

Space Efficient Approximation to Maximum Matching Size from Uniform Edge Samples

Michael Kapralov
EPFL

Slobodan Mitrović
MIT

Ashkan Norouzi-Fard
Google Zurich

Jakab Tardos
EPFL

Abstract

Given a source of iid samples of edges of an input graph G with n vertices and m edges, how many samples does one need to compute a constant factor approximation to the maximum matching size in G ? Moreover, is it possible to obtain such an estimate in a small amount of space? We show that, on the one hand, this problem cannot be solved using a nontrivially sublinear (in m) number of samples: $m^{1-o(1)}$ samples are needed. On the other hand, a surprisingly space efficient algorithm for processing the samples exists: $O(\log^2 n)$ bits of space suffice to compute an estimate.

Our main technical tool is a new peeling type algorithm for matching that we simulate using a recursive sampling process that crucially ensures that local neighborhood information from ‘dense’ regions of the graph is provided at appropriately higher sampling rates. We show that a delicate balance between exploration depth and sampling rate allows our simulation to not lose precision over a logarithmic number of levels of recursion and achieve a constant factor approximation. The previous best result on matching size estimation from random samples was a $\log^{O(1)} n$ approximation [Kapralov et al’14], which completely avoided such delicate trade-offs due to the approximation factor being much larger than exploration depth.

Our algorithm also yields a constant factor approximate local computation algorithm (LCA) for matching with $O(d \log n)$ exploration starting from any vertex. Previous approaches were based on local simulations of randomized greedy, which take $O(d)$ time *in expectation over the starting vertex or edge* (Yoshida et al’09, Onak et al’12), and could not achieve a better than d^2 runtime. Interestingly, we also show that unlike our algorithm, the local simulation of randomized greedy that is the basis of the most efficient prior results does take $\tilde{\Omega}(d^2) \gg O(d \log n)$ time for a worst case edge even for $d = \exp(\Theta(\sqrt{\log n}))$.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 3 |
| 1.1 | Related Work | 5 |
| 2 | Preliminaries | 6 |
| 3 | Our Algorithm and Technical Overview | 6 |
| 3.1 | Offline Algorithm | 6 |
| 3.2 | IID Edge Stream Version of Algorithm 1 | 8 |
| 4 | Sample Complexity of Algorithm 2 | 11 |
| 4.1 | Sample Complexity of Level Tests | 11 |
| 4.2 | Sample Complexity of IID-PEELING | 12 |
| 5 | Correctness of Algorithm 2 (proof of Theorem 5) | 14 |
| 5.1 | Definitions and Preliminaries | 14 |
| 5.2 | Lower Bound: Constructing a Near-Optimal Matching from \widehat{M} | 17 |
| 5.3 | Upper bound: Constructing a Near-Optimal Vertex Cover | 19 |
| 6 | LCA | 22 |
| 6.1 | Overview of Our Approach | 23 |
| 6.2 | Related Work | 23 |
| 6.3 | Algorithms | 24 |
| 6.4 | Query Complexity | 25 |
| 6.5 | Approximation Guarantee | 27 |
| 6.6 | Memory Complexity and Consistent Oracles | 27 |
| 6.7 | Proof of Theorem 2 | 29 |
| 7 | Lower Bound of $\widetilde{\Omega}(d^2)$ for Simulation of Randomized Greedy | 29 |
| 7.1 | Lower Bound for Infinite Graphs | 30 |
| 7.2 | Depth and Variance Analysis for Infinite Graphs | 34 |
| 7.3 | The Lower Bound Instance (Truncated $H^{d,\epsilon}$) | 35 |
| 8 | Lower-bound on the Number of Sampled Edges | 36 |
| 8.1 | Overview | 36 |
| 8.2 | Preliminaries | 38 |
| 8.3 | Constructing Graphs with Identical k -Level Degrees | 39 |
| 8.4 | Increasing Girth via Graph Lifting | 43 |
| 8.5 | k -Edge Subgraph Statistics in G and H | 45 |
| 9 | Analysis of the algorithm on a random permutation stream | 46 |
| 9.1 | Introduction and Technical Overview | 46 |
| 9.2 | Preliminaries | 47 |
| 9.3 | Padding and total variation distance | 48 |
| 9.4 | Bounding KL-divergence | 50 |
| 9.5 | The full algorithm | 55 |
| A | Proof of Lemma 5 | 62 |
| B | Oversampling Lemma | 65 |
| C | Proofs omitted from Section 7 | 67 |
| D | Details omitted from Section 8 | 68 |
| D.1 | Proof of Theorem 9 | 68 |
| D.2 | Proof of Lemma 23 | 69 |
| E | Proofs omitted from Section 9 | 70 |

1 Introduction

Large datasets are prevalent in modern data analysis, and processing them requires algorithms with a memory footprint much smaller than the size of the input, i.e. sublinear space. The streaming model of computation, which captures this setting, has received a lot of attention in the literature recently. In this model the edges of the graph are presented to the algorithm as a stream in some order. It has recently been shown that randomly ordered streams allow for surprisingly space efficient estimation of graph parameters by nontrivial memory vs sample complexity tradeoffs (see, e.g. [KKS14, MMPS17, PS18] for approximating matching size and other graph properties such as number of connected components, weight of MST and independent set size). Memory vs sample complexity tradeoffs for learning problems have also recently received a lot of attention in literature [Raz16, Raz17, KRT17, GRT18, BGY18]. In this paper we study the following question:

How many iid samples of edges of a graph G are necessary and sufficient for estimating the size of the maximum matching in G to within a constant factor?
Can such an estimate be computed using small space?

We give nearly optimal bounds for both questions, developing a collection of new techniques for efficient simulation of matching algorithms by random sampling. Our main result is

Theorem 1. *There exists an algorithm (Algorithm 2) that, given access to iid edge-samples of a graph $G = (V, E)$ with n vertices and m edges produces a constant factor approximation to maximum matching size in G using $O(\log^2 n)$ bits of memory and at most m samples with probability at least $4/5$.*

The sample complexity of Algorithm 2 is essentially optimal: for every constant C , every m between $n^{1+o(1)}$ and $\Omega(n^2)$ it is information theoretically impossible to compute a C -approximation to maximum matching size in a graph with high constant probability using fewer than $m^{1-o(1)}$ iid samples from the edge set of G , even if the algorithm is not space bounded.

The proof of the upper bound part in Theorem 1 is given in Section 4 (more precisely, Section 4.2). The proof of the lower bound part of Theorem 1 follows from Theorem 8 in Section 8.

The core algorithmic tool underlying Theorem 1 is a general method for implementing peeling type algorithms efficiently using sampling. In particular, our approach almost directly yields a constant factor approximate local computation algorithm (LCA) for maximum matching in a graph G with degrees bounded by d using $O(d \log n)$ queries and $O(\log d)$ exploration depth, whose analysis is deferred to Section 6.

Theorem 2. *Let G be a graph with n vertices and maximum degree d . Then there exists a random matching M , such that $\mathbb{E}[|M|] = \Theta(MM(G))$, and an algorithm that with high probability:*

- *Given an edge e of G , the algorithm reports whether e is in M or not by using $O(d \log n)$ queries.*
- *Given a vertex v of G , the algorithm reports whether v is in M or not by using $O(d \log n)$ queries.*

Moreover, this algorithm can be executed by using $O(d \log^3 n)$ bits of memory.

We note that the most efficient LCA's for matching [LRY17] require $d^4 \log^{O(1)} n$ exploration to achieve a constant factor approximation, and are based on the idea of simulating the randomized greedy algorithm locally. Indeed, this runtime complexity follows from the beautiful result of Yoshida et al. [YYI09] or Onak et al. [ORRR12] that shows the size of the query tree of the randomized greedy is $O(d)$ in expectation over the starting edge. Applying a Markov bound to discard vertices/edges on which the exploration takes too long leads to the desired complexity. Moreover, in a degree d bounded graph matching size could be as small as n/d , with only a $1/d$ fraction of vertices and edges involved in the matching. Therefore, a multiplicative (as opposed to multiplicative-additive) constant factor approximation based on average case results of [YYI09, ORRR12] must use a Markov bound with a loss of at least a factor of d in the size of the query tree with respect to the average, and hence cannot lead to a better than $O(d^2)$ runtime overall. At the same time, Theorem 2 yields the near-optimal runtime of $O(d \log n)$, going well beyond what is achievable with the above mentioned average case results.

At this point it is natural to wonder if the average case analysis of [YYI09, ORRR12] can be improved to show that the size of the query tree is $O(d)$ in expectation for any given edge as opposed to for a random one. Surprisingly, we show in Section 7 that this is not possible:

Theorem 3. *There exists an absolute constant $b > 0$ such that for every $n, d \in [5, \exp(b\sqrt{\log n})]$ and $\epsilon \in [1/d, 1/2]$ there exists a graph G with n vertices and maximum degree $d + 1$, and an edge e such that running $\text{YYI-MAXIMAL-MATCHING}(e, \pi)$ from [Algorithm 10](#) results in an exploration tree of size at least*

$$\frac{1}{8} \cdot \epsilon \cdot \left(\frac{d}{2}\right)^{2-\epsilon},$$

in expectation.

In [Section 9](#) we prove that our algorithm is robust to correlations that result from replacing the iid stream of edge-samples with a random permutation stream. Formally, we prove

Theorem 4. *There exists an algorithm that given access to a random permutation edge stream of a graph $G = (V, E)$, with n vertices and $m \geq 3n$ edges, produces an $O(\log^2 n)$ factor approximation to maximum matching size in G using $O(\log^2 n)$ bits of memory in a single pass over the stream with probability at least $3/4$.*

[Theorem 4](#) improves upon the previous best known $\log^{O(1)} n$ -approximation due to [\[KKS14\]](#), where the power of the logarithm was quite large (we estimate that it is at least 8). The proof is given in [Section 9](#).

Let us now provide a brief overview of some of the techniques we use in this paper.

Our techniques: approximating matching size from iid samples. As mentioned above, our approach consists of efficiently simulating a simple peeling-type algorithm, using a small amount of space and samples. This approach was previously used in [\[KKS14\]](#) to achieve a polylogarithmic approximation to maximum matching size in polylogarithmic space. As we show below, major new ideas are needed to go from a polylogarithmic approximation to a constant factor approximation. The reason for this is the fact that any matching size approximation algorithm needs to perform very deep (logarithmic depth) exploration in the graph, leading to $\omega(1)$ long chains of dependencies (i.e., recursive calls) in any such simulation. Indeed, both the work of [\[KKS14\]](#) and our result use a peeling type algorithm that performs a logarithmic number of rounds of peeling as a starting point. The work of [\[KKS14\]](#) simulated this process by random sampling, oversampling various tests by a logarithmic factor to ensure a high degree of concentration, and losing extra polylogarithmic factors in the approximation to mitigate the effect of this on sample complexity of the recursive tests. Thus, it was crucial for the analysis of [\[KKS14\]](#) that the approximation factor is much larger than the depth of exploration of the algorithm that is being simulated. In our case such an approach is provably impossible: a constant factor approximation requires access to $\Omega(\log n)$ depth neighborhoods by known lower bounds (see [\[KMW16\]](#) and [Section 8](#)). A method for circumventing this problem is exactly the main contribution of our work – we show how to increase sampling rates in deeper explorations of the graph, improving confidence of statistical estimation and thereby avoiding a union bound over the depth, while at the same time keeping the number of samples low. A detailed description of the algorithm and the analysis are presented in [Section 3](#).

Our techniques: tight sample complexity lower bound and tight instances for peeling algorithms. As the second result in [Theorem 1](#) shows, the sample complexity of our algorithm is essentially best possible even among algorithms that are not space constrained. The lower bound (see [Section 8](#)) is based on a construction of two graphs G and H on n vertices such that for a parameter k **(a)** matching size in G is smaller than matching size in H by a factor of $n^{\Omega(1)/k}$ but **(b)** there exists a bijection from vertices of G to vertices of H that preserves k -depth neighborhoods up to isomorphism. To the best of our knowledge, this construction is novel. Related constructions have been shown in the literature (e.g. cluster trees of [\[KMW16\]](#)), but these constructions would not suffice for our lower bound, since they do not provide a property as strong as **(b)** above.

Our construction proceeds in two steps. We first construct two graphs G' and H' that have identical k -level degrees (see [Section 8.3](#)). These two graphs are indistinguishable based on k -level degrees and their maximum matching size differs by an $n^{\Omega(1/k)}$ factor, but their neighborhoods are not isomorphic due to cycles. G' and H' have $n^{2-O(1/k)}$ edges and provide nearly tight instances for peeling algorithms that we hope may be useful in other contexts. The second step of our construction is a lifting map (see [Theorem 11](#) in [Section 8](#)) that relies on high girth Cayley graphs and allows us to convert graphs with identical k -level vertex degrees to graphs with isomorphic depth- k neighborhoods without changing matching size by much. The details are provided in [Section 8.4](#).

