oracle. The key property of this construction is that the preimage of any permutation $\tilde{\pi}$ under the map ϕ can only have a small constant size. For [27], this implies that in expectation over π there are only $O(1)$ recursive queries that pass through e. For us, this implies that in expectation over π there are only $O(1)$ possibilities for $P_\ell(u,v)$. However, in our

case, we need to consider the pair oracle in order to charge the pairs in X, and a path $P_\ell(u, v)$ does not necessarily specify a valid sequence of recursive queries initiated by the vertex oracle. There in fact may be many more than $O(1)$ such paths ending at a single edge. To get around this, we rely crucially on two additional properties in our case. First, each path $P_\ell(u, v)$ starts at a extra mistake $\{u, v\} \in X$ made by r-PIVOT as compared to PIVOT, which in particular is an *edge* whose two endpoints share a *common pivot* in PIVOT. This provides new incidence relations among the vertices that help us rule out certain possibilities for $P_\ell(u, v)$. Second, each path $P_\ell(u, v)$ is given by the *first* moment when the corresponding stack of recursive calls to `Vertex` reaches a certain size. This places additional constraints on the preimages of ϕ.

An additional challenge in our case as compared to [27] is that, we also need to consider pairs $\{a, b\}$ that are non-edges. In that case, we propose a modified construction of the rotation map ϕ (illustrated as Figure 4), which we show preserves the key property that the preimage of any permutation has a small constant size. Our argument for this property builds on that for the case where $\{a, b\} \in E$ but also carefully rules out several new edge cases, again by using the two additional properties in our case.

3.6 Proof of Lemma 3.16

Now we start the proof. Given distinct vertices $a, b \in V$ and a permutation π over V, we introduce a set $R_\pi(a, b)$ that accounts for the charges under π to bad triangles involving both a and b that come from paths passing through a before b. Formally, we define $R_\pi(a, b)$ to be the set of $(\{u, v\}, \ell)$, where $\{u, v\} \in X$ and $\ell \in [2, 2r]$ is an integer, such that under π:

- the last three vertices in the ordered list $P_\ell(u, v)$ are (a, b, c) or (c, a, b) for some $c \in V$, if $\{a, b\} \in E$;

- the last three vertices in the ordered list $P_\ell(u, v)$ are (a, c, b) for some $c \in V$, if $\{a, b\} \notin E$.

To bound the number of charges to bad triangles involving both a and b and prove Lemma 3.16, we bound the size of $R_\pi(a, b)$ in the following two lemmas:

Lemma 3.17. *Let $a, b \in V$ be distinct vertices such that $\{a, b\} \in E$. Then, for any $r \geq 1$, we have*

$$\mathbf{E}_\pi[|R_\pi(a, b)|] \leq 4.$$

Lemma 3.18. *Let $a, b \in V$ be distinct vertices such that $\{a, b\} \notin E$. Then, for any $r \geq 1$, we have*

$$\mathbf{E}_\pi[|R_\pi(a, b)|] \leq 2.$$

We first show that Lemma 3.17 and Lemma 3.18 together imply Lemma 3.16:

Proof of Lemma 3.16 via Lemma 3.17 and Lemma 3.18. Let $a, b \in V$ be distinct vertices. We show that the expected number of charges to bad triangles involving both a and b is upper bounded by $\frac{8}{2r-1}$, where the expectation is taken over π and ℓ. In our charging scheme, a charge to a bad triangle involving both a and b comes from an ordered list $P_\ell(u, v)$ under some permutation π and for some $\{u, v\} \in X$ and $\ell \in [2, 2r]$, such that the bad triangle consists of the last three vertices in $P_\ell(u, v)$. In the case $\{a, b\} \in E$, a and b are among the last three vertices in $P_\ell(u, v)$ and are

adjacent, although they can appear in either order. Since ℓ is chosen uniformly at random, by Lemma 3.17, the expected number of charges to bad triangles involving both a and b is at most

$$\frac{1}{2r-1}\left(\mathbf{E}_\pi[|R_\pi(a,b)|] + \mathbf{E}_\pi[|R_\pi(b,a)|]\right) \leq \frac{8}{2r-1}.$$

In the other case $\{a,b\} \notin E$, a and b, in either order, are the last and third-to-last vertices in $P_\ell(u,v)$ respectively. Since ℓ is chosen uniformly at random, by Lemma 3.18, the expected number of charges to bad triangles involving both a and b is at most

$$\frac{1}{2r-1}\left(\mathbf{E}_\pi[|R_\pi(a,b)|] + \mathbf{E}_\pi[|R_\pi(b,a)|]\right) \leq \frac{4}{2r-1} < \frac{8}{2r-1}. \qquad \square$$

3.6.1 Proof of Lemma 3.17

Throughout this subsection, we fix distinct vertices $a, b \in V$ such that $\{a,b\} \in E$. Given a permutation π over V and $i \in [n]$, we denote by $\pi_i \in V$ the vertex with rank i under π.

In our analysis, we break down the charges to bad triangles involving both a and b by specifying the *ranks* of the pair of vertices in X that initiates the charge. Given $i, j \in [n]$ with $i < j$, we introduce a set $T^{i,j}(a,b)$ that accounts for the charges that come from paths originating at vertices with ranks i and j under some permutation π and passing through a before b. Formally, we define $T^{i,j}(a,b)$ to be the set of pairs (π, ℓ) of a permutation π and an integer $\ell \in [2, 2r]$ such that under π, $\{\pi_i, \pi_j\} \in X$ and the last three vertices in the ordered list $P_\ell(\pi_i, \pi_j)$ are (a,b,c) or (c,a,b) for some $c \in V$. We will prove the following bound on the size of $T^{i,j}(a,b)$:

Claim 3.19. *For any $i, j \in [n]$ with $i < j$ and any $r \geq 1$, we have*

$$|T^{i,j}(a,b)| \leq 8(n-2)!.$$

We first use Claim 3.19 to prove Lemma 3.17:

Proof of Lemma 3.17 via Claim 3.19. Our main observation is that, as we range through all possibilities for the ranks i and j of the pair that initiates the charge, the sets $T^{i,j}(a,b)$ form a partition of the union of the sets $R_\pi(a,b)$ as π ranges through all permutations. To make this precise, we define

$$R(a,b) = \{(\pi, \{u,v\}, \ell) \mid (\{u,v\}, \ell) \in R_\pi(a,b)\},$$
$$T(a,b) = \{(i, j, \pi, \ell) \mid i < j, (\pi, \ell) \in T^{i,j}(a,b)\}.$$

Then, it is straightforward to see that the map

$$R(a,b) \to T(a,b), \quad (\pi, \{u,v\}, \ell) \mapsto (\min\{\pi(u), \pi(v)\}, \max\{\pi(u), \pi(v)\}, \pi, \ell).$$

is a bijection. By Claim 3.19, for any $r \geq 1$,

$$\sum_\pi |R_\pi(a,b)| = |R(a,b)| = |T(a,b)| = \sum_{i<j} |T^{i,j}(a,b)| \leq \binom{n}{2} 8(n-2)! = 4n!,$$

which implies that

$$\mathbf{E}_\pi[|R_\pi(a,b)|] \leq \frac{4n!}{n!} = 4. \qquad \square$$
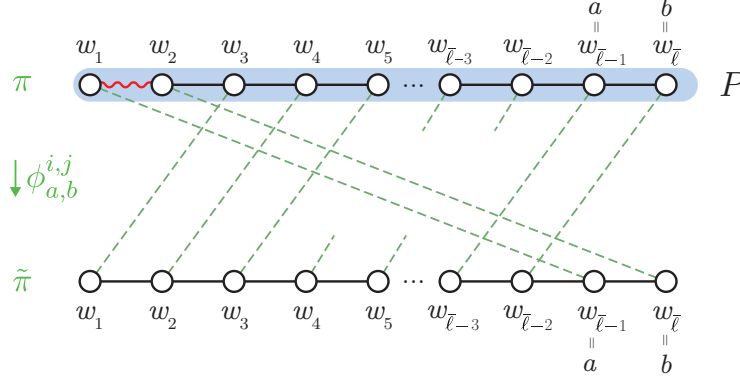
14

Figure 2: The permutation $\tilde{\pi} = \phi_{a,b}^{i,j}(\pi, \ell)$ resulting from rotating the ranks of the vertices on the prefix P of the path $P_\ell(\pi_i, \pi_j)$ under π. The prefix P is highlighted in blue. Dashed green lines connect pairs of vertices with the same rank. The ranks $\{i, j\}$ are rotated from vertices $\{w_1, w_2\}$ under π to $\{a, b\}$ under $\tilde{\pi}$. The rank of any vertex not on P is kept unchanged.