Finally, the proof of the sampling lower bound proceeds as follows. To rule out factor C approximation in $m^{1-o(1)}$ space, take a pair of constant (rather, $m^{o(1)}$) size graphs G and H such that **(a)** matching size in G is smaller than matching size in H by a factor of C and **(b)** for some large k one has that k -depth neighborhoods in G are isomorphic to k -depth neighborhoods in H . Then the actual hard input distribution consists of a large number of disjoint copies of G in the **NO** case and a large number of copies of H in the **YES** case, possibly with a small disjoint clique added in both cases to increase the number of edges appropriately. Since the vertices are assigned uniformly random labels in both cases, the only way to distinguish between the **YES** and the **NO** case is to ensure that at least k edge-samples land in one of the small copies of H or G . Since k is small, the result follows. The details can be found in [Section 8](#).

Our techniques: random permutation streams. As mentioned above, in [Section 9](#) (see [Theorem 4](#)) we extend our result to a stream of edges in random order which improves upon the previous best known $\log^{O(1)} n$ -approximation due to [\[KKS14\]](#), where the power of the logarithm was quite large (we estimate that it is at least 8). In order to establish [Theorem 4](#) we exhibit a coupling between the distribution of the state of the algorithm when run on an iid stream and when run on a permutation stream. The approach is similar in spirit to that of [\[KKS14\]](#), but carrying it out for our algorithm requires several new techniques. Our approach consists of bounding the KL divergence between the distribution of the state of the algorithm in the iid and random permutation settings by induction on the level of the tests performed in the algorithm. In order to make this approach work, we develop a restricted version of triangle inequality for KL divergence that may be of independent interest (see [Lemma 27](#) in [Section 9](#)).

1.1 Related Work

Over the past decade, matchings have been extensively studied in the context of streaming and related settings. The prior work closest to ours is by Kapralov et al. [\[KKS14\]](#). They design an algorithm that estimates the maximum matching size up to $\text{polylog } n$ factors by using at most m iid edge-samples. Their algorithm requires $O(\log^2 n)$ bits of memory. As they prove, this algorithm is also applicable to the scenario in which the edges are provided as a random permutation stream. The problem of approximating matching size using $o(n)$ space has received a lot of attention in the literature, including random order streams [\[CJMM17, MMPS17\]](#) and worst case streams [\[MV18, BGM⁺19, MV16, CCE⁺16, EHL⁺15, EHM16, BS15, AKL17\]](#). The former, i.e., [\[MMPS17, CJMM17\]](#), are the closest to our setting since both of these works consider random streams of edges. However, the results mostly apply to bounded degree graphs due to an (at least) exponential dependence of the space on the degree. The latter consider worst case edge arrivals, but operate under a bounded arboricity assumption on the input graph. Very recently constant space algorithms for approximating some graph parameters (such as number of connected components and weight of the minimum spanning tree) from random order streams were obtained in [\[PS18\]](#) (see also [\[MMPS17\]](#)).

An extensive line of work focused on computing approximate maximum matchings by using $\tilde{O}(n)$ memory. The standard greedy algorithm provide a 2 approximation. It is known that no single-pass streaming algorithm (possibly randomized) can achieve better than $1 - 1/e$ approximation while using $\tilde{O}(n)$ memory [\[Kap13, GKK12\]](#). For both unweighted and weighted graphs, better results are known when a stream is randomly ordered. In this scenario, it was shown how to break the barrier of 2 approximation and obtain a $2 - \epsilon$ approximation, for some small constant $\epsilon > 0$ [\[KMM12, Kon18, AKLY16, AK17, PS17, ABB⁺19, AB19, GW19, GKMS18\]](#). If multiple passes over a stream are allowed, a line of work [\[FKM⁺05, EKS09, AG11b\]](#) culminated in an algorithm that in $O(\text{poly}(1/\epsilon))$ passes and $\tilde{O}(n)$ memory outputs a $(1 + \epsilon)$ -approximate maximum matching in bipartite graphs. In the case of general graphs, $(1 + \epsilon)$ -approximate weighted matching can be computed in $(1/\epsilon)^{O(1/\epsilon)}$ passes and also $\tilde{O}(n)$ memory [\[McG05, ELMS11, Zel12, AG11a, AG11b\]](#) and [\[GKMS18\]](#).

In addition to the LCA results we pointed to above, close to our work are [\[RTVX11, ARVX12, LRY17, Gha16, GU19\]](#) who also study the worst-case oracle behavior. In particular, [\[GU19\]](#) showed that there exists an oracle that given an arbitrary chosen vertex v outputs whether v is in some fixed maximal independent set or not by performing $d^{O(\log \log d)} \cdot \text{polylog } n$ queries. When this oracle is applied to the line graph, then it reports whether a given edge is in a fixed maximal matching or not. We provide further comments on LCA related work in [Section 6.2](#).

2 Preliminaries

Graphs In this paper we consider unweighted undirected graphs $G = (V, E)$ where V is the vertex set and E is the edge set of the graph. We denote $|V|$ by n and $|E|$ by m . Furthermore, we use d to signify the maximum degree of G or an upper bound on the maximum degree.

Matching In this paper, we are concerned with estimating the size of a maximum matching. Given a graph $G = (V, E)$, a *matching* $M \subseteq E$ of G is a set of pairwise disjoint edges, i.e., no two edges share a common vertex. If M is a matching of the maximum cardinality, then M is also said to be a *maximum matching*. We use $\text{MM}(G)$ to refer to the cardinality of a maximum matching of G . A *fractional matching* $M_f : E \rightarrow \mathbb{R}^+$ is a function assigning weights to the edges such that the summation of weights assigned to the edges connected to each vertex is at most one, i.e., $\sum_{e \in E: v \in e} M_f(e) \leq 1$, for each vertex v . The size of a fractional matching (denoted by $|M_f|$) is defined to be the summation of the weights assigned to the edges. Notice that any matching can also be seen as a fractional matching with weights in $\{0, 1\}$. It is well-known that

$$\text{MM}(G) \leq \max_{\text{fractional matching } M_f \text{ of } G} |M_f| \leq \frac{3}{2} \text{MM}(G),$$

hence,

$$\text{MM}(G) = \max_{\text{fractional matching } M_f \text{ of } G} \Theta(|M_f|).$$

Therefore, to estimate the cardinality of a maximum matching, it suffice to estimates the size of a fractional maximum matching. In this work, we show that the estimate returned by our algorithm is by a constant factor smaller than the size of a fractional maximum matching, which implies that it is also by a constant factor smaller than $\text{MM}(G)$.

Vertex cover We also lower-bound the estimate returned by our algorithm. To achieve that, we prove that there is a vertex cover in G of the size within a constant factor of the output of our algorithm. Given a graph $G = (V, E)$, a set $C \subseteq V$ is a *vertex cover* if each edge of the graph is incident to at least one vertex of the set C . It is folklore that the size of any vertex cover is at least the size of any matching.

Vertex neighborhood Given a graph $G = (V, E)$, we use $N(v)$ to denote the vertex neighborhood of v . That is, $N(v) = \{w : \{v, w\} \in E\}$.

3 Our Algorithm and Technical Overview

In this section we present our algorithm for estimating the matching size from at most m iid edge-samples. We begin by providing an algorithm that estimates the matching size while having full access to the graph (see [Algorithm 1](#)) and then, in [Section 3.2](#), we show how to simulate [Algorithm 1](#) on iid edge-samples. We point out that [Algorithm 1](#) uses $O(m)$ memory, while our simulation uses $O(\log^2 n)$ bits of memory.

3.1 Offline Algorithm

First we introduce a simple peeling algorithm which we will be simulating locally. This algorithm constructs a fractional matching M and a vertex cover C which are within a constant factor of each other. As noted in [Section 2](#), by duality theory this implies that both M and C are within a constant factor of the optimum. [Algorithm 1](#) begins with both M and C being empty sets and augments them simultaneously in rounds. When the weight of a vertex¹ v is higher than a threshold δ , the algorithm adds v to the vertex cover C and removes v along with all its incident edges. When this happens, we say that v and its edges have been *peeled off*. In the i^{th} round the weight of the matching on each edge that has not yet been peeled off is increased by c^{i-1}/d . For any $v \in V$, we denote the total weight of M adjacent to v by

$$M(v) = \sum_{w \in N(v)} M((v, w)).$$

¹Remark that we use the term *weight of an edge* to refer to the fractional matching assigned to that edge, i.e., the value computed by [EDGE-LEVEL-TEST Algorithm 3](#). Similarly, we use the term *weight of a vertex* as the summation of the weights of the edges connected to it.

Algorithm 1 Offline peeling algorithm for constructing a constant factor approximate maximum fractional matching and a constant factor approximate minimum vertex cover

```

1: procedure GLOBAL-PEELING( $G = (V, E), \delta, c$ )           ▷  $\delta$  and  $c$  are two constants that we fix later
2:    $A \leftarrow V$                                            ▷ Set of active vertices
3:    $C \leftarrow \emptyset$                                    ▷  $C$  is the vertex cover that the algorithm constructs
4:    $M : E \rightarrow \mathbb{R}$ 
5:    $M(e) \leftarrow 0 \quad \forall e \in E$                      ▷ Initially, the fractional matching for all the edges is zero
6:   for  $i = 1$  to  $J + 1$  do                               ▷ Represents the rounds of peeling
7:     for  $e \in E \cap A \times A$  do                         ▷ Remaining edges
8:        $M(e) \leftarrow M(e) + c^{i-1}/d$                  ▷ Increases weight on edges
9:     for  $v \in A$  do
10:      if  $M(v) \geq \delta$  then                               ▷ Recall that  $M(v) = \sum_{w \in N(v)} M((v, w))$ 
11:         $A \leftarrow A \setminus \{v\}$                        ▷ Remove the vertex
12:         $C \leftarrow C \cup \{v\}$                          ▷ Add the vertex to the vertex cover
13:   return  $(M, C)$ 

```

This process continues for $J + 1$ rounds before all the edges (but not necessarily all the vertices) are peeled off from the graph. Note that at the last ($J + 1^{\text{st}}$) iteration the amount of weight assigned to each of the non-peeled-off edges is at least c^J/d and hence it suffices to set $J = \Theta(\log d)$ for all the vertices v with non-zero degree to be peeled off. Therefore, at this point C is indeed a vertex cover.

At the termination of **Algorithm 1**, any vertex v that has been added to C at some point must have at least δ matching adjacent to it. More precisely,

$$\sum_{e \in E} M(e) \geq \delta |C| \geq \delta \text{MM}(G). \quad (1)$$

However, since the rate at which M is increased on each edge only increases by a multiplicative factor of c per round, the weight adjacent to any vertex cannot be much higher. Indeed, if v is peeled off in round $i + 1$, then $M(v)$ is at most δ in round i . In the intervening one round $M(v)$ could increase by at most $c\delta$, therefore $M(v)$ is at most $(c + 1)\delta$ at the point when v is peeled off. Naturally, if a vertex is peeled off in the first round, $M(v)$ is at most 1. In conclusion $M(v)$ is at most $\max((c + 1)\delta, 1)$; thus scaling down M by a factor of $\max((c + 1)\delta, 1)$ produces a (valid) fractional matching which is within a $\max(c + 1, 1/\delta)$ factor of C . Therefore,

$$\sum_{e \in E} M(e) \leq \max((c + 1)\delta, 1) \frac{3}{2} \text{MM}(G), \quad (2)$$

where the factor $3/2$ is the result of the gap between a fractional matching and the maximum matching explained in preliminaries. Combining **Eq. (1)** and **Eq. (2)** we get a constant approximation guarantee.

Main challenges in simulating **Algorithm 1 and comparison to [KKS14].** The approach of starting with a peeling type algorithm and performing a small space simulation of that algorithm by using edge-samples of the input graph has been used in [KKS14]. However, the peeling algorithm that was used is significantly weaker than **Algorithm 1** and hence much easier to simulate, as we now explain. Specifically, the algorithm of [KKS14] repeatedly peels off vertices of sufficiently high degree, thereby partitioning edges of the input graph into classes, and assigns *uniform weights* on edges of every class (vertices of degree $\approx c^i$ in the residual graph are assigned weight $\approx c^{i-1}/n$). This fact that the weights are uniform simplifies simulation dramatically, at the expense of only an $O(\log n)$ approximation. Indeed, in the algorithm of [KKS14] the weight on an edge is equivalent to the level at which the edge disappears from the graph, and only depends on the number of neighbors that the endpoints have one level lower. In our case the weight of an edge at level i is composed of contributions from **all levels smaller than i** . As we show below, estimating such weights is much more challenging due to the possibility to errors accumulating across the chain of $\Omega(\log d)$ (possibly $\Omega(\log n)$) levels. In fact, techniques for coping with this issue, i.e., avoiding a union bound over $\log n$ levels, are a major contribution of our work.

3.2 IID Edge Stream Version of Algorithm 1

In this section we describe our simulation of Algorithm 1 in the regime in which an algorithm can learn about the underlying graph only by accessing iid edges from the graph.

Remark 1. In the following we will assume that $m \geq n$ for simplicity. In the case where this is not true we can simply modify the graph by adding a new vertex v_0 to V and adding an n -star centered at v_0 to the input graph G to get G' . This increases the matching size by at most 1. Also, knowing m and V we can simply simulate sampling an iid edge G' . Indeed, whenever we need to sample an edge of G' with probability $\frac{m}{n+m}$ we sample an iid edge of G and with probability $\frac{n}{n+m}$ we sample uniformly from $v_0 \times V$. For simplicity we will omit this subroutine from our algorithms and assume that $m \geq n$ in the input graph G .