Proof of Claim 3.19. We define a map

$$\phi_{a,b}^{i,j} : T^{i,j}(a, b) \to U_{a,b}^{i,j},$$

where $U_{a,b}^{i,j}$ is a set of permutations $\tilde{\pi}$ over V defined by

$$U_{a,b}^{i,j} = \{\tilde{\pi} \mid \{\tilde{\pi}_i, \tilde{\pi}_j\} = \{a, b\}\}.$$

Note that $U_{a,b}^{i,j}$ has size $2(n-2)!$. For $(\pi, \ell) \in T^{i,j}(a, b)$, the permutation $\phi_{a,b}^{i,j}(\pi, \ell)$ is defined by rotating the ranks of the vertices along the prefix P of $P_\ell(\pi_i, \pi_j)$ ending at the directed edge (a, b). Formally, we write

$$P = (w_1, w_2, \ldots, w_{\bar{\ell}-2}, w_{\bar{\ell}-1} = a, w_{\bar{\ell}} = b),$$

where $\{w_1, w_2\} = \{\pi_i, \pi_j\}$. If (a, b) is the last edge on $P_\ell(\pi_i, \pi_j)$, then P is the same as $P_\ell(\pi_i, \pi_j)$ and $\bar{\ell} = \ell + 2$. Otherwise, (a, b) is the second-to-last edge on $P_\ell(\pi_i, \pi_j)$, in which case P equals $P_\ell(\pi_i, \pi_j)$ with the last edge removed and $\bar{\ell} = \ell + 1$. Then, we define $\tilde{\pi} = \phi_{a,b}^{i,j}(\pi, \ell)$ by

$$\tilde{\pi}(a) = \pi(w_1), \quad \tilde{\pi}(b) = \pi(w_2), \quad \tilde{\pi}(w_i) = \pi(w_{i+2}) \quad \text{for } i \in [1, \bar{\ell} - 2],$$

$$\tilde{\pi}(v) = \pi(v) \quad \text{for } v \in V \setminus \{w_1, \ldots, w_{\bar{\ell}}\}.$$

See Figure 2.

We make the following claim on the map $\phi_{a,b}^{i,j}$, which will imply that for any $\tilde{\pi} \in U_{a,b}^{i,j}$, the preimage $(\phi_{a,b}^{i,j})^{-1}(\tilde{\pi})$ has size at most a small constant:

Claim 3.20. *Suppose $\phi_{a,b}^{i,j}(\pi, \ell) = \phi_{a,b}^{i,j}(\pi', \ell')$ for two pairs $(\pi, \ell), (\pi', \ell') \in T^{i,j}(a, b)$. Let P (resp. P') be the prefix of the path $P_\ell(\pi_i, \pi_j)$ (resp. $P_{\ell'}(\pi'_i, \pi'_j)$) ending at (a, b). Then one of the following holds:*

- $P = P'$ *and* $\pi = \pi'$.

- *One of P, P' contains the other as a subpath with one fewer vertex.*

15

We defer the proof of Claim 3.20 to the end of this subsection, and first use it to finish proving Claim 3.19. We show that for any $\tilde{\pi} \in U_{a,b}^{i,j}$, the preimage $(\phi_{a,b}^{i,j})^{-1}(\tilde{\pi})$ has size at most 4. This will imply that

$$|T^{i,j}(a,b)| \le 4|U_{a,b}^{i,j}| = 8(n-2)!.$$

We write

$$(\phi_{a,b}^{i,j})^{-1}(\tilde{\pi}) = \{(\pi^1, \ell_1), \dots, (\pi^m, \ell_m)\}$$

and show that $m \le 4$. For each $k = 1, \dots, m$, let P^k denote the prefix of the path $P_{\ell_k}(\pi_i^k, \pi_j^k)$ ending at (a, b). By Claim 3.20, there are only two possibilities among the prefixes $P^1, \dots, P^m$. By permuting the indices, we assume that

$$P^1 = \dots = P^{m'} \ne P^{m'+1} = \dots = P^m$$

for some $0 \le m' \le m$. Next, we show that $m' \le 2$. For the first m' pairs in $(\phi_{a,b}^{i,j})^{-1}(\tilde{\pi})$, we have by Claim 3.20 that

$$\pi^1 = \dots = \pi^{m'}.$$

We denote this common permutation by π. Now by the definition of $T^{i,j}(a,b)$, (a, b) appears as one of the last two edges on each of $P_{\ell_1}(\pi_i, \pi_j)$, $\dots$, $P_{\ell_m}(\pi_i, \pi_j)$. This implies that there are only two possibilities among $\ell_1, \dots, \ell_{m'}$, i.e. $m' \le 2$. Similarly, we have $m - m' \le 2$. Thus, $m \le 4$. $\square$

Proof of Claim 3.20. Note that if $P = P'$, then $\phi_{a,b}^{i,j}(\pi, \ell) = \phi_{a,b}^{i,j}(\pi', \ell')$ would imply that $\pi = \pi'$. Thus in what follows, we assume that $P \ne P'$ and show that one must contain the other as a subpath with one fewer vertex.

We start by recalling some notations for π and setting up their counterparts for π'. Let P_{PIV} (resp. P'_{PIV}) denote the set of pivots found by PIVOT under the permutation π (resp. π') (Definition 3.3). For each vertex $v \in V$, let $p_v \in P_{\text{PIV}}$ (resp. $p'_v \in P'_{\text{PIV}}$) denote the pivot of v in PIVOT under π (resp. π'). (Definition 3.4).

Now, we write

$$P = (w_1, \dots, w_{\bar{\ell}-2}, w_{\bar{\ell}-1} = a, w_{\bar{\ell}} = b), \quad P' = (w'_1, \dots, w'_{\bar{\ell}'-2}, w'_{\bar{\ell}'-1} = a, w'_{\bar{\ell}'} = b).$$

Note that $\{\pi(w_1), \pi(w_2)\} = \{\pi'(w'_1), \pi'(w'_2)\} = \{i, j\}$. Let m be the largest integer such that

$$(w_{\bar{\ell}-m+1}, \dots, w_{\bar{\ell}-1} = a, w_{\bar{\ell}} = b) = (w'_{\bar{\ell}'-m+1}, \dots, w'_{\bar{\ell}'-1} = a, w'_{\bar{\ell}'} = b),$$

which is the maximal common suffix of P and P'. Then $m \ge 2$. Observe that $\phi_{a,b}^{i,j}(\pi, \ell) = \phi_{a,b}^{i,j}(\pi', \ell')$ directly implies the following properties:

Observation 3.21. *The following properties hold for the permutations π, π', paths P, P', and integer m:*

(a) $\pi(v) = \pi'(v)$ *for any vertex* $v \in V \setminus (\{w_1, \dots, w_{\bar{\ell}}\} \cup \{w'_1, \dots, w'_{\bar{\ell}'}\})$.

(b) $\pi(w_1) = \pi'(w'_1)$, $\pi(w_2) = \pi'(w'_2)$.

(c) $\pi(w_{\bar{\ell}-k}) = \pi'(w'_{\bar{\ell}'-k})$ *for any* $0 \le k \le m - 3$.

16

Now we show that one of P, P' must contain the other as a subpath with one fewer vertex. We denote

$$y = w_{\bar{\ell}-m+1} = w'_{\bar{\ell}'-m+1}, \quad z = w_{\bar{\ell}-m+2} = w'_{\bar{\ell}'-m+2}.$$

That is, y is the starting vertex of the maximal common suffix of P and P', and z is the next vertex. Then there are only two possibilities for the relation between P and P':

(A) Neither of P, P' contains the other as a subpath. We assume that $\pi(z) < \pi'(z)$.

(B) One of P, P' contains the other as a subpath. We assume that P contains P'. Then $m = \bar{\ell}' < \bar{\ell}$.

See Figure 3. We show that Case (A) is impossible, and that in Case (B), we must have $\bar{\ell} - \bar{\ell}' = 1$.
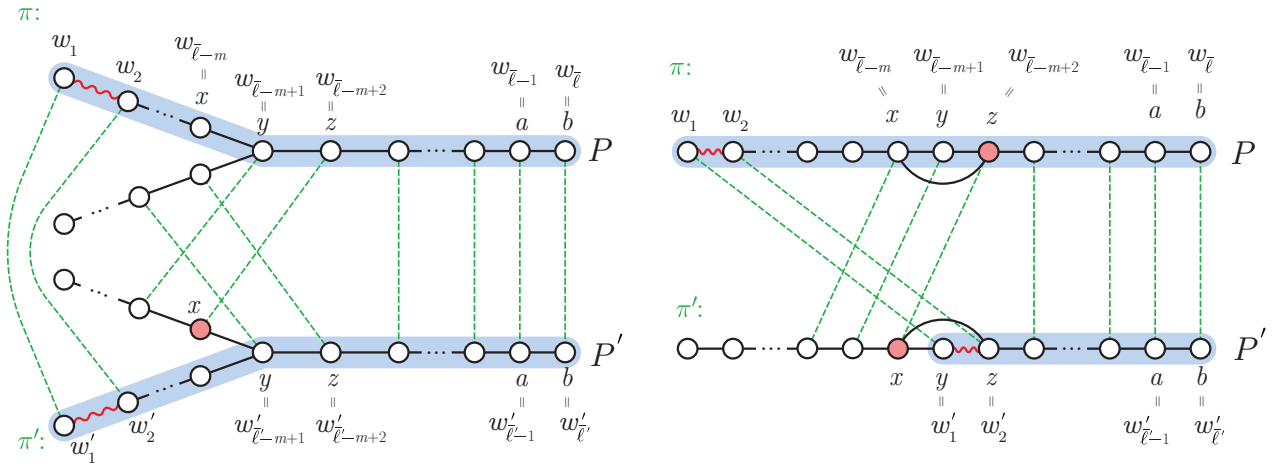

Figure 3: Cases (A) and (B). In each case, the paths P and P' under permutations π and π' respectively are highlighted in blue. Dashed green lines connect pairs of vertices with the same rank. In both cases, the vertex $x = w_{\bar{\ell}-m}$ highlighted in red is the pivot of $y = w'_{\bar{\ell}'-m+1}$ in PIVOT under π'. In Case (B), where $m = \bar{\ell}'$, x is the pivot of both $y = w'_1$ and $z = w'_2$ in PIVOT under π', and z is a pivot in PIVOT under π, also highlighted in red.

We note that $\pi(z) < \pi'(z)$ in Case (B) as well, since $\{y, z\} = \{\pi'_i, \pi'_j\}$ in this case, which implies that $\pi'(z) \geq i > \pi(z)$ (recall $i < j$). As a consequence of Observation 3.21 (a) (c), we have

$$\pi_k = \pi'_k, \quad \text{for all } k < \pi(z). \tag{2}$$

This implies that, if $v \in V$ is a vertex such that $\pi(v) < \pi(z)$ or $\pi'(v) < \pi(z)$, then $\pi(v) = \pi'(v)$. Moreover, Vertex(v) makes the same *recursive* calls to Vertex under π and π'. In particular, v is a pivot in PIVOT under π (i.e. $v \in P_{\text{PIV}}$) if and only if v is a pivot under π' (i.e. $v \in P_{\text{PIV}}'$). The following observation directly follows:

Observation 3.22. *Let $v \in V$ be a vertex with pivot p_v in PIVOT under π and p'_v under π'. If $\pi(p_v) < \pi(z)$ or $\pi'(p'_v) < \pi(z)$, then $p_v = p'_v$ and $\pi(p_v) = \pi'(p'_v)$.*

We set $x = w_{\bar{\ell}-m}$ in both cases. Then

$$\pi'(x) = \pi(z).$$

Note by Observation 3.14 that y directly queries z under π. Moreover, x directly queries y under π unless $\{x, y\} = \{\pi_i, \pi_j\}$, in which case $\{x, y\}$ directly queries z.

We first claim that in both cases, x is a pivot in PIVOT under π', i.e. $x \in P_{\text{PIV}}'$. Otherwise, $x \neq p_x'$ or equivalently $\pi'(p_x') < \pi'(x) = \pi(z)$. Then Observation 3.22 implies that $p_x = p_x'$ and $\pi(p_x) = \pi'(p_x')$. Thus, $\pi(p_x) = \pi'(p_x') < \pi(z) < \pi(y)$. From Observation 3.10 (b) (d), this contradicts that x directly queries y under π in the case $\{x, y\} \neq \{\pi_i, \pi_j\}$, or $\{x, y\}$ directly queries z in the case $\{x, y\} = \{\pi_i, \pi_j\}$.

Next, we claim that in both cases, x is the pivot of y in PIVOT under π', i.e. $x = p_y'$. Otherwise, we have $\pi'(p_y') < \pi'(x) = \pi(z)$. By Observation 3.22, $p_y = p_y'$ and $\pi(p_y) = \pi'(p_y')$. Thus, $\pi(p_y) = \pi'(p_y') < \pi(z)$. From Observation 3.10 (b), this contradicts that y directly queries z under π.

At this point, we can already show that Case (A) is impossible, as follows. In this case, by Observation 3.14, y directly queries z under π'. However, this contradicts that $\pi'(p_y') = \pi'(x) = \pi(z) < \pi'(z)$, again from Observation 3.10 (b).

It remains to complete the proof in Case (B). In this case, since $\{y, z\} = \{\pi_i', \pi_j'\}$ is a pair in X under the permutation π' and $x = p_y'$, we also have $x = p_z'$. In particular, $\{x, z\} \in E$.

We now claim that z is a pivot in PIVOT under π, i.e. $z \in P_{\text{PIV}}$. Otherwise, $z \neq p_z$ or equivalently $\pi(p_z) < \pi(z)$. Then Observation 3.22 implies that $p_z' = p_z$ and $\pi'(p_z') = \pi(p_z)$. But now, $\pi'(p_z') = \pi(p_z) < \pi(z) = \pi'(x)$. This contradicts that x is the pivot in PIVOT that z joins under π'.

As a consequence, since y directly queries z under π, we have by Observation 3.10 (b) that z must be the pivot of y in PIVOT under π, i.e. $z = p_y$. Moreover, since z is a also a neighbor of x, we have $\pi(p_x) \leq \pi(z)$. Thus $\pi(y) > \pi(z) \geq \pi(p_x)$. By Observation 3.10 (a) (b), x cannot directly query y under π. Then we must have $\{x, y\} = \{\pi_i, \pi_j\}$, i.e. $\{x, y\}$ is the first edge on P (or $P_\ell(\pi_i, \pi_j)$). This then implies that $\bar{\ell} = \bar{\ell}' + 1$, as desired. $\qquad \square$

3.6.2 Proof of Lemma 3.18

Throughout this subsection, we fix distinct vertices $a, b \in V$ such that $\{a, b\} \notin E$. As in Section 3.6.1, given a permutation π over V and $i \in [n]$, we denote by $\pi_i \in V$ the vertex with rank i under π.

Similar to the proof of Lemma 3.17, we break down the charges to bad triangles involving both a and b by specifying the ranks of the pair of vertices in X that initiates the charge. Given $i, j \in [n]$ with $i < j$, we define $T^{i,j}(a, b)$ to be the set of pairs (π, ℓ) of a permutation π and an integer $\ell \in [2, 2r]$ such that under π, $\{\pi_i, \pi_j\} \in X$ and the last three vertices in the ordered list $P_\ell(\pi_i, \pi_j)$ are (a, c, b) for some $c \in V$. We will prove the following bound on the size of $T^{i,j}(a, b)$, which is analogous to Claim 3.19:

Claim 3.23. *For any $i, j \in [n]$ with $i < j$ and any $r \geq 1$, we have*

$$|T^{i,j}(a, b)| \leq 4(n - 2)!.$$

Claim 3.23 implies Lemma 3.18 in the same way as how Claim 3.19 implies Lemma 3.17 (see Section 3.6.1), and we omit the proof here.

Proof of Claim 3.23. We define a map

$$\phi_{a,b}^{i,j} : T^{i,j}(a, b) \to U_{a,b}^{i,j},$$

18

where
$$U_{a,b}^{i,j} = \{\tilde{\pi} \mid \{\tilde{\pi}_i, \tilde{\pi}_j\} = \{a, b\}\}$$

as in the proof of Claim 3.19. For $(\pi, \ell) \in T^{i,j}(a,b)$, the permutation $\phi_{a,b}^{i,j}(\pi, \ell)$ is defined by rotating the ranks of the vertices along the path $P_\ell(\pi_i, \pi_j)$ while skipping the vertex between a and b. Formally, we write

$$P_\ell(\pi_i, \pi_j) = (w_1, w_2, \ldots, w_\ell = a, w_{\ell+1}, w_{\ell+2} = b),$$

where $\{w_1, w_2\} = \{\pi_i, \pi_j\}$. Then, we define $\tilde{\pi} = \phi_{a,b}^{i,j}(\pi, \ell)$ by

$$\tilde{\pi}(a) = \pi(w_1), \quad \tilde{\pi}(b) = \pi(w_2), \quad \tilde{\pi}(w_{\ell+1}) = \pi(w_{\ell+1}),$$

$$\tilde{\pi}(w_i) = \pi(w_{i+2}) \quad \text{for } i \in [1, \ell - 2], \quad \tilde{\pi}(w_{\ell-1}) = \pi(w_{\ell+2}),$$

$$\tilde{\pi}(v) = \pi(v) \quad \text{for } v \in V \setminus \{w_1, \ldots, w_{\ell+2}\}.$$
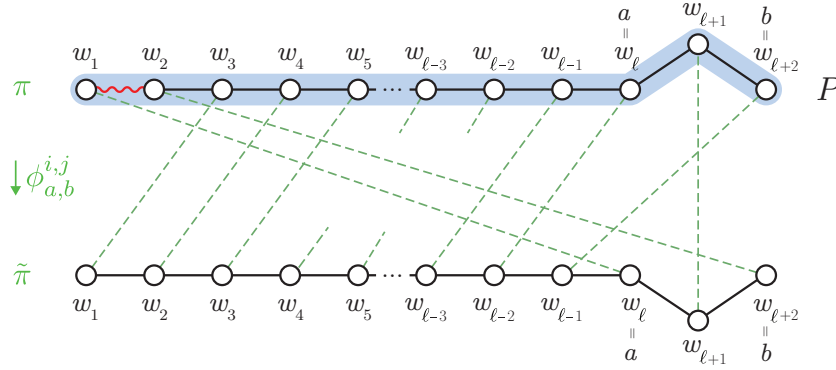
See Figure 4.