Remark 2. Recall that d is an upper bound on the maximum degree of G . For the purposes of the iid sampling algorithm, we can simply set d to be n , as it can be any upper bound on the maximum degree. In this model of computation the runtime will actually not depend on d . We still carry d throughout the analysis, as it will be crucial in the LCA implementation (Section 6). However, the reader is encouraged to consider $d = n$ for simplicity.

Algorithm 2 IID edge sampling algorithm approximating the maximum matching size of G , $\text{MM}(G)$.

```

1: procedure IID-PEELING( $G = (V, E)$ )
2:    $M' \leftarrow 0$  ▷ The value of estimated matching
3:    $t \leftarrow 1$  ▷ The number of iid edges that we run EDGE-LEVEL-TEST on
4:   while samples lasts do
5:      $M' \leftarrow \text{SAMPLE}(G, t)$ 
6:      $t \leftarrow 2t$ 
7:   return  $M'$  ▷ The estimated size of the matching that our algorithm returns
8:
9: procedure SAMPLE( $G = (V, E), t$ )
10:   $M' \leftarrow 0$  ▷ Starting fractional matching
11:  for  $k = 1$  to  $t$  do
12:     $e \leftarrow$  iid edge from the stream
13:     $M' \leftarrow M' + \text{EDGE-LEVEL-TEST}(e)$  ▷ The fractional matching assigned to this edge
14:  return  $m \cdot \frac{M'}{t}$ 

```

Algorithm 3 Given an edge e , this algorithm returns a fractional matching-weight of e .

```

1: procedure EDGE-LEVEL-TEST( $e = (u, v)$ )
2:   $w \leftarrow 1/d$  ▷ By definition, every edge gets the weight  $1/d$ 
3:  for  $i = 1$  to  $J + 1$  do ▷ Recall that  $J = \lfloor \log_c d \rfloor - 1$ 
4:    if LEVEL- $i$ -TEST( $u$ ) and LEVEL- $i$ -TEST( $v$ ) then ▷ Check if both endpoints pass the  $i$ -th test
5:       $w \leftarrow w + c^i/d$  ▷ Increase the weight of the edge
6:    else
7:      return  $w$ 
8:  return  $w$ 

```

Algorithm 2 is a local version of Algorithm 1 which can be implemented by using iid edge-samples. Notice that for the sake of simplicity in both presentation of the algorithms and analysis, we referred to the iid edge oracle as stream of random edges. Instead of running the peeling algorithm on the entire graph, we select a uniformly random edge e and estimate what the value of M in Algorithm 1 would be on this edge. This is done by the procedure EDGE-LEVEL-TEST(e) (Algorithm 3). The procedures IID-PEELING and SAMPLE are then used to achieve the desired variance and will be discussed in Section 4; they are not necessary for now, to understand the intuition behind Algorithm 2.

Notice that in Algorithm 1 if the two endpoints of $e = \{u, v\}$ get peeled off at rounds i_1 and i_2 respectively, then we can determine the value of $M(e)$ to be $\sum_{i=1}^{\min(i_1, i_2)} c^{i-1}/d$. Therefore, when evaluating EDGE-LEVEL-TEST(e) we need only to estimate what round u and v make it to. To this end we use the procedure LEVEL- j -TEST(v) which estimates whether a vertex survives the j^{th} round of

Algorithm 4 Given a vertex v that has passed the first j rounds of peeling, this algorithm returns whether or not it passes the $j + 1^{\text{st}}$ round.

```

1: procedure LEVEL- $(j + 1)$ -TEST( $v$ )
2:    $S \leftarrow 0$  ▷ The estimate of the weight of this vertex
3:   for  $k = 1$  to  $c^j \cdot \frac{m}{d}$  do
4:      $e \leftarrow$  next edge in the stream ▷ Equivalent of sampling and iid edge
5:     if  $e$  is adjacent to  $v$  then
6:        $w \leftarrow$  the other endpoint of  $e$ 
7:        $i \leftarrow 0$  ▷ Represents the last level that  $w$  passes
8:       while  $i \leq j$  and LEVEL- $i$ -TEST( $w$ ) do ▷ LEVEL-0-TEST returns TRUE by definition
9:          $S \leftarrow S + c^{i-j}$  ▷ The contribution that edge  $e$  gives with respect to the current
           sampling rate
10:      if  $S \geq \delta$  then ▷ If yes, then the weight of the vertex is high
11:        return FALSE
12:       $i \leftarrow i + 1$ 
13: return TRUE

```

Algorithm 1. However, instead of calculating $M(v)$ exactly, by looking recursively at all neighbors of v , we only sample the neighborhood of v at some appropriate rate to get an unbiased estimator of $M(v)$.

Additionally, both EDGE-LEVEL-TEST and LEVEL- j -TEST determinate early if the output becomes clear. In EDGE-LEVEL-TEST(e), if either endpoint returns FALSE in one of the tests we may stop, since the value of $M(e)$ depends only on the endpoint that is peeled off earlier. In LEVEL- j -TEST, if the variable 'S', which is used as an accumulator, reaches the threshold δ we may stop and return FALSE even if the designated chunk of the stream has not yet been exhausted. This speed-up will be crucial in the sample complexity analysis that we provide in [Section 4](#).

Main challenges that [Algorithm 3](#) resolves and comparison to [\[KKS14\]](#). We now outline the major ideas behind our constant factor approximation algorithm and compare them to the techniques used by the polylogarithmic approximation of [\[KKS14\]](#).

Precision of level estimates despite lack of concentration. As discussed above, the crucial feature of our algorithm is the fact that the total weight of a vertex that is assigned to level j (i.e., fails LEVEL- j -TEST) is contributed by vertices at all lower levels, in contrast to the work of [\[KKS14\]](#), where only contributions from the previous level were important. Intuitively, a test is always terminated as soon as it would need to consume more samples than expected. In fact, one can show that without this even a single call of LEVEL- j -TEST may run for $\omega(m)$ samples in expectation, in graphs similar to the hard instances for greedy, constructed in [Section 7](#). As we will see in [Section 4](#) the early stopping rule lends itself to extremely short and elegant analysis of sample complexity. However, the correctness of [Algorithm 2](#) is much less straightforward. For example, note that we are approximately computing the weight of a vertex at level j by sampling, and need such estimates to be precise. The approach of [\[KKS14\]](#) to this problem was simple: $C \log n$ neighbors on the previous level were observed for a large constant factor C to ensure that all estimates concentrate well around their expectation. This lead to a blowup in sample complexity, which was reduced by imposing an aggressive pruning threshold on the contribution of vertices from the previous level, at the expense of losing another logarithmic factor in the approximation quality. The work of [\[KKS14\]](#) could afford such an approach because the approximation factor was much larger than the depth of the recursive tests (approximation factor was polylogarithmic, whereas the depth merely logarithmic). Since we are aiming for a constant factor approximation, we cannot afford this approach.

The core of our algorithm is LEVEL- j -TEST of [Algorithm 4](#). Indeed, once the level of two vertices, u and v , have been determined by repeated use of LEVEL- j -TEST to be $\widehat{L}(u)$ and $\widehat{L}(v)$ respectively, the connecting edge has fractional weight of exactly

$$\widehat{M}(u, v) \stackrel{\text{def}}{=} \sum_{i=0}^{\min\{\widehat{L}(u), \widehat{L}(v)\}} c^i / d. \quad (3)$$

This means that the size of the matching constructed by the algorithm is $\sum_{e=(u,v) \in E} \widehat{M}(u, v)$, and our matching size estimation algorithm ([Algorithm 2](#)) simply samples enough edges to approximate the

sum to a constant factor multiplicatively. Thus, in order to establish correctness of our algorithm it suffices to show that $\sum_{e=(u,v) \in E} \widehat{M}(u,v)$ is a constant factor approximation to the size of the maximum matching in G . We provide the formal analysis in [Section 5](#), and give an outline here for convenience of the reader.

Our goal will be to prove that the total weight of \widehat{M} on all the edges is approximately equal to the maximum matching size of G . Consider by analogy the edge weighting M of [Algorithm 1](#). Here, M is designed to be such that the weight adjacent to any vertex (that is the sum of the weight of adjacent edges) is about a constant, with the exception of the vertices that make it to the last $(T + 1^{\text{st}})$ peeling level. This is sufficient to prove correctness. \widehat{M} is designed similarly, however, classifications of some vertices may be inaccurate.

First of all, if a vertex is misclassified to a peeling level much higher than where it should be, it may have a superconstant amount of weight adjacent to it. Since among n vertices some few are bound to be hugely misclassified we cannot put a meaningful upper bound on the weight adjacent to a worst case vertex; we cannot simply say that \widehat{M} , when normalized by a constant, is a fractional matching. Instead, we choose some large constant threshold λ and discard all vertices whose adjacent weight is higher than λ (we call these bad vertices), then normalize the remaining matching by λ . By analyzing the concentration properties of \widehat{M} in [Corollary 1](#), we get that \widehat{M} concentrates quadratically around its expectation, that is

$$\mathbb{P}[\widehat{M}(v) > x] = O\left(\mathbb{E}[\widehat{M}(v)]/x^2\right).$$

Using this we can show that discarding bad vertices will not significantly change the total weight of the graph. For details see [Section 5.2](#) and particularly [Corollary 2](#), which states that only a small fraction of the total weight is adjacent to bad vertices, in expectation:

$$\mathbb{E}[\widehat{M}(v) \cdot \mathbb{1}[\widehat{M}(v) \geq \lambda]] \leq \frac{1}{4} \mathbb{E}[\widehat{M}(v)].$$

A more difficult problem to deal with is that vertices may be misclassified to lower peeling levels than where they should be. Indeed, the early stopping rule, (see [Line 10](#) of [Algorithm 4](#)) means that vertices have a lot of chances to fail early. For example, a vertex v which *should* reside at some high $(\Theta(\log n))$ level must survive $\Theta(\log n)$ inaccurate tests to get there. One might reasonably think that some vertices are misclassified to lower levels even in the typical case; this however turns out to not be true. The key realization is that the $\text{LEVEL-}(j+1)\text{-TEST}(v)$ behaves very similarly to the $\text{LEVEL-}j\text{-TEST}(v)$ with one of the crucial differences being that $\text{LEVEL-}(j+1)\text{-TEST}(v)$ samples the neighborhood of v c -times more aggressively. A multiplicative increase in sampling rate translates to a multiplicative decrease in the error probability of the test, as formalized by [Lemma 1](#) from [Section 5.3](#):

Lemma 1 (Oversampling lemma). *For sufficiently small $\delta > 0$ and large enough c the following holds. Let $X = \sum_{k=1}^K Y_k$ be a sum of independent random variables Y_k taking values in $[0, 1]$, and $\bar{X} \stackrel{\text{def}}{=} \frac{1}{c} \sum_{i=1}^c X_i$ where X_i are iid copies of X . If $\mathbb{E}[X] \leq \delta/3$ and $\mathbb{P}[X \geq \delta] = p$, then $\mathbb{P}[\bar{X} \geq \delta] \leq p/2$.*

More formally, consider some vertex v whose peeling level *should* be about $j^* = \Theta(\log n)$. When running $\text{LEVEL-}j\text{-TEST}(v)$ for some $j \ll j^*$ the variable S in [Algorithm 4](#) is an unbiased estimator of the weight currently adjacent on v , and $\mathbb{E}[S] \ll \delta$. However, we have no bound on the variance of S and so it is entirely possible that with as much as constant probability S exceeds δ and the test exits in [Line 10](#) to return false. Over a logarithmic number of levels we cannot use union bound to bound the error probability. Instead, let S' be the same variable in the subsequent run of $\text{LEVEL-}(j+1)\text{-TEST}(v)$. S' can be broken into contributions from neighbors at levels below j (denoted by A below), and contributions from the neighbors at level j (denoted by B below). It can be shown that

$$S' = A + B$$

and

$$\mathbb{P}[S' \geq \delta] \leq \mathbb{P}[A \geq \delta] + \mathbb{P}[B \geq 1],$$

due to the integrality of B . However, A is simply an average taken over c iid copies of S , and so we can apply the oversampling lemma, with $X = S$. (Note that S satisfies the condition of being the independent sum of bounded variables, as different iterations of the for loop in [Line 3](#) are independent and the contribution of each to S is small.) As a result we get

$$\mathbb{P}[A \geq \delta] \geq \frac{1}{2} \cdot \mathbb{P}[S \geq \delta]$$

which ultimately allows us to get a good bound on the total error probability, summing over all j^* levels.

Sample complexity analysis of tests. Note that as [Algorithm 4](#), in order to test whether a vertex v is peeled off at iteration at most j it suffices to sample a c^{j-1}/d fraction of the stream and run recursive tests on neighbors of v found in that random sample. It is crucial for our analysis that the tests are sample efficient, as otherwise our algorithm would not gather sufficient information to approximate matching size from only m samples (or, from a single pass over a randomly ordered permutation stream – see [Section 9](#)). One might hope that the sample complexity of LEVEL- j -TEST is dominated by the samples that it accesses explicitly (as opposed to the ones contributed by recursive calls), and this turns out to be the case. The proof follows rather directly by induction, essentially exactly because lower level tests, say LEVEL- i -TEST for $i < j$, by the inductive hypothesis use a c^{i-1}/d fraction of the stream, and contribute c^{i-j-1} to the counter S maintained by the algorithm (see [Algorithm 4](#)). Since the algorithm terminates as soon as the counter reaches a small constant δ , the claim essentially follows, and holds deterministically for any stream of edges – the proof is given in [Section 4](#) below (see [Lemma 2](#)).