Figure 4: The permutation $\tilde{\pi} = \phi_{a,b}^{i,j}(\pi, \ell)$ resulting from rotating the ranks of the vertices on the path $P_\ell(\pi_i, \pi_j)$ under π. The path $P_\ell(\pi_i, \pi_j)$ is highlighted in blue. Dashed green lines connect pairs of vertices with the same rank. The ranks $\{i, j\}$ are rotated from vertices $\{w_1, w_2\}$ under π to $\{a, b\}$ under $\tilde{\pi}$. The rotation skips the vertex $w_{\ell+1}$ between a and b and leaves its rank unchanged. The rank of any vertex not on $P_\ell(\pi_i, \pi_j)$ is kept unchanged as well.

We make the following claim on the map $\phi_{a,b}^{i,j}$, which directly implies that for any $\tilde{\pi} \in U_{a,b}^{i,j}$, the preimage $(\phi_{a,b}^{i,j})^{-1}(\tilde{\pi})$ has size at most 2:

Claim 3.24. *Suppose $\phi_{a,b}^{i,j}(\pi, \ell) = \phi_{a,b}^{i,j}(\pi', \ell')$ for two distinct pairs $(\pi, \ell), (\pi', \ell') \in T^{i,j}(a,b)$. Then one of $P_\ell(\pi_i, \pi_j)$, $P_{\ell'}(\pi'_i, \pi'_j)$ contains the other as a subpath with one fewer vertex.*

It follows from Claim 3.24 that

$$|T^{i,j}(a,b)| \leq 2|U_{a,b}^{i,j}| = 4(n-2)!.$$

We prove Claim 3.24 below. $\qquad\square$

Proof of Claim 3.24. Note that $P_\ell(\pi_i, \pi_j) = P_{\ell'}(\pi'_i, \pi'_j)$ would imply that $\ell = \ell'$, and additionally that $\pi = \pi'$ since $\phi_{a,b}^{i,j}(\pi, \ell) = \phi_{a,b}^{i,j}(\pi', \ell')$. Thus we must have $P_\ell(\pi_i, \pi_j) \neq P_{\ell'}(\pi'_i, \pi'_j)$.

We use the same notations for π and π' as in the proof of Claim 3.20. We write

$$P = P_\ell(\pi_i, \pi_j) = (w_1, \ldots, w_\ell = a, w_{\ell+1} = c, w_{\ell+2} = b),$$

$$P' = P_{\ell'}(\pi'_i, \pi'_j) = (w'_1, \ldots, w'_{\ell'} = a, w'_{\ell'+1} = c', w'_{\ell'+2} = b).$$

Note that $\{\pi(w_1), \pi(w_2)\} = \{\pi'(w'_1), \pi'(w'_2)\} = \{i, j\}$. We consider three cases separately:

(I) $c = c'$ and $w_{\ell-1} = w'_{\ell'-1}$.

(II) $c = c'$ and $w_{\ell-1} \neq w'_{\ell'-1}$.

(III) $c \neq c'$.

In Cases (I) or (II), the two paths P and P' end at the same three vertices (a, c, b). Let m be the largest integer such that

$$(w_{\ell-m+3}, \ldots, w_\ell = a, w_{\ell+1} = c, w_{\ell+2} = b) = (w'_{\ell'-m+3}, \ldots, w'_{\ell'} = a, w'_{\ell'+1} = c, w'_{\ell'+2} = b),$$

which is the maximal common suffix of P and P'. Then $m \geq 3$, and equality holds if and only if we are in Case (II).

We first consider Case (I), where $m \geq 4$. Then, after setting $\bar{\ell} = \ell + 2, \bar{\ell}' = \ell' + 2$, we see that $\phi_{a,b}^{i,j}(\pi, \ell) = \phi_{a,b}^{i,j}(\pi', \ell')$ directly implies that the properties in Observation 3.21 hold for the permutations π, π', paths P, P', and integer m. Therefore, we may finish Case (I) by the same paragraphs after Observation 3.21 up to the end of the proof of Claim 3.20.
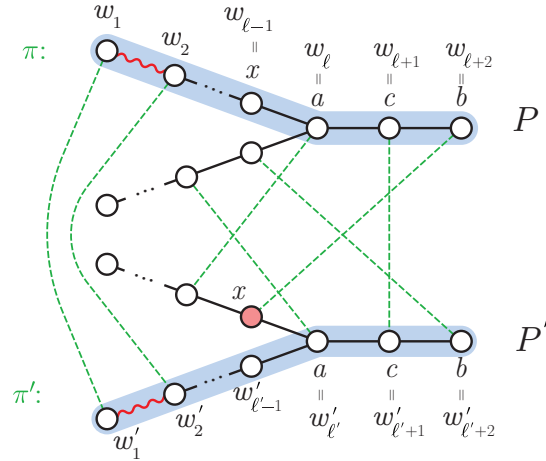

Figure 5: Case (II) where $m = 3$. The paths P and P' under permutations π and π' respectively are highlighted in blue. Dashed green lines connect pairs of vertices with the same rank. The vertex $x = w_{\ell-1}$ highlighted in red is the pivot of a in PIVOT under π'.

Next, we show that Case (II), where $m = 3$, is impossible. See Figure 5. In this case, $\pi(c) = \pi'(c)$ but $\pi(b) \neq \pi'(b)$, and we assume without loss of generality that $\pi(b) < \pi'(b)$. From $\phi_{a,b}^{i,j}(\pi, \ell) = \phi_{a,b}^{i,j}(\pi', \ell')$, we directly have

$$\pi_k = \pi'_k, \quad \text{for all } k < \pi(b),$$

which is similar to Eq (2). This implies the following observation, similar to Observation 3.22:

20

Observation 3.25. *Let $v \in V$ be a vertex with pivot p_v in PIVOT under π and p'_v under π'. If $\pi(p_v) < \pi(b)$ or $\pi'(p'_v) < \pi(b)$, then $p_v = p'_v$ and $\pi(p_v) = \pi'(p'_v)$.*

We set $x = w_{\ell-1}$. Then

$$\pi'(x) = \pi(b).$$

Based on Observation 3.25, we can use the same argument as that after Observation 3.22 in the proof of Claim 3.20 to deduce that under π', x is the pivot of a in PIVOT, i.e. $x = p'_a \in P'_{\text{PIV}}$. However, since $\pi'(c) = \pi(c) > \pi(b) = \pi'(x)$, a cannot directly query c' under π' by Observation 3.10 (b). This is a contradiction to Observation 3.14.

Finally, we show that Case (III) is impossible as well. In this case, $\pi(c) \neq \pi'(c')$, and we assume without loss of generality that $\pi(c) < \pi'(c')$. We will consider two subcases depending on whether c is on the path P' separately. See Figure 6.
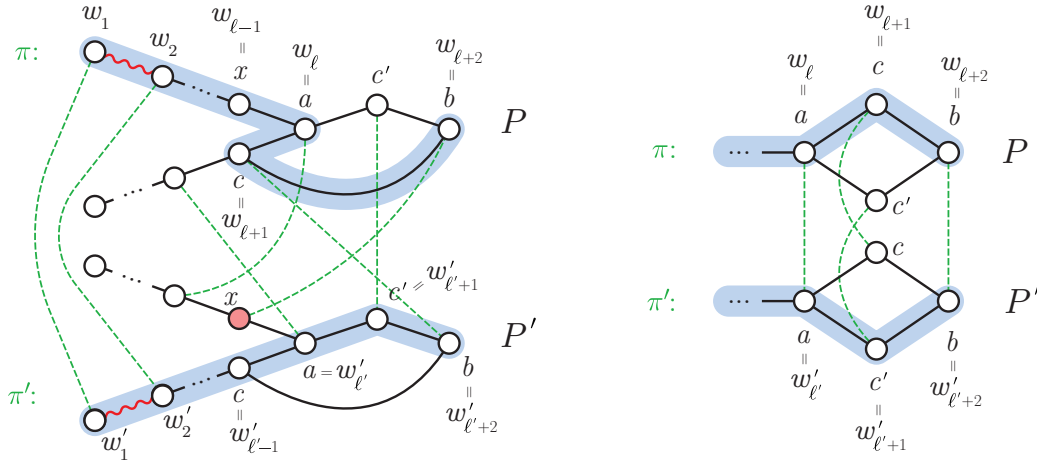

Figure 6: The two subcases of Case (III) where $c \neq c'$. In each subcase, the paths P and P' under permutations π and π' respectively are highlighted in blue. Dashed green lines connect pairs of vertices with the same rank. In the subcase where c is on P', the vertex $x = w_{\ell-1}$ highlighted in red is the pivot of a in PIVOT under π'.

First suppose that c is on P'. By the definition of $\phi^{i,j}_{a,b}$, $\tilde{\pi}(c) = \pi'(v)$ for some vertex v on P'. Now recall that $\tilde{\pi}(c) = \pi(c) < \pi'(c')$, which implies that $\pi'(v) < \pi'(c')$. However, the only vertex on P' with rank less than $\pi'(c')$ under π' is b, which means that $v = b$. Since $\pi'(b) = \tilde{\pi}(w'_{\ell'-1})$, we have $c = w'_{\ell'-1}$. This implies that

$$\pi(b) < \pi(c) = \tilde{\pi}(c) = \pi'(b) < \pi'(c').$$

The proof from this point on is similar to Case (II) above. Here, $\phi^{i,j}_{a,b}(\pi, \ell) = \phi^{i,j}_{a,b}(\pi', \ell')$ also implies that

$$\pi_k = \pi'_k, \quad \text{for all } k < \pi(b).$$

Thus Observation 3.25 also holds. We again set $x = w_{\ell-1}$. Then

$$\pi'(x) = \pi(b).$$

In particular, $\pi'(x) = \pi(b) < \pi'(c')$ implies that $x \neq c'$. Based on Observation 3.25, we can use the same argument as that after Observation 3.22 in the proof of Claim 3.20 to deduce that $x = p'_a \in$

21

P'_{PIV}. However, since $\pi'(c') > \pi'(x)$, a cannot directly query c' under π' by Observation 3.10 (b). This is a contradiction to Observation 3.14.

It remains to rule out the subcase where c is not on P'. Here, we have

$$\pi'(c) = \tilde{\pi}(c) = \pi(c) < \pi'(c').$$

By Observation 3.14, under π', a directly queries c'. Thus by Observation 3.10 (a) (b), a also directly queries c under π' since it is a neighbor with rank smaller than c'. Now, if there exists a vertex $d \in V$ that c directly queries under π', then at some point, the stack of recursive calls to $\texttt{Vertex}$ when we call $\texttt{Pair}(w'_1, w'_2)$ consists of the following ℓ' elements

$$(w'_3, \ldots, w'_{\ell'} = a, c, d),$$

and this happens before when the stack consists of $(w'_3, \ldots, w'_{\ell'} = a, c', b)$. This contradicts the definition of $S_{\ell'}(w'_1, w'_2)$ and $P_{\ell'}(w'_1, w'_2)$. Otherwise, c does not directly query any vertex under π', which means that c is a pivot in PIVOT, i.e. $c \in P'_{\text{PIV}}$. But this contradicts that a directly queries the neighbor c' whose rank is greater than that of c under π', by Observation 3.10 (b). $\qquad\square$

3.7 Deferred Proofs

To show Claim 3.11, we will make use of the following claim on the stack of recursive calls to $\texttt{Vertex}$ when we call $\texttt{Vertex}(w)$ for a vertex $w \in V$:

Claim 3.26. *Let w be a vertex that remains unsettled after t rounds in r-PIVOT. Then when we call $\texttt{Vertex}(w)$, at some point the stack of recursive calls to $\texttt{Vertex}$ includes at least $2t$ elements (excluding w).*

Proof. We proceed by induction on t. The base case $t = 0$ clearly holds. Now assume the statement for $t = k$, and let w be a vertex that remains unsettled after $k + 1$ rounds. We consider two cases depending on whether w is a pivot in P_{PIV} separately:

We first consider the case where $w \in P_{\text{PIV}}$ is a pivot. By Observation 3.10 (a), the set of vertices that w directly queries is
$$Q_w = \{z \in N(w) \mid \pi(z) < \pi(w)\},$$
and that Q_w does not contain any pivots in P_{PIV}. Without loss of generality, we may assume that there does not exist a vertex $z \in Q_w$ that remains unsettled after $k + 1$ rounds, since otherwise to prove the claim for w it suffices to prove it for z.

Next, we claim that there exists a vertex $z \in Q_w$ that remains unsettled after k rounds. Otherwise, since all neighbors of w with smaller rank under π are settled after k rounds, w would be marked as a pivot and marked as settled in round $k + 1$, a contradiction.

As a consequence, p_z must also remain unsettled after k rounds. Since $z \notin P_{\text{PIV}}$, we have $p_z \neq z$. Thus by Observation 3.10 (b), z directly queries p_z. By the inductive hypothesis applied to p_z, when we call $\texttt{Vertex}(p_z)$, at some point the stack of recursive calls to $\texttt{Vertex}$ includes $2k$ elements (excluding p_z). This implies the claim for w.

Now, we consider the other case where $w \notin P_{\text{PIV}}$ is non-pivot. By Observation 3.10 (b), w directly queries p_w. We claim that p_w also remains unsettled after $k + 1$ rounds. Otherwise p_w would have been marked as a pivot by the end of round $k + 1$ and w would have been removed together with p_w, a contradiction. By the previous case, the claim holds for p_w, which then implies the claim for w. $\qquad\square$

Proof of Claim 3.11 via Claim 3.26. It suffices to prove the case $\ell = 2r$. We consider the two cases for the pair $\{u, v\}$ given in Lemma 3.8 separately. In Case (i), $p_{\{u,v\}}$ remains unsettled after r rounds in r-PIVOT. If $p_{\{u,v\}} \notin \{u, v\}$, then by Observation 3.10 (d), $\{u, v\}$ directly queries $p_{\{u,v\}}$, and we may thus conclude by Claim 3.26 applied to $w = p_{\{u,v\}}$. If $p_{\{u,v\}} \in \{u, v\}$, then we may see from Observation 3.10 (a) (c) that every vertex z directly queried by $p_{\{u,v\}}$ is also directly queried by $\{u, v\}$. We again conclude by Claim 3.26 applied to $w = p_{\{u,v\}}$.

In Case (ii), $w_{\{u,v\}} \in N(u) \cup N(v)$ is an unsettled vertex after r rounds in r-PIVOT with $\pi(w_{\{u,v\}}) < \pi(p_{\{u,v\}})$. Thus $\{u, v\}$ directly queries $w_{\{u,v\}}$ by Observation 3.10 (c) (d). We may then conclude by Claim 3.26 applied to $w = w_{\{u,v\}}$. $\square$

4 Implementations

In this section, we prove the implications of Theorem 1.1 in the models discussed. In each model, given $\varepsilon > 0$, we take $r = O(1/\varepsilon)$ and provide an implementation of our r-PIVOT algorithm. We introduce the following notation: For each $t \in [1, r]$, and each vertex $v \in V$ that is unsettled at the beginning of round t or r-PIVOT, let $\eta_t(v)$ denote the vertex that has the smallest rank under π among the vertices in $\{v\} \cup N(v)$ that are unsettled at the beginning of round t. Note that v is marked as a pivot in round t if and only if $\eta_t(v) = v$.

The Massively Parallel Computations (MPC) Model: In the MPC model, the edge set E of the input graph G is distributed to a collection of machines. Computation then proceeds in synchronous rounds. In each round, a machine can receive messages from other machines in the previous round, perform some local computation, and send messages to other machines as input for the next round. Each message has size $O(1)$ words. Each machine has limited local space, which restricts the total number of messages it can receive or send in a round. For correlation clustering, at the end of the computation, each machine is required to know the cluster IDs of the vertices for which it initially holds edges. We focus on the *strictly sublinear regime* of MPC where the computation uses $O(n^\delta)$ space per machine, where $\delta > 0$ is a constant that can be made arbitrarily small, and $O(m)$ total space.

We now show that there is a randomized $O(r)$-round MPC algorithm that obtains an expected $\left(3 + \frac{8}{2r-1} + n^{-\Omega(1)}\right)$-approximation of correlation clustering and uses $O(n^\delta)$ space per machine, where constant $\delta > 0$ can be made arbitrarily small, and $O(m)$ total space.

Proof of Corollary 1.3. Since a random permutation of V cannot be drawn and stored on a single machine, we implement the following variant of r-PIVOT in the MPC model:

Algorithm r-PIVOT-VARIANT : A variant of the algorithm r-PIVOT.