Sample complexity of approximating matching size. A question that remains to be addressed is how to actually use the level tests to approximate the maximum matching size. We employ the following natural approach: keep sampling edges of the graph using the stream of iid samples and testing the received edges (the way we process these samples is novel). In [\[KKS14\]](#), where a polylogarithmic approximation was achieved, the algorithm only needed to ascertain that at least one of the logarithmic classes (similar to the ones defined by our peeling algorithm) contains nontrivial edge mass, and could discard the others.

Since we aim to obtain a constant factor approximation, this would not be sufficient. Instead, our [Line 9](#) maintains a counter that it updates with *estimated weights* of the edges sampled from the stream. Specifically, an edge $e = (u, v)$ is declared to be a level j edge if both u and v pass all LEVEL- i -TEST tests with $i < j$ and at least one of them fails LEVEL- j -TEST – we refer to this procedure as EDGE-LEVEL-TEST(e) (see [Algorithm 3](#)). Such an edge contributes approximately c^{j-1}/d to a counter M' that estimates matching size. It turns out to be very helpful to think of approximating matching size as **a fraction of the stream length**, i.e., have $M' \in [0, 1]$ (see [Line 9](#) for the actual test and [Line 13](#) for the application inside [Line 9](#)). Our estimate is then the average of the weights of all sampled edges. We then need to argue that the variance of our estimate is small enough to ensure that we can get a constant multiplicative approximation without consuming more than m samples. This turns out to be a very natural variance calculation: the crucial observation is that whenever an edge e sampled from the stream is assigned weight c^{j-1}/d due to the outcomes of LEVEL- j -TEST on the endpoints (see [Line 13](#) of [Line 9](#)), the cost of testing it (in terms of samples consumed by recursive calls) is comparable to its contribution to the estimate. This implies that at most m samples are sufficient – the details are given in the proof of [Theorem 6](#).

4 Sample Complexity of [Algorithm 2](#)

We begin by analyzing the sample complexity of a single LEVEL- j -TEST test. After that, in [Theorem 6](#), we prove that this sample complexity suffices to estimate the matching size by using no more than m iid edge-samples.

4.1 Sample Complexity of Level Tests

Lemma 2. *For every $c \geq 1$, $\delta \leq 1/2$, and graph $G = (V, E)$, let τ_j be the maximum possible number of samples required by LEVEL- j -TEST defined in [Algorithm 3](#), for $j \in [1, J + 1]$. Then, with probability one we have:*

$$\tau_j \leq 2c^{j-1} \cdot \frac{m}{d}. \quad (4)$$

Proof. We prove this lemma by induction that [Eq. \(4\)](#) holds for each j .

Base of induction For $j = 1$ the bound [Eq. \(4\)](#) holds directly as

$$\tau_1 \leq \frac{m}{d},$$

by the definition of the algorithm.

Inductive step. Assume that Eq. (4) holds for j . We now show that Eq. (4) holds for $j + 1$ as well.

Consider any vertex $v \in V$ and LEVEL- $(j + 1)$ -TEST(v). Let α_i be the number of recursive LEVEL- i -TEST calls invoked during a worst case run of LEVEL- $(j + 1)$ -TEST(v) (that is when LEVEL- $(j + 1)$ -TEST(v) consumes τ_{j+1} samples). Then, τ_{j+1} can be upper-bounded as

$$\tau_{j+1} \leq c^j \cdot \frac{m}{d} + \sum_{i=1}^j \alpha_i \tau_i.$$

Moreover, from Eq. (4) and our inductive hypothesis, it holds

$$\begin{aligned} \tau_{j+1} &\leq c^j \cdot \frac{m}{d} + \sum_{i=1}^j \alpha_i \cdot 2c^{i-1} \cdot \frac{m}{d} \\ &= c^j \cdot \frac{m}{d} \cdot \left(1 + 2 \sum_{i=1}^j c^{i-1-j} \alpha_i \right). \end{aligned} \quad (5)$$

Our goal now is to upper-bound $\sum_{i \leq j} c^{i-1-j} \alpha_i$. During the execution, LEVEL- $(j + 1)$ -TEST maintains the variable S . Consider any time during LEVEL- $(j + 1)$ -TEST when a recursive call is made to LEVEL- i -TEST, for $i \geq 1$, in Line 8 of Algorithm 4. Immediately preceding this, in Line 9 of the previous iteration of the loop, the variable S would have been incremented by c^{i-1-j} . This happens α_i times for every $i \geq 1$. Consider now the state of the algorithm just before the *last* recursive call is made to a lower level test. By the time the last recursive call is made, S has been increased by $\sum_{i \leq j} c^{i-1-j} \alpha_i$ in total. However, just before the last recursive test is made, the algorithm *does not* exit to return FALSE in Line 10, meaning that $S < \delta$ at this point. Hence

$$\sum_{i \leq j} c^{i-1-j} \alpha_i \leq \delta.$$

This together with Eq. (5) leads to

$$\tau_{j+1} \leq c^j \cdot \frac{m}{d} \cdot (1 + 2\delta) \leq 2c^j \cdot \frac{m}{d},$$

as desired, where the last inequality follows by the assumption that $\delta \leq 1/2$. \square

4.2 Sample Complexity of IID-Peeling

Lemma 3. For every $c \geq 2$, $0 < \delta \leq 1/2$, and graph $G = (V, E)$, for any edge $e = (u, v) \in E$, if M_e is the output of an invocation of EDGE-LEVEL-TEST(e), then with probability one this invocation used at most $4M_e \cdot m$ edge-samples.

Proof. As before, let τ_j be the maximum possible number samples required by LEVEL- j -TEST. From Lemma 2 we have $\tau_j \leq 2c^{j-1} \cdot \frac{m}{d}$.

Let I be the last value of i , at which the algorithm EDGE-LEVEL-TEST(e) exits the while loop in Line 7 of Algorithm 3. Alternately if the algorithm exits in Line 8, let $I = J$. This means that the variable w has been incremented for all values of i from 1 to $I - 1$, making w equal to $\sum_{i=0}^{I-1} c^i/d$. On the other hand, in the worst case scenario, LEVEL- i -TEST has been called on both u and v for values of i from 1 to I . Therefore, the number of samples used by this invocation of EDGE-LEVEL-TEST(e) is at most

$$2 \sum_{i=1}^I \tau_i \leq 2 \sum_{i=1}^I \frac{2c^{i-1} \cdot m}{d} = 4m \cdot \sum_{i=0}^{I-1} \frac{c^i}{d} = 4M_e \cdot m.$$

\square

In Section 5, we show the following theorem.

Theorem 5. For sufficiently small $\delta > 0$ and large enough c the following holds. For a graph $G = (V, E)$ and an edge $e \in E$, let M_e denote the value returned by EDGE-LEVEL-TEST(e) (Algorithm 3). Then,

$$\sum_{e \in E} \mathbb{E}[M_e] = \Theta(MM(G)).$$

Given this, we now prove that our main algorithm outputs a constant factor approximation of the maximum matching size.

Theorem 6. *For sufficiently small $\delta > 0$ and large enough c the following holds. For a graph $G = (V, E)$, IID-PEELING (Algorithm 2) with probability $4/5$ outputs a constant factor approximation of $MM(G)$ by using at most m iid edge-samples.*

We now give the proof of the main algorithmic result of the paper:

Proof of Theorem 1 (first part): The proof of the first part of Theorem 1 now follows from Theorem 6 and the fact that IID-PEELING has recursion depth of $O(\log n)$, where each procedure in the recursion maintains $O(1)$ variables, hence requiring $O(\log n)$ bits of space. Therefore, the total memory is $O(\log^2 n)$. \square

Proof of Theorem 6: We first derive an upper bound on the number of edges that SAMPLE (see Algorithm 2) needs to test in order to obtain a constant factor approximation to maximum matching size with probability at least $9/10$, and then show that the number of iid samples that the corresponding edge tests use is bounded by m , as required. We define

$$\mu \stackrel{\text{def}}{=} \mathbb{E}_{e \sim U(E)} [M_e]$$

for convenience, where $U(E)$ is the uniform distribution on E . Now we have $\mu \cdot m = \Theta(MM(G))$ by Theorem 5. We now show that our algorithm obtains a multiplicative approximation to μ using at most m samples.

Upper bounding number of edge tests that suffice for multiplicative approximation of μ . We now analyze the number of edge-samples used by Algorithm 2. We first analyze the sample-complexity of method SAMPLE, and later of method IID-PEELING.

Let $E_t = \{e_1, \dots, e_t\}$ be a list of t iid edge-samples taken by SAMPLE (G, t) (Line 11 of Algorithm 2), where t is a parameter passed on Line 5 of IID-PEELING. We now show that if $t \geq 160/\mu$, then

$$\mathbb{P} \left[\left| \frac{1}{t} \sum_{i=1}^t M_{e_i} - \mu \right| > \mu/2 \right] < 1/10, \quad (6)$$

where the probability is over the choice of E_t as well as over the randomness involved in sampling M_{e_i} for $i = 1, \dots, t$.

We prove Eq. (6) by Chebyshev's inequality. For $e \sim U(E)$ we have that $M_e \leq \sum_{i=0}^J c^i/d \leq 2c^J/d \leq 2/c$, since $J = \lfloor \log_c d \rfloor - 1$.

$$\text{Var} [M_e] \leq \mathbb{E} [M_e^2] \leq 2/c \cdot \mathbb{E} [M_e] = 2\mu/c.$$

We thus get by Chebyshev's inequality, using the fact that $\text{Var} \left[\frac{1}{t} \sum_{i=1}^t M_{e_i} \right] = \frac{1}{t} \text{Var} [M_e]$, that

$$\mathbb{P} \left[\left| \frac{1}{t} \sum_{i=1}^t M_{e_i} - \mu \right| \geq \mu/2 \right] \leq \text{Var} [M_e] / (t \cdot (\mu/2)^2) \leq \frac{8}{tc\mu},$$

and hence Eq. (6) holds for any $t \geq 160/c\mu$, as required.

Upper bounding total sample complexity of edge tests. It remains to upper bound the total number of iid edge-samples consumed by the edge tests. For an edge $e \in E$ let Z_e denote the fraction of our overall budget of m samples needed to finish the invocation EDGE-LEVEL-TEST(e) (equivalently, let $m \cdot Z_e$ be the number of samples taken). By Lemma 3 we have

$$Z_e \leq 4M_e. \quad (7)$$

Since e_1, \dots, e_t are uniform samples from the edges of the graph, we have

$$\mathbb{E} \left[\sum_{i=1}^t Z_{e_i} \right] = \sum_{i=1}^t \mathbb{E} [Z_{e_i}] \leq \sum_{i=1}^t 4\mathbb{E} [M_{e_i}] = 4\mu t.$$

Hence, t executions of EDGE-LEVEL-TEST(e_i) in expectation require at most $4\mu t \cdot m$ edges. Each of these invocations of EDGE-LEVEL-TEST is performed by SAMPLE(G, t). In addition to invoking

EDGE-LEVEL-TEST, $\text{SAMPLE}(G, t)$ samples t edges on [Line 12](#) to obtain e_1, \dots, e_t . Therefore, the total sample complexity of $\text{SAMPLE}(G, t)$ in expectation is

$$4t\mu \cdot m + t \leq 5t\mu \cdot m.$$

In the last inequality we upper-bounded t by $t\mu \cdot m$. This upper-bound holds as $M_e \geq 1/d$ ([Line 2](#) of [Algorithm 3](#)), so $\mu \geq 1/d \geq 1/n$ and hence $\mu \cdot m \geq 1$. Also, here we used that $m \geq n$ by [Remark 1](#).

Upper bounding sample complexity of IID-Peeling. Finally, we bound the expected sample complexity of IID-PEELING from [Algorithm 2](#). IID-PEELING terminates only when it runs out of samples, but we consider it to have had enough samples if it completes the call of SAMPLE with a parameter $t \geq 160/c\mu$. (In particular let t^* be the smallest power of 2 greater than $160\mu/c$; this is the value of t we aim for, as all values of t are powers of 2. $t^* \leq 320/c\mu$.) The expected number of iid samples required is at most

$$5\mu \cdot m + 10\mu \cdot m + \dots + 5t^*\mu \cdot m \leq 10t^*\mu \cdot m \leq 3200m/c$$

By Markov's inequality we thus have that IID-PEELING the call of SAMPLE for $t = t^*$ within $3200m/c \leq m$ samples with probability at least $9/10$. (Let $c \geq 32000$.) Conditioned on this, the algorithm succeeds with probability at least $9/10$, therefore it succeeds with overall probability at least $4/5$, as claimed. \square

5 Correctness of [Algorithm 2](#) (proof of [Theorem 5](#))

The main result of this section is the following theorem.

Theorem 5. *For sufficiently small $\delta > 0$ and large enough c the following holds. For a graph $G = (V, E)$ and an edge $e \in E$, let M_e denote the value returned by $\text{EDGE-LEVEL-TEST}(e)$ ([Algorithm 3](#)). Then,*

$$\sum_{e \in E} \mathbb{E}[M_e] = \Theta(\text{MM}(G)).$$

Overview of techniques. The main challenge in proving this claim is that the function $\text{LEVEL-}j\text{-TEST}$ ([Algorithm 3](#)) potentially returns different outputs for the same vertex on different runs. Moreover, the outputs of two independent invocations could differ with constant probability. The propagation of such unstable estimates over $\Theta(\log d)$ (possibly $\Theta(\log n)$) peeling steps could potentially result in a significant error in the estimation of $\text{MM}(G)$. The previous works avoid this issue by loosening the approximation factor to $O(\text{poly log } n)$, which allows a union bound over all vertices of the graph. We cannot afford this. Instead, we control error propagation by showing that contributions of lower levels to higher level tests have progressively smaller variance, and hence the total error stays bounded. This is nontrivial to show, however, since none of the involved random variables concentrate particularly well around their expectation – our main tool for dealing with this is the Oversampling Lemma ([Lemma 1](#)) below.

An orthogonal source of difficulty stems from the fact that without polylogarithmic oversampling $\text{LEVEL-}j\text{-TEST}$'s are inherently noisy, and might misclassify nontrivial fractions of vertices across $\omega(1)$ levels. Informally, to cope with this issue we charge the cost of the vertices that the $\text{LEVEL-}j\text{-TEST}$'s misclassify to the rest of the vertices. More precisely, we show that although the algorithm might misclassify the layer that a vertex belongs to for a constant fraction of the vertices, the amount of resulting error of the edges adjacent to such vertices in the estimation of the matching is low compared to the contribution of the rest of the edges. This enables us to bound the total error cost by a small constant with respect to the size of the maximum matching.

5.1 Definitions and Preliminaries

In order to establish [Theorem 5](#) we analyze the propagation of the error introduced by misclassification in a new setting, not strictly corresponding to any of our algorithms. We first consider a hypothetical set of tests. Namely, for each vertex $v \in V$, we consider executions of $\text{LEVEL-}i\text{-TEST}(v)$ for $i = 1, 2, \dots$ until a test fails. Then we use these outcomes to categorize all the vertices of G into levels; these levels loosely corresponding to those of [Algorithm 1](#). We then define a set of edge weights based on these levels,

$\widehat{M} : E \rightarrow \mathbb{R}$. Most of the section is then devoted to showing that this is close to a maximum fractional matching in the sense that

$$\sum_{e \in E} \mathbb{E} [\widehat{M}(e)] = \Theta(\text{MM}(G)).$$

Defining a (nearly optimal) vertex cover. Our main tool in upper bounding the size of the maximum matching in G is a carefully defined (random) nested sequence of $V = \widehat{V}_0 \supseteq \widehat{V}_1 \supseteq \dots \supseteq \widehat{V}_{J+1}$. As we show later in [Section 5.3](#) (see the proof of [Lemma 6](#)), the set

$$C \stackrel{\text{def}}{=} \left\{ v \in V \mid \mathbb{P} \left[v \in \widehat{V}_{T+1} \right] \leq 1 - \frac{1}{3c^2} \right\} \quad (8)$$

turns out to be a nearly optimal deterministic vertex cover in G .

Since our algorithm is essentially an approximate and randomized version of a peeling algorithm for approximating matching size ([Algorithm 1](#)), our analysis is naturally guided by a sequence of random subsets \widehat{V}_j of the vertex set V . These subsets loosely correspond to the set of vertices in V that survived j rounds of the peeling process. The sets are defined by the following process performed **independently** by all vertices of G . Every $v \in V$ keeps running $\text{LEVEL-}j\text{-TEST}(v)$ ([Algorithm 3](#)) for $j = 0, 1, 2, \dots$, while the tests return TRUE. Let $\widehat{L}(v)$ denote the largest j such that the corresponding test returned TRUE. We then let

$$\widehat{V}_j \stackrel{\text{def}}{=} \left\{ v \in V : \widehat{L}(v) \geq j \right\}, \text{ for } j = 0, \dots, J+1. \quad (9)$$

Note that the variables $\widehat{L}(v)$ are independent, and $V = \widehat{V}_0 \supseteq \widehat{V}_1 \supseteq \widehat{V}_2 \supseteq \dots \supseteq \widehat{V}_{J+1}$ with probability 1. Also, $\widehat{L}(v)$ values can be defined via the sets \widehat{V}_j as

$$\widehat{L}(v) \stackrel{\text{def}}{=} \text{maximum } i \text{ such that } v \in \widehat{V}_i. \quad (10)$$

Recall that \widehat{V}_0 equals V , so for any v there is at least one i such that $v \in \widehat{V}_i$, and hence the definition [Eq. \(10\)](#) is valid.

Our proof crucially relies on a delicate analysis of the probability of a given vertex v belonging to \widehat{V}_{j+1} conditioned on v belonging to \widehat{V}_j for various $j = 0, \dots, J$. To analyze such events we define, for every $v \in V$, the random variable $S_j(v)$ as follows. Let $r = c^j m/d$ and let e_1, \dots, e_r be i.i.d. uniform samples (with repetition) from the edge set E of G . Let $i_1 \leq \dots \leq i_Q$, where $Q \leq r$, be the subset of indices corresponding to edges in the sample that are incident on v , i.e., $e_{i_a} = (v, w_a)$ for every $a = 1, \dots, Q$ (note that Q is a random variable). For every $a = 1, \dots, Q$ let $L_a \sim \widehat{L}(w_a)$ be independent samples from the distribution $\widehat{L}(w_a)$ defined above. We now let

$$S_j(v) \stackrel{\text{def}}{=} \sum_{a=1}^Q \sum_{i=0}^{\min\{L_a, j\}} c^{i-j}. \quad (11)$$

In other words, $S_j(v)$ is simply the value of the variable S during the call $\text{LEVEL-}(j+1)\text{-TEST}$ ([Algorithm 3](#)). In particular, we have

$$\widehat{V}_{j+1} = \left\{ v \in \widehat{V}_j \mid S_j(v) < \delta \right\}. \quad (12)$$

Remark 3. Note that [Eq. \(11\)](#) and [Eq. \(12\)](#), together with $\widehat{V}_0 \stackrel{\text{def}}{=} V$ define \widehat{V}_j recursively (and in a non-cyclic manner). Indeed, \widehat{V}_{j+1} depends on $S_j(v)$ which depends on $\min\{L_a, j\}$ whose distribution is defined by $\widehat{V}_i, i \leq j$ only.

We also define $S_j(v)$ in a compact way and use that definition in the proofs extensively

$$S_j(v) \stackrel{\text{def}}{=} \sum_{\substack{k=1 \\ e_k \sim U(E)}}^{c^j m/d} \mathbb{1}[v \in e_k] \sum_{i=0}^{\min(L(e_k \setminus v), j)} c^{i-j}. \quad (13)$$

Remark 4. Note that here $L(w)$ is a random variable independently sampled from the distribution of $\widehat{L}(w)$, similarly to L_a in [Eq. \(11\)](#).

Remark 5. Here $U(E)$ denotes the uniform distribution over E . Also we denote by $v \in e$ the fact that e is adjacent to v , where v is a vertex and e is an edge. In this case we denote the other endpoint of e by $e \setminus v$. We use this notation heavily throughout the analysis.

Defining the fractional pseudo-matching \widehat{M} . We now define a (random) fractional assignment \widehat{M} of mass to the edges of G that our analysis will be based on: we will later show (see [Lemma 4](#) in [Section 5.2](#)) that \widehat{M} is close to some matching of G . For every edge $e = (u, v) \in E$ we let

$$\widehat{M}(u, v) \stackrel{\text{def}}{=} \sum_{i=0}^{\min\{\widehat{L}(u), \widehat{L}(v)\}} c^i / d, \quad (14)$$

and define the size $|\widehat{M}|$ of the pseudo-matching \widehat{M} to be the summation of its fractional matching mass along all the edges:

$$|\widehat{M}| \stackrel{\text{def}}{=} \sum_{e \in E} \widehat{M}(e). \quad (15)$$

We note that \widehat{M} is a random variable, and we show later in [Section 5.2](#) (see [Lemmas 4](#) and [6](#)) that $\mathbb{E}[|\widehat{M}|]$ is a constant factor approximation to the size of a maximum matching in G . The following natural auxiliary definitions will be useful.

For every vertex $v \in V$, we let

$$\widehat{M}(v) \stackrel{\text{def}}{=} \sum_{w \in N(v)} \sum_{i=0}^{\min\{\widehat{L}(v), \widehat{L}(w)\}} c^i / d = \sum_{w \in N(v)} \widehat{M}(v, w), \quad (16)$$

denote amount of fractional mass incident to v . Similarly, we let

$$\widehat{M}_j(v) \stackrel{\text{def}}{=} \sum_{w \in N(v)} \sum_{i=0}^{\min\{\widehat{L}(w), j\}} c^i / d, \quad (17)$$

denote the amount of fractional mass contributed to v by its neighbors conditioned on $\widehat{L}(v) = j$.

In the upcoming section we will prove upper and lower bounds on \widehat{M} to prove that it is within a constant factor of the matching number of G . Supposing we have this result, it is easy to deduce [Theorem 5](#).

Proof of [Theorem 5](#). Consider the marginal distribution of $\widehat{M}(e)$ for some edge $e = (u, v)$. $\widehat{M}(e)$ is essentially the result of running independent LEVEL- i -TEST's on u and v to determine at which level either vertex is peeled off, and then updating the edge weight in accordance with EDGE-LEVEL-TEST. Thus the marginal distribution of $\widehat{M}(e)$ is identical to that of EDGE-LEVEL-TEST (e). Therefore

$$\sum_{e \in E} \mathbb{E}[M_e] = \sum_{e \in E} \mathbb{E}[\widehat{M}(e)] = \mathbb{E}[|\widehat{M}|] = \Theta(|M|),$$

thus proving the theorem. □

Next we state two observations that follow directly from the definitions above.

Observation 1. For any vertex v and any j the following holds:

- (a) $\widehat{M}_j(v)$ depends only on vertices other than v , therefore it is independent of $\widehat{L}(v)$.
- (b) $\widehat{M}(v) = \widehat{M}_{\widehat{L}(v)}(v)$.
- (c) $\mathbb{E}[\widehat{M}_j(v)] = \mathbb{E}[S_j(v)] = \sum_{w \in N(v)} \sum_{i=0}^j \mathbb{P}[w \in \widehat{V}_i] \cdot c^i / d$.

Observation 2. For any vertex v and any j , it holds that

$$(c + 1) \cdot \mathbb{E}[S_j(v)] \geq \mathbb{E}[S_{j+1}(v)],$$

where $S_j(v)$ is defined in [Eq. \(13\)](#).

Proof. This follows directly from the formula of $\mathbb{E}[S_j(v)]$ in [Observation 1 c](#):

$$\begin{aligned}
\mathbb{E}[S_{j+1}(v)] - \mathbb{E}[S_j(v)] &= \sum_{w \in N(v)} \mathbb{P}[v \in \widehat{V}_{j+1}] \cdot c^{j+1}/d \\
&\leq \sum_{w \in N(v)} \mathbb{P}[v \in \widehat{V}_j] \cdot c^{j+1}/d \\
&\leq c \cdot \sum_{i=0}^j \sum_{w \in N(v)} \mathbb{P}[w \in \widehat{V}_i] \cdot c^i/d \\
&= c \cdot \mathbb{E}[S_j(v)],
\end{aligned}$$

which implies the observation. \square

We will use the following well-known concentration inequalities.

Theorem 7 (Chernoff bound). *Let X_1, \dots, X_k be independent random variables taking values in $[0, a]$. Let $X \stackrel{\text{def}}{=} \sum_{i=1}^k X_i$. Then, the following inequalities hold:*

(a) *For any $\delta \in [0, 1]$ if $\mathbb{E}[X] \leq U$ we have*

$$\mathbb{P}[X \geq (1 + \delta)U] \leq \exp(-\delta^2 U / (3a)).$$

(b) *For any $\delta > 1$ if $\mathbb{E}[X] \leq U$ we have*

$$\mathbb{P}[X \geq (1 + \delta)U] \leq \exp(-(\delta + 1) \log(\delta + 1) U / 3a) \leq \exp(-\delta U / (3a)).$$

(c) *For any $\delta > 0$ if $\mathbb{E}[X] \geq U$ we have*

$$\mathbb{P}[X \leq (1 - \delta)U] \leq \exp(-\delta^2 U / (2a)).$$

5.2 Lower Bound: Constructing a Near-Optimal Matching from \widehat{M}

As the main result of this section, we prove a lower bound on $|\widehat{M}|$. Namely, we show that the mass defined by \widehat{M} is at most a constant factor larger than the size of a maximum matching.

Lemma 4 (Lower-bound on $|\widehat{M}|$). *Let $c \geq 20$ and $0 < \delta \leq 1$. For any graph $G = (V, E)$, there exists a feasible fractional matching M such that $\mathbb{E}[|\widehat{M}|] = O(|M|)$, where \widehat{M} is defined in [Eq. \(15\)](#).*

The following lemma, which is the main technical result that we use in the proof of [Lemma 4](#), shows that if the pseudo-matching weight $\widehat{M}(v)$ is large at some level, then it is very likely that v does not pass the next vertex tests. This lemma is crucial in proving an upper-bound on the estimated size of the matching.