- Instead of drawing a random permutation π of the vertex set V, each vertex $v \in V$ draws a rank $\pi(v)$ from $\{0, \ldots, n^c - 1\}$ independently and uniformly at random, where c is a sufficiently large constant.

- Proceed in the same way as in r-PIVOT, breaking ties in ranks in favor of the vertex with smaller ID.

Claim 4.1. *For any $r \geq 1$, the algorithm r-PIVOT-VARIANT outputs an expected $\left(3 + \frac{8}{2r-1} + n^{-\Omega(1)}\right)$-approximation of correlation clustering.*

Proof. First, observe that any connected component of G that is a clique can be successfully identified by both r-PIVOT and r-PIVOT-VARIANT with probability 1. Thus, if all connected components of G are cliques, then both r-PIVOT and r-PIVOT-VARIANT output the optimal solution with probability 1.

Now consider the case where at least one connected component of G is not a clique. Then $\text{OPT}(G) \geq 1$. With probability at least $1 - n^{2-c}$, the ranks $\pi(v)$ drawn in r-PIVOT-VARIANT are all distinct, in which case π is equivalent to a uniformly drawn permutation of V, and the output of r-PIVOT-VARIANT is the same as the output of r-PIVOT, which has cost at most $\left(3 + \frac{8}{2r-1}\right)\cdot\text{OPT}(G)$ by Corollary 1.2. In the case there are repeated ranks, which happens with probability at most n^{2-c}, we simply upper bound the cost by $\binom{n}{2} \leq n^2 \cdot \text{OPT}(G)$, since $\text{OPT}(G) \geq 1$. Thus the expected cost of the above variant is at most

$$\left(3 + \frac{8}{2r-1} + n^{2-c} \cdot n^2\right)\text{OPT}(G) \leq \left(3 + \frac{8}{2r-1} + n^{-\Omega(1)}\right)\text{OPT}(G). \quad \square$$

Now we describe the MPC implementation. Take $r = O(1/\varepsilon)$ such that $\frac{8}{2r-1} + n^{-\Omega(1)} < \varepsilon$, and fix $\delta > 0$. We use a collection of $O(n^{1-\delta})$ machines to draw the rank $\pi(v)$ of each vertex $v \in V$ from $\{0, \ldots, n^c - 1\}$ independently and uniformly at random, where c is a sufficiently large constant. Let M_v denote the machine that holds the rank $\pi(v)$ of a vertex v. We will also let M_v keep track of whether v is settled or a pivot throughout the computation. All vertices are initially marked as unsettled and non-pivot.

Moreover, the input edges in E are stored in a collection of $O(m/n^\delta)$ machines. In the implementation, we will need to perform the following two operations:

- Every machine M_v informs each machine that holds an input edge incident to v whether v is settled or a pivot.

- Every machine that holds input edges requests to mark some of the vertices it holds edges to as settled, by communicating with the corresponding machines M_v's.

We note that each operation can be done in $O(1/\delta)$ rounds of MPC with $O(n^\delta)$ space per machine and $O(m)$ total space.

For each $t \in [1, r]$, we implement round t of r-PIVOT by $O(1/\delta)$ rounds of MPC as follows: First, we compute $\eta_t(v)$ for all unsettled vertices $v \in V$ and store them in the respective machines M_v's. To do this, we construct a set L_t of at most $2m + n$ *ordered* pairs of vertices as follows:

- For each edge $\{u, v\} \in E$ such that both u, v are unsettled, add both (u, v) and (v, u) to L_t. This is done by the machine holding the input edge $\{u, v\}$, which communicates with M_u and M_v to check whether u, v are both unsettled.

- For each unsettled vertex $v \in V$, add (v, v) to L_t. This is done by the machine M_v.

Next, we sort the pairs in L_t by the rank of the first vertex, and in case of ties, by the rank of the second vertex. This can be done in $O(1/\delta)$ rounds of MPC with $O(n^\delta)$ space per machine and $O(m)$ total space [20]. The sorted list L_t has form

$$(u_1, v_{1,1}), \ldots, (u_1, v_{1,\deg_t(u_1)+1}), (u_2, v_{2,1}), \ldots, (u_2, v_{2,\deg_t(u_2)+1}), \ldots, (u_{n_t}, v_{n_t,1}), \ldots, (u_{n_t}, v_{n_t,\deg_t(u_{n_t})+1}),$$

where $u_1, \ldots, u_{n_t}$ is a listing of the unsettled vertices in increasing order of their ranks, and for each i, $v_{i,1}, \ldots, v_{i,\deg_t(u_i)+1}$ is a listing of the unsettled vertices in $\{u_i\} \cup N(u_i)$ in increasing order of their ranks. In particular, for each i, $v_{i,1} = \eta_t(u_i)$. Then, any machine that holds $(u_i, v_{i,1})$ sends $v_{i,1} = \eta_t(u_i)$ to M_{u_i}.

Now, for any unsettled vertex v, if M_v sees that $\eta_t(v) = v$, it marks v as settled and a pivot. Then, any machine holding an input edge $\{u, v\} \in E$ checks whether u is a newly-identified pivot and v is unsettled, in which case it asks M_v to mark v as settled. Thereby we have implemented round t of r-PIVOT by $O(1/\delta)$ rounds of MPC.

Now that we have implemented all r rounds of r-PIVOT by $O(r/\delta)$ rounds of MPC, in $O(1/\delta)$ additional rounds, each M_v determines the cluster ID of v, as follows: Every pivot starts a cluster which includes itself. Then, any non-pivot vertex u can determine the vertex v that has the smallest rank under π among all pivots in $N(u)$ (if any), as well as the vertex w that has the smallest rank under π among all unsettled vertices in $N(u)$ (if any). The values v and w can be computed and sent to M_u in a similar way as how the values $\eta_t(v)$'s are computed and sent above. If v doesn't exist, or both v, w exist and $\pi(w) < \pi(v)$, u forms a singleton cluster. Otherwise, u joins the cluster of v. In the end, the cluster IDs are broadcast to the machines which initially hold input edges. $\square$

The Graph Streaming Model: In the streaming correlation clustering problem, the edges of graph G arrive one by one in a stream, and in an arbitrary order. The algorithm has to take few passes over the input, use a small space, and output the clusters at the end. It is not hard to see that $\Omega(n)$ words of space are needed just to store the final output.

We now show that there is a randomized $(2r + 1)$-pass streaming algorithm using $O(n \log n)$ bits of space that obtains an expected $\left(3 + \frac{8}{2r-1}\right)$-approximation of correlation clustering.

Proof of Corollary 1.4. We take $r = O(1/\varepsilon)$ such that $\frac{8}{2r-1} < \varepsilon$ and implement r-PIVOT in the streaming model. We start by drawing a random permutation π of V and marking all vertices as unsettled and non-pivot. Then, for each $t \in [1, r]$, we implement round t of r-PIVOT by making two passes of the stream. In the first pass, we maintain $\eta_t(v)$ associated to each unsettled vertex $v \in V$, which is initialized to v, as follows: whenever an edge $\{u, v\} \in E$ arrives with both u, v unsettled, if $\pi(u) < \pi(\eta_t(v))$, we update $\eta_t(v)$ to be u; similarly, if $\pi(v) < \pi(\eta_t(u))$, we update $\eta_t(u)$ to be v. After this pass, any unsettled vertex v with $\eta_t(v) = v$ is marked as settled and a pivot. In the second pass, we mark all neighbors of the newly-identified pivots as settled, as follows: whenever an edge $\{u, v\} \in E$ arrives with one of u, v marked as a pivot and the other unsettled, we mark the unsettled vertex as settled.

Now that we have implemented all r rounds of r-PIVOT by $2r$ passes of the stream, we determine the output clustering by making an additional final pass. Every pivot starts a cluster which includes itself. During the final pass, for each non-pivot vertex u, we find the vertex v that has the smallest rank under π among all pivots in $N(u)$ (if any), as well as the vertex w that has the smallest rank under π among all unsettled vertices in $N(u)$ (if any). If v doesn't exist, or both v, w exist and $\pi(w) < \pi(v)$, u forms a singleton cluster. Otherwise, u joins the cluster of v. $\square$

The Local Model: In the local model, each vertex of the input graph G hosts a computationally unbounded processor, computation proceeds in rounds, and adjacent vertices can exchange messages of any size in each round. The main question is the number of rounds it takes to solve a graph problem where the input is the same as the communication network G.

We now show that there is a randomized $(2r+1)$-round local algorithm that obtains an expected $\left(3 + \frac{8}{2r-1} + n^{-\Omega(1)}\right)$-approximation of correlation clustering using $O(\log n)$-bit messages.