Lemma 5 (Concentration on $\widehat{M}(v)$). *For any vertex v , any $j > 0$, and constants c and x such that $c \geq 20$ and $x \geq 100c \log c$, we have:*

$$\mathbb{P}[\widehat{M}(v) \geq x \wedge \widehat{L}(v) = j + 1] \leq \frac{10c^2}{x^2} \cdot \mathbb{E}[\widehat{M}(v) \cdot \mathbb{1}[\widehat{L}(v) = j]]. \quad (18)$$

Where $\widehat{L}(v)$ and $\widehat{M}(v)$ are defined in [Eq. \(10\)](#) and [Eq. \(16\)](#), respectively.

The full proof of [Lemma 5](#) is deferred to [Appendix A](#). Next, we show certain basic properties of \widehat{M} that are derived from [Lemma 5](#).

Corollary 1. *For any vertex v , $c \geq 20$, and $x \geq 100c \log c$, we have:*

$$\mathbb{P}[\widehat{M}(v) \geq x] \leq \frac{10c^2}{x^2} \cdot \mathbb{E}[\widehat{M}(v)],$$

where $\widehat{M}(v)$ is defined in [Eq. \(16\)](#).

Proof. We simply sum [Eq. \(18\)](#) from [Lemma 5](#) over $j = 1, \dots, J$. The term corresponding to $\widehat{L}(v) = J+1$ is missing from the right hand side, but this only makes the inequality stronger. The term corresponding to $\widehat{L}(v) = 1$ is missing from the left hand side, but the corresponding probability is in fact 0. Indeed, if $\widehat{L}(v) = 1$, each edge adjacent to v has only $1/d$ weight on it, and there are at most d such edges. $\widehat{M}(v) \leq 1 < x$. \square

The following corollary enables us to bound the contribution of the high degree vertices to the fractional matching. This is a key ingredient to proving a lower bound on the estimated matching size. Namely, in the proof of [Lemma 4](#) we ignore all the vertices such that $\widehat{M}(v) \geq \lambda$, where we think of λ being some large constant. Ignoring those vertices reduces the matching mass contained in \widehat{M} . The following corollary essentially bounds the matching mass lost in this process.

Corollary 2. *For any vertex v , $c \geq 20$, and $\lambda \geq 100c^2$:*

$$\mathbb{E} \left[\widehat{M}(v) \cdot \mathbb{1} \left[\widehat{M}(v) \geq \lambda \right] \right] \leq \frac{1}{4} \mathbb{E} \left[\widehat{M}(v) \right],$$

where $\widehat{M}(v)$ is defined in [Eq. \(16\)](#).

Proof. We prove this corollary by applying [Corollary 1](#) for different values of x .

$$\begin{aligned} \mathbb{E} \left[\widehat{M}(v) \cdot \mathbb{1} \left[\widehat{M}(v) \geq \lambda \right] \right] &\leq \sum_{i=0}^{\infty} \lambda 2^{i+1} \mathbb{P} \left[\widehat{M}(v) \geq \lambda 2^i \right] \\ &\stackrel{\text{by Corollary 1}}{\leq} \sum_{i=0}^{\infty} \lambda 2^{i+1} \cdot \frac{10c^2}{\lambda^2 2^{2i}} \mathbb{E} \left[\widehat{M}(v) \right] \\ &= \frac{20c^2 \mathbb{E} \left[\widehat{M}(v) \right]}{\lambda} \sum_{i=0}^{\infty} 2^{-i} \\ &\leq \frac{1}{4} \mathbb{E} \left[\widehat{M}(v) \right] \end{aligned}$$

Since $\lambda > 80c^2$. \square

We now have all necessary tools to prove [Lemma 4](#).

Proof of Lemma 4. We prove this lemma by constructing such a matching, M . To that end, let $\lambda \stackrel{\text{def}}{=} 100c^2$. If $\widehat{M}(v) \geq \lambda$ we say that v is a *violating* vertex. Similarly, any edge adjacent to at least one violating vertex we also call violating. We add all the non-violating edges of \widehat{M} to M . Moreover, we reduce the weight of each edge in fractional matching M by the factor $1/\lambda$. Then, M is a feasible fractional matching since the summation of the weights of the edges connected to each vertex is at most one.

We now compute the expected weight of M . Recall that, in any fractional matching, the summation of the weights of the edges is half the summation of the weights of the vertices, since each edge has two endpoints. Therefore

$$\begin{aligned} \mathbb{E} [|M|] &= \frac{1}{2} \sum_{v \in V} \mathbb{E} [|M(v)|] \\ &= \frac{1}{2\lambda} \sum_{v \in V} \left(\mathbb{E} \left[\widehat{M}(v) \right] - 2 \mathbb{E} \left[\widehat{M}(v) \cdot \mathbb{1} \left[\widehat{M}(v) \geq \lambda \right] \right] \right) \\ &\stackrel{\text{by Corollary 2}}{\geq} \frac{1}{2\lambda} \sum_{v \in V} \frac{1}{2} \mathbb{E} \left[\widehat{M}(v) \right] \\ &\geq \frac{1}{4\lambda} \mathbb{E} \left[\widehat{M} \right]. \end{aligned}$$

Since λ is a constant by definition, this completes the proof. \square

5.3 Upper bound: Constructing a Near-Optimal Vertex Cover

Lemma 4 essentially states that the size of the pseudo-matching \widehat{M} that our algorithm (implicitly) constructs does not exceed by more than a constant factor the size of a maximum matching. By the next lemma, we also provide a lower-bound on the size of \widehat{M} . Namely, we show that the size of \widehat{M} is only by a constant factor smaller than a vertex cover of the input graph. Since from duality theory the size of a vertex cover is an upper-bound on the size of a maximum matching, the next lemma together with **Lemma 4** shows that \widehat{M} is a $\Theta(1)$ -approximate maximum matching.

Lemma 6 (Upper-bound on $|\widehat{M}|$). *For sufficiently small $\delta > 0$ and large enough c the following holds. For any graph $G = (V, E)$, there exists a feasible vertex cover C such that $\mathbb{E}[|\widehat{M}|] = \Omega(|C|)$, where $|\widehat{M}|$ is defined in [Eq. \(15\)](#).*

In our proof, we choose C to be the set of vertices that with constant probability do not pass to the very last level, i.e., to the level $T + 1$. (C corresponds to the set that we defined in [Eq. \(8\)](#).) Then, the high-level approach in our proof of **Lemma 6** is to choose a vertex $v \in C$ and show that with constant probability the matching incident to v is sufficiently large.

Observe that v is added to C if the algorithm estimates that at some level the matching weight of v is at least δ . So, in light of our approach, we aim to show that if v has expected matching at least δ then it is unlikely that its *actual* matching weight is much smaller than δ in a realization. So, for every j independently we first upper-bound the probability that v gets added to C at level j if its actual matching mass by level j is much smaller than δ . To provide this type of upper-bound across all the levels simultaneously, one standard approach would be to take a union bound over all the levels. Unfortunately, the union bound would result in a loose upper-bound for our needs.

Instead we will show that over the levels the algorithm's estimates of the weight adjacent to a specific vertex becomes more and more accurate (since it takes more and more samples). This allows us to show that the likelihood of misclassifying a vertex by peeling it too early is small even across potentially $\Omega(\log n)$ levels. The matching weight that the algorithm estimates while testing level j can be decomposed into two parts:

- A_j – the weight coming from the neighbors up to level $j - 1$.
- B_j – the weight coming from the neighbors that pass to level j .

Now, compared to the weight estimate while performing the test for level $j - 1$, to obtain A_j the algorithm performs c times more tests and takes their average. This means that A_j is a more precise version of a test the algorithm has already done. Since this is the case, we can amortize the error coming from estimating A_j to the tests that the algorithms has already applied, and bound only the error coming from estimating B_j . This observation enables us to provide a more precise analysis than just applying a union bound.

The next lemma makes formal our discussion of relating A_j and the weight estimate the algorithm performs while testing level $j - 1$. In this lemma, it is instructive to think of \bar{X} as of A_j , of each X as a single instance of level $j - 1$ testing (that corresponds to $A_{j-1} + B_{j-1}$), and of Y_k as the test performed on a single sampled edge.

Lemma 1 (Oversampling lemma). *For sufficiently small $\delta > 0$ and large enough c the following holds. Let $X = \sum_{k=1}^K Y_k$ be a sum of independent random variables Y_k taking values in $[0, 1]$, and $\bar{X} \stackrel{\text{def}}{=} \frac{1}{c} \sum_{i=1}^c X_i$ where X_i are iid copies of X . If $\mathbb{E}[X] \leq \delta/3$ and $\mathbb{P}[X \geq \delta] = p$, then $\mathbb{P}[\bar{X} \geq \delta] \leq p/2$.*

This lemma formalizes the intuition that given a random variable X , taking the average of many independent copies gives a more accurate estimate of the mean than X itself, and it should therefore be less likely to exceed a threshold significantly above the mean. In the general case however, this is not true. Indeed consider a variable X such that $X > c\delta$ with some extremely small probability and $X < \delta$ the rest of the time. In this case if even one instance of the independent samples exceeds δ then the average of all the samples (\bar{X}) will as well; therefore the probability of exceeding δ actually increases. To be able to prove the lemma we need to use an additional characteristic of X : namely that it is the independent sum of bounded variables, and therefore it concentrates reasonably well around its expectation. This characteristic will hold for the particular random variables to which we want to apply the oversampling lemma in the proof of **Lemma 6**, specifically A_j .

We present the proof of this lemma in [Appendix B](#). We are now ready to prove **Lemma 6**.

Proof of Lemma 6. Define the following set of vertices

$$C \stackrel{\text{def}}{=} \left\{ v \in V \mid \mathbb{P} \left[v \in \widehat{V}_{J+1} \right] \leq 1 - \frac{1}{3c^2} \right\}. \quad (19)$$

Notice that this set is deterministic and does not depend on the outcome of \widehat{V}_{J+1} , only its distribution.

First we prove that C is indeed a vertex cover. Suppose toward contradiction that both $u, v \notin C$ for some edge $e = (u, v)$. Suppose u has already made it to \widehat{V}_J . Then in the course of deciding whether u makes it further into \widehat{V}_{J+1} we take $mc^J/n \geq m/c^2$ iid edge-samples. Hence the probability of sampling the edge e at least one of those times is

$$1 - \left(1 - \frac{1}{m}\right)^{mc^J/d} \geq 1 - \left(1 - \frac{1}{m}\right)^{m/c^2} \geq \frac{1}{2c^2},$$

for $c \geq 1$. If e is sampled, then the probability that v makes it into \widehat{V}_J is at least $1 - 1/(3c^2)$ which is strictly greater than $2/3$, since $v \notin C$. This means that with more than $1/(3c^2)$ probability u would fail at level J even if it made it that far and therefore must be in C . By contradiction C is a vertex cover.

Consider a vertex $v \in C$. Our goal is to show that the fractional matching adjacent to v is small with small probability. To that end, we upper-bound the probability that the matching adjacent to v is less than γ , for some positive constant $\gamma \ll \delta$. Indeed we will see that $\gamma \leq 1/(12c^2)$ works.

Let j^* be the largest $j \in [0, J+1]$ such that $\mathbb{E}[S_{j^*}(v)] < \gamma$. We prove that the combined probability of v failing any test up to j^* is at most $1/(6c^2)$. Before proving this, let us explain the rest of the proof, assuming we get such guarantee.

Note that the random variable $\widehat{L}(v)$ is independent from the sequence of random variables $\widehat{M}_j(v)$ and $\widehat{M}(v) = \widehat{M}_{\widehat{L}(v)}(v)$ by **Item a** and **Item b** of **Observation 1**. Thus the expected size of \widehat{M} is sufficiently large (at least γ) conditioned on the event that $\widehat{L} > j^*$. This does not happen in two cases. Either v fails a test at a level lower than or equal to j^* , or v fails no test, but $j^* = T+1$, so v reaches the last level but expected size of the incident matching $\widehat{M}(v)$ is too small in expectation regardless. The first event is bounded by $1/(6c^2)$ by the above guarantee; the second event is bounded by $1 - 1/(3c^2)$ since $v \in C$. Therefore v must reach a level greater than j^* with probability at least $1/(6c^2)$. Note that this also shows that $j^* < T+1$, which is not clear from definition. Hence,

$$\begin{aligned} \mathbb{E}[\widehat{M}(v)] &\geq \sum_{j=0}^{T+1} \mathbb{P}[\widehat{L}(v) = j] \mathbb{E}[\widehat{M}_j(v)] \\ &\geq \sum_{j=j^*+1}^{T+1} \mathbb{P}[\widehat{L}(v) = j] \mathbb{E}[\widehat{M}_{j^*+1}(v)] \\ &\geq \mathbb{P}[\widehat{L}(v) > j^*] \cdot \gamma > \frac{\gamma}{6c^2}, \end{aligned}$$

and consequently by linearity of expectation follows $\mathbb{E}[\widehat{M}] \geq \frac{\gamma}{6c^2} \cdot |C|$.