Proof of Corollary 1.5. Take $r = O(1/\varepsilon)$ such that $\frac{8}{2r-1} + n^{-\Omega(1)} < \varepsilon$. Since we cannot globally draw a random permutation of V in the local model, we implement r-PIVOT-VARIANT given in the MPC implementation (see the proof of Corollary 1.3 above). To start, each vertex $v \in V$ draws its rank $\pi(v)$ from $\{0, \dots, n^c - 1\}$ independently and uniformly at random, where c is a sufficiently large constant, and gets marked as unsettled and non-pivot. Then, for each $t \in [1, r]$, we implement round t of r-PIVOT by two rounds in the local model. In the first round, any unsettled vertex sends its rank to all of its neighbors. Then, any unsettled vertex v can determine $\eta_t(v)$ from the ranks of all of its unsettled neighbors, and if $\eta_t(v) = v$, it gets marked as settled and a pivot. In the second round, any newly-identified pivot informs all of its neighbors. Then, any unsettled vertex that sees a newly-identified pivot neighbor gets marked as settled.

Now that we have implemented all r rounds of r-PIVOT by $2r$ rounds in the local model, we determine the output clustering by an additional final round. Every pivot starts a cluster which includes itself. In the final round, any pivot informs all of its neighbors that it is pivot and sends over its rank under π. Moreover, any unsettled vertex informs all of its neighbors that it is unsettled and sends over its rank under π. Then, any non-pivot vertex u can determine the vertex v that has the smallest rank under π among all pivots in $N(u)$ (if any), as well as the vertex w that has the smallest rank under π among all unsettled vertices in $N(u)$ (if any). If v doesn't exist, or both v, w exist and $\pi(w) < \pi(v)$, u forms a singleton cluster. Otherwise, u joins the cluster of v. $\qquad\square$

Local Computation Algorithms: Centralized Local Computation Algorithms (LCAs) are useful for problems where both the input and output are too large. An LCA is not required to output the whole solution, but instead should answer queries about parts of the output. For example, for the correlation clustering problem, the query to an LCA is a vertex v and the answer should return the cluster ID of v. For graph problems, it is common to assume that the algorithm has query access to the adjacency lists. That is, for any vertex v, the LCA can query the degree of v in G, and for any $i \in [\deg(v)]$, the LCA can request the ID of i-th neighbor of v. The goal is to answer each query using small time and space.

As it is by now standard and because r-PIVOT can be implemented in $O(r)$ rounds in the local model, we can obtain a $\Delta^{O(r)} \operatorname{poly} \log n$ time/space LCA implementation for r-PIVOT by simply collecting the whole $O(r)$-hop of any vertex [25, 6]. This shows Corollary 1.6, with a choice of $r = O(1/\varepsilon)$ such that $\frac{8}{2r-1} + n^{-\Omega(1)} < \varepsilon$.

5   Conclusion & Open Problems

In this work, we showed that a $(3 + \varepsilon)$-approximation of correlation clustering can be obtained in $O(1/\varepsilon)$ rounds in models such as (strictly sublinear) MPC, local, and streaming. This is a culminating point for low-depth algorithms for correlation clustering as the approximation gets close to a barrier of 3 for combinatorial algorithms and the round-complexity is essentially constant.

Several interesting questions remain open, especially in big data settings where there is no direct notion of round complexity. For example, is it possible to obtain an (almost) 3-approximation of correlation clustering in:

- Sublinear time? See [7] for the formal model for correlation clustering.

- $O(1)$ update-time in the fully dynamic model? See [10] for a poly log n update-time algorithm.
- $O(\Delta)$ time of the LCA model?
- A single pass of the streaming setting using $O(n \operatorname{poly} \log(n))$ space?

Finally, recall from the last paragraph of Section 1.2 that our algorithm combined with the algorithm of [15] gives an efficient $O(1/\varepsilon)$-round algorithm for rounding the natural LP solution up to a factor of $(2.06 + \varepsilon)$. Unfortunately, solving the LP remains the main bottleneck. It would thus be extremely interesting to study whether a low-round algorithm exists for solving the natural correlation clustering LP (see [15]).

References

[1] Rakesh Agrawal, Alan Halverson, Krishnaram Kenthapadi, Nina Mishra, and Panayiotis Tsaparas. Generating labels from clicks. In *Proceedings of the Second International Conference on Web Search and Web Data Mining, WSDM 2009, Barcelona, Spain, February 9-11, 2009*, pages 172–181, 2009.

[2] Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation clustering in data streams. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2237–2246, 2015.

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

[4] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: ranking and clustering. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 684–693. ACM, 2005.

[5] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):23:1–23:27, 2008.

[6] Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1132–1139. SIAM, 2012.

[7] Sepehr Assadi and Chen Wang. Sublinear time and space algorithms for correlation clustering via sparse-dense decompositions. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, to appear*.

[8] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, page 238, 2002.

[9] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Mach. Learn.*, 56 (1-3):89–113, 2004.

[10] Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Cliff Stein, and Madhu Sudan. Fully dynamic maximal independent set with polylogarithmic update time. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 382–405, 2019.

[11] Guy E. Blelloch, Jeremy T. Fineman, and Julian Shun. Greedy sequential maximal independent set and matching are parallel on average. In *24th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '12, Pittsburgh, PA, USA, June 25-27, 2012*, pages 308–317. ACM, 2012.

[12] 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 2021, October 4-8, 2021, Freiburg, Germany (Virtual Conference)*, pages 15:1–15:18, 2021.

[13] 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 2008, Beijing, China, April 21-25, 2008*, pages 377–386, 2008.

[14] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 524–533. IEEE Computer Society, 2003.

[15] Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. Near optimal LP rounding algorithm for correlation clustering on complete and complete k-partite graphs. *CoRR*, abs/1412.0681, 2014.

[16] Flavio Chierichetti, Nilesh N. Dalvi, and Ravi Kumar. Correlation clustering in mapreduce. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 641–650. ACM, 2014.

[17] Vincent Cohen-Addad, Silvio Lattanzi, Slobodan Mitrovic, Ashkan Norouzi-Fard, Nikos Parotsidis, and Jakub Tarnawski. Correlation clustering in constant many parallel rounds. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 2069–2078. PMLR, 2021.

[18] Manuela Fischer and Andreas Noever. Tight analysis of parallel randomized greedy MIS. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2152–2160, 2018.

[19] Manuela Fischer and Andreas Noever. Tight analysis of parallel randomized greedy MIS. *ACM Trans. Algorithms*, 16(1):6:1–6:13, 2020.

[20] Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the mapreduce framework. In *Proceedings of the 22nd International Conference on Algorithms and Computation (ISAAC)*, pages 374–383, 2011.

[21] Dmitri V. Kalashnikov, Zhaoqi Chen, Sharad Mehrotra, and Rabia Nuray-Turan. Web people search via connection analysis. *IEEE Trans. Knowl. Data Eng.*, 20(11):1550–1565, 2008.

[22] Sungwoong Kim, Chang Dong Yoo, Sebastian Nowozin, and Pushmeet Kohli. Image segmentation usinghigher-order correlation clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36 (9):1761–1774, 2014.

[23] Huy N. Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 327–336. IEEE Computer Society, 2008.

[24] 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 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 82–90, 2015.

[25] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In *Innovations in Computer Science - ICS 2011, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 223–238. Tsinghua University Press, 2011.

[26] Jessica Shi, Laxman Dhulipala, David Eisenstat, Jakub Lacki, and Vahab S. Mirrokni. Scalable community detection via parallel correlation clustering. *Proc. VLDB Endow.*, 14(11):2305–2313, 2021.

[27] Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum matchings. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 225–234. ACM, 2009.

A Size of $P_{r\text{-PIV}}$ vs P_{PIV}

In this section, we show that even though, by Theorem 1.1, algorithm r-PIVOT is almost as good as PIVOT in terms of approximating correlation clustering, the set $P_{r\text{-PIV}}$ of the pivots found by r-PIVOT tends to be significantly smaller than the set P_{PIV} of the pivots found by PIVOT.

Lemma A.1. *For any $r \geq 1$, there is an infinite family of n-vertex graphs for which*

$$\mathbf{E}\,|P_{r\text{-}PIV}| \leq r \cdot n^{-\Omega(1/r)} \cdot \mathbf{E}\,|P_{PIV}|.$$

Proof. We first construct a layered graph H, then the graph G of the lemma will be the line-graph of H. That is, G includes a vertex for each edge of H, and two vertices of G are adjacent if their corresponding edges in H share an endpoint. Observe that P_{PIV} for G corresponds to a maximal matching of H, and $P_{r\text{-PIV}}$ for G corresponds to a (not necessarily maximal) matching of H.