In the rest of the proof we upper-bound the probability that v fails before or at the j^{th} level.

$$\begin{aligned} \mathbb{P}[\widehat{L}(v) \leq j^*] &= \sum_{j=0}^{j^*} \mathbb{P}[\widehat{L}(v) = j] \\ &\leq \sum_{j=0}^{j^*} \mathbb{P}[\widehat{L}(v) = j \mid v \in \widehat{V}_j] \\ &= \sum_{j=0}^{j^*} \mathbb{P}[S_j(v) \geq \delta]. \end{aligned} \quad (20)$$

$$= \sum_{j=0}^{j^*} \mathbb{P}[S_j(v) \geq \delta]. \quad (21)$$

Eq. (20) follows from the fact that $\widehat{L}(v) = j$ implies that $v \in \widehat{V}_j$ (while the other direction does not

necessarily hold). We next rewrite $\mathbb{P}[S_j(v) \geq \delta]$. By definition [Eq. \(13\)](#), we have

$$\mathbb{P}[S_j(v) \geq \delta] = \mathbb{P} \left[\sum_{\substack{k=1 \\ e_k \sim U(E)}}^{mc^j/d} \mathbb{1}[v \in e_k] \sum_{i=0}^{\min(L(e \setminus v), j)} c^{i-j} \geq \delta \right]$$

We split the contribution to $S_j(v)$ into two parts: the weight coming from the sampled edges incident to v up to level $j-1$ (defined as the sum A_j below); and, to contribution coming from the sampled edges incident to v that passed to level j (corresponding to the sum B_j below). More precisely, for each j , the sums A_j and B_j are defined as follows

$$\begin{aligned} A_j &\stackrel{\text{def}}{=} \sum_{\substack{k=1 \\ e_k \sim U_E}}^{mc^j/d} \mathbb{1}[v \in e] \sum_{i=0}^{\min(L(e \setminus v), (j-1))} c^{i-j}, \\ B_j &\stackrel{\text{def}}{=} \sum_{\substack{k=1 \\ e_k \sim U_E}}^{mc^j/d} \mathbb{1}[v \in e] \mathbb{1}[L(e \setminus v) \geq j], \end{aligned}$$

where we use the notation $v \in e$ for v being an endpoint of e ; in this case $e \setminus v$ denotes the other endpoint.

Observe that if $S_j(v) \geq \delta$, then either $A_j \geq \delta$, or $A_j < \delta$ and $B_j \geq \delta - A_j > 0$. If $B_j > 0$, it means that at least one edge incident to v was sampled and it passed to level j . This sample contributes 1 to B_j , and hence if $B_j > 0$ it implies $B_j \geq 1$. Then we can write

$$\mathbb{P}[S_j(v) \geq \delta] = \mathbb{P}[A_j + B_j \geq \delta] = \mathbb{P}[A_j \geq \delta \vee B_j \geq 1] \leq \alpha_j + \beta_j, \quad (22)$$

where we define $\alpha_j \stackrel{\text{def}}{=} \mathbb{P}[A_j \geq \delta]$ and $\beta_j \stackrel{\text{def}}{=} \mathbb{P}[B_j \geq 1]$. To upper-bound $\mathbb{P}[S_j(v) \geq \delta]$, we upper-bound α_j 's and β_j 's separately.

Upper-bounding α_j and β_j . We first upper-bound α_j by $(\alpha_{j-1} + \beta_{j-1})/2$ by applying [Lemma 1](#). We begin by defining X , Y and \bar{X} that correspond to the setup of [Lemma 1](#). Let Y_k be the following random variable

$$Y = \mathbb{1}[v \in e] \sum_{i=0}^{\min(L(e \setminus v), (j-1))} c^{i-(j-1)},$$

where e is an edge sampled uniformly at random. Then, $A_{j-1} + B_{j-1} = \sum_{k=1}^{mc^{j-1}/n} Y_k$, where Y_k is a copy of Y . Let $X = A_{j-1} + B_{j-1}$ and $\bar{X} = A_j$. Observe that $\bar{X} = \sum_{i=1}^c X_i/c$. Then, for $j > 0$, it holds

$$\begin{aligned} \alpha_j &= \mathbb{P}[A_j \geq \delta] \\ &\stackrel{\text{by Lemma 1}}{\leq} \mathbb{P}[A_{j-1} + B_{j-1} \geq \delta] / 2 \\ &\stackrel{\text{by Eq. (22)}}{\leq} (\alpha_{j-1} + \beta_{j-1})/2. \end{aligned} \quad (23)$$

In the case of $j = 0$, we have $\alpha_0 = 0$.

Applying [Section 5.3](#) recursively, we derive

$$\alpha_j \leq \sum_{i=0}^{j-1} \frac{1}{2^{j-i}} \beta_i. \quad (24)$$

Now we upper-bound the sum of β_j 's by using Markov's inequality: since for every $j = 0, \dots, j^*$

$$\mathbb{E}[B_j] = \sum_{\substack{k=1 \\ e_k \sim U_E}}^{mc^j/d} \mathbb{E}[\mathbb{1}[v \in e] \mathbb{1}[L(e \setminus v) \geq j]] = \mathbb{E}[|N_j(v)|] c^j/d,$$

we get

$$\begin{aligned}
\sum_{j=0}^{j^*} \beta_j &\leq \sum_{j=0}^{j^*} \mathbb{E} \left[|N(v) \cap \widehat{V}_j| \right] \cdot c^j / d \\
&= \sum_{w \in N(v)} \sum_{j=0}^{j^*} \mathbb{P} \left[w \in \widehat{V}_j \right] \cdot c^j / d \\
&= \mathbb{E} [S_{j^*}(v)] \\
&\leq \gamma
\end{aligned} \tag{25}$$

Finalizing the proof. Combining the above inequalities together, we derive

$$\begin{aligned}
\mathbb{P} \left[\widehat{L}(v) \leq j^* \right] &\stackrel{\text{from Eq. (21) and Eq. (22)}}{\leq} \sum_{j=0}^{j^*} (\alpha_j + \beta_j) \\
&\stackrel{\text{from Eq. (24)}}{=} \sum_{j=0}^{j^*} \left(\beta_j + \sum_{i=0}^{j-1} \frac{1}{2^{j-i}} \beta_i \right) \\
&\leq \sum_{j=0}^{j^*} 2\beta_j \\
&\stackrel{\text{from Eq. (25)}}{\leq} 2\gamma \\
&\leq 1/(6c^2),
\end{aligned}$$

for $\gamma \leq 1/(12c^2)$, as desired \square

6 LCA

Local computational algorithms (or LCA's) have been introduced in [RTVX11] and have since been studied extensively, particularly in context of graph algorithms [ARVX12, MRVX12, MV13, EMR14, LRY17, GU19]. The LCA model is designed to deal with algorithms on massive data, such that both the input *and the output* are too large to store in memory. Instead we deal with both via query access. In the setting of graphs, we have access to a graph G via queries which can return the neighbors of a particular vertex. We must then construct our output (in our case a constant factor maximum matching) implicitly, such that we can answer queries about it consistently. That is, for any edge we must be able to say whether or not it is in the matching and for any vertex we must be able to say whether or not any edge adjacent to it is in the matching.

In this section we show that our approach, detailed in the previous sections, can be implemented in the LCA model as well. Specifically, we prove [Theorem 2](#).

Theorem 2. *Let G be a graph with n vertices and maximum degree d . Then there exists a random matching M , such that $\mathbb{E}[|M|] = \Theta(\text{MM}(G))$, and an algorithm that with high probability:*

- *Given an edge e of G , the algorithm reports whether e is in M or not by using $O(d \log n)$ queries.*
- *Given a vertex v of G , the algorithm reports whether v is in M or not by using $O(d \log n)$ queries.*

Moreover, this algorithm can be executed by using $O(d \log^3 n)$ bits of memory.

Remark 6. *It can also be shown with more careful analysis that if $d = O((n/\log n)^{1/4})$ then $|M| = \Theta(\text{MM}(G))$ with high probability. A proof sketch of this claim can be found in [Section 6.7](#).*

The proof of this theorem is organized as follows. First, in [Section 6.3](#), we state our LCA algorithms. Then, in [Section 6.4](#) we analyze the query complexity of the provided algorithms, essentially proving the two bullets of [Theorem 2](#). In [Section 6.5](#) we show that the matching fixed by our algorithms is $\Theta(1)$ approximation of $\text{MM}(G)$. In [Section 6.6](#) we discuss about the memory requirement of our approach and the implementation of consistent randomness. These conclusions are combined in [Section 6.7](#) into a proof of [Theorem 2](#).

6.1 Overview of Our Approach

Our main LCA algorithm simulates [Algorithm 3](#), i.e., it simulates methods `LEVEL-(j + 1)-TEST` and `EDGE-LEVEL-TEST` provided in [Section 3.2](#). However, instead of taking $c^j m/d$ random edge-samples from the entire graph (as done on [Line 4](#) of `LEVEL-j-TEST` (v)), we first sample the number of edges D incident to a given vertex v , where D is drawn from binomial distribution $B(c^j m/d, d(v)/m)$. Then, we query D random neighbors of v . This simulation is given as [Algorithm 5](#).

In our analysis, we tie the fractional matching weight of an edge to the query complexity. Essentially, we show that a matching weight w of an edge is computed by performing $O(w \cdot d)$ queries. As we will see, this allows us to transform a fractional to an integral matching by using only $O(d \log n)$ queries per an edge.

From fractional to integral matching. Given an edge $e = \{u, v\}$, `LCA-EDGE-LEVEL-TEST` outputs the fractional matching weight w_e of e . However, our goal is to implement an oracle corresponding to an integral matching. To that end, we round those fractional to 0/1 weights as follows. First, each edge e is marked with probability $w_e/10\lambda$, for some large constant λ , specifically the constant from [Corollary 2](#), the result of which we will be relying on heavily. Then, each edge that is the only one marked in its neighborhood is added to the matching. We show that in expectation this rounding procedure outputs a $\Theta(1)$ -approximate maximum matching.

Consistency of the oracles. Our oracles are randomized. Nevertheless, they are designed in such a way that if the oracle is invoked on an edge e multiple times, each time it provides the same output. We first present our algorithms by ignoring this property, and then in [Section 6.6](#) describe how to obtain these consistent outputs.

6.2 Related Work

Parnas and Ron [[PR07](#)] initiated the question of estimating the minimum vertex cover and the maximum matching size in sublinear time. First, they propose a general reduction scheme that takes a k -round distributed algorithm and design an algorithm that for graphs of maximum degree d has $O(d^k)$ query complexity. Second, they show how to instantiate this reduction with some known distributed algorithms to estimate the maximum matching size with a constant multiplicative and ϵn additive factor with $d^{O(\log(d/\epsilon))}$ queries. An algorithm with better dependence on ϵ , but worse dependence on d , was developed by Nguyen and Onak [[NO08](#)] who showed how to obtain the same approximation result by using $2^{O(d)}/\epsilon^2$ queries. Significantly stronger query complexity, i.e., $O(d^4/\epsilon^2)$, was obtained by Yoshida et al. [[YYI09](#)]. Both [[NO08](#)] and [[YYI09](#)] analyze the following randomized greedy algorithm: choose a random permutation π of the edges; visit the edges sequentially in the order as given by π ; add the current edge to matching if none of its incident edge is already in the matching. We analyze this algorithm in great detail in [Section 7](#). As their main result, assuming that the edges are sorted with respect to π , [[YYI09](#)] show that this randomized greedy algorithm in expectation requires $O(d)$ queries to output whether a given edge is in the matching fixed by π or not. When the edges are not sorted, their algorithm in expectation requires $O(d^2)$ queries to simulate the randomized greedy algorithm.

The result of [[YYI09](#)] was improved by Onak et al. [[ORRR12](#)], who showed how to estimate the maximum matching size by using $\tilde{O}(\bar{d} \cdot \text{poly}(1/\epsilon))$ queries, where \bar{d} is the average degree of the graph. Instead of querying randomly chosen edges, [[ORRR12](#)] query randomly chosen vertices, which in turn allows them to choose a sample of $\Theta(1/\epsilon^2)$ vertices rather than a sample of $\Theta(d^2/\epsilon^2)$ edges. Then, given a vertex v , the approach of [[ORRR12](#)] calls the randomized greedy algorithm on (some of) the edges incident to v . By adapting the analysis of [[YYI09](#)], [[ORRR12](#)] are able to show that the expected vertex-query complexity of their algorithm is $O(d)$. As noted, these results estimate the maximum matching size up to a constant multiplicative and ϵn additive factor. Hence, assuming that the graph does not have isolated vertices, to turn this additive to a constant multiplicative factor it suffices to set $\epsilon = 1/d$.

To approximate the maximum matching size, the aforementioned results design oracles that given an edge e outputs whether e is in some fixed $\Theta(1)$ -approximate maximum matching, e.g., a maximal matching, or not. Then, they query a small number of edges chosen randomly, and use the oracle-outputs on those edges to estimate the matching size. Concerning the query complexity, the usual strategy here is to show that running the oracle on most of the edges requires a “small” number of queries, leading to the desired query complexity in expectation. When those oracles are queried on arbitrary chosen edge, their query complexity might be significantly higher than the complexity needed to estimate the

maximum matching size. We devote [Section 7](#) to analyzing the randomized greedy algorithm mentioned above, and show that in some cases it requires at least $\Omega(d^{2-\epsilon})$ queries, for arbitrary small constant ϵ . This is in stark contrast with the expected query complexity of $O(d)$.