Let $t := 2r + 2$ where r is the parameter we run r-PIVOT with. Let N be a sufficiently large even integer controlling the number of vertices in H and let $\alpha = N^{1/3t} \geq 2$. The graph H has t layers of vertices $V_1, \ldots, V_t$ where $|V_i| = N/\alpha^{t-i}$. The graph has a perfect matching among the vertices in V_t. Additionally, there is a bipartite graph between every two consecutive layers. For any $i \in [t]$ define:

$$d_i^{\leftarrow} := \alpha^{2(t-i)+1}, \qquad d_i^{\rightarrow} := \alpha^{2(t-i)}.$$

For any $i \in [t-1]$, each vertex $v \in V_i$ has exactly $d_i^{\rightarrow}$ neighbors in V_{i+1}, and each vertex in V_{i+1} has exactly $d_{i+1}^{\leftarrow}$ neighbors in V_i. For this to be doable we need to have $|V_i| \cdot d_i^{\rightarrow} = |V_{i+1}| \cdot d_{i+1}^{\leftarrow}$ and $|V_1| \geq d_2^{\leftarrow}$. The former holds because

$$|V_i| \cdot d_i^{\rightarrow} = \frac{N}{\alpha^{t-i}} \cdot \alpha^{2(t-i)} = \frac{N}{\alpha^{t-(i+1)}} \cdot \alpha^{2(t-(i+1))+1} = |V_{i+1}| \cdot d_{i+1}^{\leftarrow},$$

and the latter holds because

$$|V_1| = N/\alpha^{t-1} = N/N^{\frac{t-1}{3t}} = N^{2/3+1/3t} \qquad \& \qquad d_2^{\leftarrow} = \alpha^{2(t-2)+1} = N^{\frac{2(t-2)+1}{3t}} \le N^{2/3}.$$

Since P_{PIV} is a maximal matching of H, it should be at least half the size of its maximum matching. Our construction has a perfect matching among the vertices of V_t. Thus, in any outcome of the PIVOT algorithm we must have

$$|P_{\mathrm{PIV}}| \ge \frac{1}{2} \cdot \frac{|V_t|}{2} = \Omega(N).$$

Next we show that $\mathbf{E}\,|P_{r\text{-PIV}}| = O(tN/\alpha) = rN^{1-\Omega(1/r)}$, which proves the claim.

Let π be a random rank over the *edges* of H (or equivalently the vertices of G). Given a vertex $v_t \in V_t$, define $Q(v_t)$ to be the event of having a path $(v_t, v_{t-1}, \ldots, v_1)$ in H where $v_i \in V_i$ and additionally (v_i, v_{i-1}) has the lowest rank in π among all edges of v_i. Observe that under this event, all edges of v_t remain unsettled by r-PIVOT after $(t-2)/2 = r$ rounds.

Let us lower bound $\Pr[Q(v_t)]$ for fixed $v_t \in V_t$. First, vertex v_t has $d_t^{\leftarrow}$ edges in V_{t-1} and only one edge in V_t. So the lowest rank edge of v is to some vertex $v_{t-1} \in V_{t-1}$ with probability

$$\frac{d_t^{\leftarrow}}{\deg(v_t)} = \frac{d_t^{\leftarrow}}{d_t^{\leftarrow}+1} = \frac{\alpha}{\alpha+1}.$$

Conditioned on this, the lowest rank edge of v_{t-1} goes to $v_{t-2} \in V_{t-2}$ with probability

$$\frac{d_{t-1}^{\leftarrow}}{\deg(v_{t-1}) + \deg(v_t) - 1} = \frac{d_{t-1}^{\leftarrow}}{d_{t-1}^{\leftarrow} + d_{t-1}^{\rightarrow} + d_t^{\leftarrow}} = \frac{\alpha^3}{\alpha^3 + \alpha^2 + \alpha} \ge \frac{\alpha}{\alpha+2},$$

(Where here we are using the fact that for disjoint sets of edges A and B, the lowest rank edge among B has a lower rank than the lowest rank edge of A with probability $\frac{|B|}{|B|+|A|}$.) Conditioned on this, the lowest rank edge of v_{t-2} goes to $v_{t-3} \in V_{t-3}$ with probability

$$\frac{d_{t-2}^{\leftarrow}}{\deg(v_{t-2}) + \deg(v_{t-1}) + \deg(v_t) - 2} = \frac{\alpha^5}{\alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha - 1} \ge \frac{\alpha}{\alpha+2}.$$
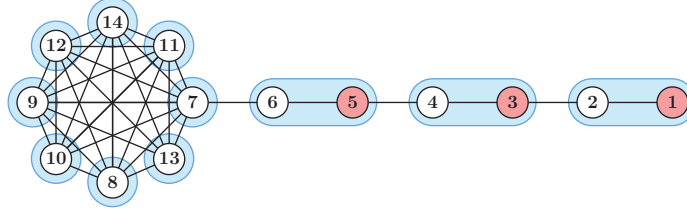
Continuing this argument all the way to V_1, we get

$$\Pr[Q(v_t)] \ge \left(\frac{\alpha}{\alpha+2}\right)^{t-1} = \left(1 - \frac{2}{\alpha+2}\right)^{t-1} \ge 1 - \frac{2(t-1)}{\alpha+2} = 1 - \Theta(t/\alpha). \quad \square$$

B Deterministic Approximation of r-PIVOT

In this section, we prove the following:

Lemma B.1. *For any $r < n/2$, there exists a graph G and a choice of π for r-PIVOT where*

$$\mathrm{COST}_{r\text{-PIV}}(G, \pi) = \Omega\left(\frac{n^2}{r} \cdot \mathrm{OPT}(G)\right).$$

Proof. For some parameter N, let G include a clique K_N and a path of length $2r$ attached to the clique. Let π be such that the vertices on the path have the lowest ranks in the graph and in a monotone increasing way from the degree one vertex of the path to its vertex attached to the clique. The ranks of the vertices in the clique can be arbitrary. An example construction for $r = 3$ and $N = 8$ is shown below, with the numbers on the vertices corresponding to the ranks in π.

The idea is that within the first r rounds of r-PIVOT, only the vertices of the path will be marked as pivots. Thus, all vertices of the clique will form singleton clusters. As a result, the cost of this clustering is at least $\Omega(N^2)$. On the other hand, by simply putting all the vertices of the clique in a cluster, the optimal solution only pays a cost of $O(r)$. Thus the cost of r-PIVOT is $\Omega(N^2/r)$ times that of the optimal solution. $\qquad\square$

C   Proof of Lemma 3.2

We provide a proof of Lemma 3.2 based on the following result:

Claim C.1 ([5])**.** *Let BT be the set of all bad triangles in G. Let $y : BT \to [0, 1]$ be such that $\sum_{t \in BT:a,b \in t} y(t) \leq 1$ for every distinct pair of vertices $a, b \in V$ (not necessarily connected in G). Then $\mathrm{OPT}(G) \geq \sum_{t \in BT} y(t)$.*

Proof. Consider the following LP, which clearly lower bounds the optimal solution since at least one pair of any bad triangle must be clustered incorrectly:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{a \neq b \in V} x_{\{a,b\}} \\
\text{subject to} \quad & x_{\{a,b\}} + x_{\{a,c\}} + x_{\{b,c\}} \geq 1 && \forall \{a, b, c\} \in BT \\
& x_{\{a,b\}} \geq 0 && \forall a \neq b \in V
\end{aligned}
$$

Now consider the dual:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{t \in BT} y_t \\
\text{subject to} \quad & \sum_{t \in BT:a,b \in t} y_t \leq 1 && \forall a \neq b \in V \\
& y_t \geq 0 && \forall t \in BT
\end{aligned}
$$

The lemma then follows because the objective value of any feasible solution to the dual lower bounds the primal objective. $\qquad\square$

Proof of Lemma 3.2 via Claim C.1. For any bad triangle $t \in BT$, let $x(t)$ be the expected number of charges of charging scheme $\mathcal{S}$ to t where the expectation is taken over π and the randomization of $\mathcal{S}$. First, note from the definition of charging schemes that

$$\mathbf{E}[|X|] = \sum_{t \in BT} \mathbf{E}\left[x(t)\right]. \tag{3}$$

Fix a pair $a, b \in V$ of distinct vertices. Since $\mathcal{S}$ is assumed to have width w, we have

$$\sum_{t \in BT: a, b \in t} \mathbf{E}[x(t)] \leq w. \tag{4}$$

Now let us define $y(t) := \mathbf{E}[x(t)]/w$ for any $t \in BT$. We have

$$\sum_{t \in BT: a, b \in t} y(t) = \frac{1}{w} \sum_{t \in BT: a, b \in t} \mathbf{E}[x(t)] \overset{(4)}{\leq} 1.$$

Therefore, we can apply Claim C.1 to get

$$\mathrm{OPT}(G) \geq \sum_{t \in BT} y(t) \overset{\text{(by definition of } y)}{=} \frac{1}{w} \sum_{t \in BT} \mathbf{E}[x(t)]. \tag{5}$$

Equations (3) and (5) together prove $\mathbf{E}[|X|] \leq w \cdot \mathrm{OPT}(G)$. □