Recently, [\[GU19\]](#) showed that there exists an oracle that given an arbitrary chosen vertex v outputs whether v is in some fixed maximal independent set or not by performing $d^{O(\log \log d)} \cdot \text{poly log } n$ queries, which improves on the prior work obtaining $d^{O(\text{poly log } d)} \cdot \text{poly log } n$ complexity [\[RTVX11, ARVX12, LRY17, Gha16\]](#). When this oracle is applied to the line graph, then it reports whether a given edge is in a fixed maximal matching or not.

6.3 Algorithms

[Algorithm 5](#) is an LCA simulation of [Algorithm 3](#). In LCA-LEVEL- $(j+1)$ -TEST, that is an LCA simulation of LEVEL- $(j+1)$ -TEST, we can not choose a random edge-sample from the entire graph as it is done on [Line 4](#) of LEVEL- $(j+1)$ -TEST. So, instead, we first sample from the distribution corresponding to how many of those random edge-samples will be incident to a given vertex v . In this way we obtain a number D (see [Line 3](#) of LCA-LEVEL- $(j+1)$ -TEST). Then, our algorithm samples D random edges incident to v and performs computation on them. Note that this is equivalent to the iid variant, as we would ignore all edges not adjacent to v .

Algorithm 5 This is an LCA simulation of [Algorithm 3](#) for a graph of maximum degree d . Given an edge e , this algorithm returns a fractional matching-weight of e .

```

1: procedure LCA-EDGE-LEVEL-TEST( $e = (u, v)$ )
2:   for  $i = 1$  to  $J + 1$  do
3:     if LCA-LEVEL- $i$ -TEST( $u$ ) and LCA-LEVEL- $i$ -TEST( $v$ ) then
4:        $w \leftarrow w + c^i / n$ 
5:     else
6:       return  $w$ 
7:   return  $w$ 

```

Algorithm 6 This is an LCA simulation of [Algorithm 4](#). Given a vertex v , this algorithm returns true if it belongs to level $j+1$ and false otherwise.

```

1: procedure LCA-LEVEL- $(j+1)$ -TEST( $v$ )
2:    $S \leftarrow 0$ 
3:   Sample  $D$  from binomial distribution  $B(c^j \cdot \frac{m}{d}, d(v)/m)$ 
4:   for  $k = 1$  to  $D$  do
5:     Let  $e = \{v, w\}$  be a random edge incident to  $v$ .
6:      $i \leftarrow 0$ 
7:     while  $i \leq j$  and LCA-LEVEL- $i$ -TEST( $w$ ) do  $\triangleright$  LCA-LEVEL-0-TEST( $w$ ) returns TRUE by definition.
8:        $S \leftarrow S + c^{i-j}$ 
9:       if  $S \geq \delta$  then
10:        return FALSE
11:        $i \leftarrow i + 1$ 
12:   return TRUE

```

We now build on LCA-EDGE-LEVEL-TEST and LCA-LEVEL- $(j+1)$ -TEST to design an oracle that for some fixed $\Theta(1)$ -approximate maximum matching returns whether a given edge is in this matching or not.

Our final LCA algorithm is ORACLE-EDGE (see [Algorithm 8](#)). Given an edge $e = \{u, v\}$, this algorithm reports whether e is in some fixed matching M or not. This matching M is fixed for all the queries. ORACLE-EDGE performs rounding of a fractional matching as outlined above. As a helper method, it uses MATCHING-CANDIDATE that as a parameter gets an edge e , and returns 0 and 1 randomly chosen with respect to LCA-EDGE-LEVEL-TEST(e). We note that if MATCHING-CANDIDATE(e) returns 1 that it *does not* necessarily mean that e is included in M . It only means that e is a candidate for being added to M . In fact, e is added to M if e is the only matching candidate in its 1-hop neighborhood.

Algorithm 7 Given an edge e , this method rounds fractional matching mass returned by $\text{LCA-EDGE-LEVEL-TEST}(e)$ (that is first scaled by 10λ) to an integral one.

```

1: procedure MATCHING-CANDIDATE( $e$ )
2:   Let  $\lambda$  be the constant from Corollary 2.
3:   Let  $X_e$  be 1 with probability  $\text{LCA-EDGE-LEVEL-TEST}(e)/(10\lambda)$ , and be 0 otherwise.
4:   return  $X_e$ 

```

Algorithm 8 An oracle that returns TRUE if a given edge e is in the matching, and returns FALSE otherwise.

```

1: procedure ORACLE-EDGE( $e = (u, v)$ )
2:    $X_e \leftarrow \text{MATCHING-CANDIDATE}(e)$ 
3:   if  $X_e = 0$  then
4:     return FALSE
5:   for each edge  $e'$  incident to  $e$  do
6:      $X_{e'} \leftarrow \text{MATCHING-CANDIDATE}(e')$ 
7:     if  $X_{e'} = 1$  then
8:       return FALSE  $\triangleright e$  is in the matching only if it is the only candidate in its neighborhood
9:   return TRUE

```

We also design a vertex-oracle (see [Algorithm 9](#)) that for a given vertex v reports whether an edge incident to v is in the matching M or not. This oracle iterates over all the edges incident to v , and on each invokes MATCHING-CANDIDATE. If any invocation of MATCHING-CANDIDATE(e) returns 1 it means that there is at least one matching candidate in the neighborhood of v . Recall that having two or more matching candidates in the neighborhood of v would result in none of those candidates being added to M . Hence, v is in M only if ORACLE-EDGE(e) return TRUE. Otherwise, v is not in M .

Algorithm 9 An oracle that returns TRUE if a given vertex v is in the matching, and returns FALSE otherwise.

```

1: procedure ORACLE-VERTEX( $v$ )
2:   for each edge  $e$  incident to  $v$  do
3:      $X_e \leftarrow \text{MATCHING-CANDIDATE}(e)$ 
4:     if  $X_e = 1$  then
5:       return ORACLE-EDGE( $e$ )
6:   return FALSE  $\triangleright$  None of the edges incident to  $v$  is a matching candidate.

```

6.4 Query Complexity

We begin by analyzing the query complexity of LCA-LEVEL- j -TEST. Statement of the next lemma is an adapted version of [Lemma 2](#) to LCA.

Lemma 7. *For every $c \geq 2$, $0 < \delta \leq 1/2$, and graph $G = (V, E)$ with the maximum degree at most d , let τ_j be the maximum query complexity of LCA-LEVEL- j -TEST defined in [Algorithm 5](#), for $j \in [1, J + 1]$. Then, with probability one we have:*

$$\tau_j \leq 2c^{j-1}. \quad (26)$$

Proof. We prove this lemma by induction that [Eq. \(26\)](#) holds for each j . This proof follows the lines of [Lemma 2](#).

Base of induction For $j = 1$ the bound [Eq. \(26\)](#) holds directly as

$$\tau_1 \leq 1$$

by definition of the algorithm. Indeed, as soon as we sample a single neighbor the algorithm returns FALSE.

Inductive step. Assume that Eq. (26) holds for j . We now show that Eq. (26) holds for $j+1$ as well.

Consider any vertex $v \in V$ and LCA-LEVEL- $(j+1)$ -TEST(v). Let α_i be the number of recursive LCA-LEVEL- i -TEST calls invoked. Then, τ_{j+1} can be upper-bounded as

$$\tau_{j+1} \leq \alpha_0 + \sum_{i=1}^j \alpha_i \tau_i.$$

For $\delta < 1$, we have $\alpha_0 \leq c^j$. Moreover, from Eq. (26) and our inductive hypothesis, it holds

$$\begin{aligned} \tau_{j+1} &\leq c^j + \sum_{i=1}^j \alpha_i \cdot 2c^{i-1} \\ &= c^j \cdot \left(1 + 2 \sum_{i=1}^j c^{i-1-j} \alpha_i \right). \end{aligned}$$

The rest of the proof now follows in the same way as in Lemma 2. \square

The next claim is an LCA variant of Lemma 3. The proof is almost identical.

Lemma 8. *For every $c \geq 2$, $0 < \delta \leq 1/2$, and graph $G = (V, E)$, for any edge $e = (u, v) \in E$, if M_e is the output of an invocation of LCA-EDGE-LEVEL-TEST(e), then with probability one this invocation used at most $4M_e \cdot d$ queries.*

Proof. As before, let τ_j be the maximum possible number of samples required by LCA-LEVEL- j -TEST. From Lemma 7 we have $\tau_j \leq 2c^{j-1}$.

Let I be the last value of i , at which the algorithm LCA-EDGE-LEVEL-TEST(e) exits the while loop in Line 6. Alternately if the algorithm exits in Line 7, let $I = J$. This means that the variable w has been incremented for all values of i from 0 to $I-1$, making w equal to $\sum_{i=0}^{I-1} c^i/n$. On the other hand, in the worst case scenario, LCA-LEVEL- i -TEST has been called on both u and v for values of i from 1 to I . Therefore, the number of queries used by this invocation of LCA-EDGE-LEVEL-TEST (e) is at most

$$2 \sum_{i=1}^I \tau_i \leq 2 \sum_{i=1}^I 2c^{i-1} = 4 \sum_{i=0}^{I-1} c^i = 4M_e \cdot d,$$

where we used the fact that $I \leq T$. \square

We are now ready to prove the query complexity of our oracles, and at the same time prove the query complexity part of Theorem 2.

Lemma 9. *For every $c \geq 2$, $0 < \delta \leq 1/2$ and $G = (V, E)$ be a graph of maximum degree at most d . Then, for any edge $e \in E$ and with probability at least $1 - n^{-5}$, ORACLE-EDGE(e) requires $O(d \log n)$ queries.*

Proof. Each loop of ORACLE-EDGE($e = \{u, v\}$) queries one of the edges incident to u or v . Hence, obtaining these incident edges takes $O(d)$ queries.

Let E' be the set of edges on which LCA-EDGE-LEVEL-TEST is called from MATCHING-CANDIDATE as a result of running ORACLE-EDGE(e) of Line 6. Let W be the sum of outputs of LCA-EDGE-LEVEL-TEST on these edges. First, by Lemma 8, the query complexity of all the tests performed on Line 6 is $O(W \cdot d)$. Second, following that $X_{e'}$ is obtained by rounding LCA-EDGE-LEVEL-TEST(e')/(10 λ) in MATCHING-CANDIDATE, we have that

$$\mathbb{E} \left[\sum_{e' \in E'} X_{e'} \right] = \frac{W}{10\lambda}.$$

As soon as $X_{e'} = 1$ for any $e' \in E'$, the algorithm terminates and returns FALSE on Line 8. Since $X_{e'}$ s are independent random variables, by Chernoff bound (Theorem 7 (b)), with probability at least $1 - n^{-5}$ we have $W/(10\lambda) \leq 20 \log n$. Hence, we conclude that the loop of ORACLE-EDGE requires $O(d \log n)$ queries with probability at least $1 - n^{-5}$. \square

Lemma 10. *For every $c \geq 2$, $0 < \delta \leq 1/2$ and $G = (V, E)$ be a graph of maximum degree at most d . Then, for any vertex $v \in V$, with high probability ORACLE-VERTEX(v) requires $O(d \log n)$ queries.*

Proof. Each loop of `ORACLE-VERTEX`(v) queries one of the edges incident to v . Hence, obtaining these incident edges takes $O(d)$ queries.

Following the same arguments as in [Lemma 9](#) we have that with probability at least $1 - n^{-5}$ the total query complexity of all invocations of `MATCHING-CANDIDATE` on [Line 3](#) is $O(d \log n)$. In addition, the algorithm invokes `ORACLE-EDGE` at most once. Hence, from [Lemma 9](#), the total query complexity of `ORACLE-VERTEX` is $O(d \log n)$ with high probability. \square

6.5 Approximation Guarantee

We now prove that outputs of `ORACLE-EDGE` correspond to a $\Theta(1)$ -approximate maximum matching of G .

Lemma 11. *For sufficiently small $\delta > 0$ and large enough c the following holds. Let $G = (V, E)$ be a graph whose maximum degree is at most d . Let M be the set of edges for which `ORACLE-EDGE` outputs TRUE. Then, M is a matching and $\mathbb{E}[|M|] = \Theta(\text{MM}(G))$.*

Proof. We first argue that M is a matching. For an edge e , let X_e be the output of `MATCHING-CANDIDATE`(e). The edge e is a candidate to be a matching edge (but e will not necessarily be added to the matching M) only if $X_e = 1$. Hence, if $X_e = 0$, then the `ORACLE-EDGE`(e) returns FALSE on [Line 4](#). Otherwise, [Line 7](#) verifies whether $X_{e'} = 1$ for any e' incident to e . If for at least one such edge $X_{e'} = 1$, then e and e' are both candidates to be matching edges. However, adding both e and e' would lead to a collision and hence not a valid matching. This collision is resolved by adding neither e nor e' to M , implying that the set of edges added to M indeed forms a matching.

We now argue that $\mathbb{E}[|M|] = \Theta(\text{MM}(G))$. We use the fact that `LCA-LEVEL- j -TEST` and `LCA-EDGE-LEVEL-TEST` are perfect simulations of `LEVEL- j -TEST` and `EDGE-LEVEL-TEST` respectively. Therefore, the fractional pseudo-matching defined by the return values of `LCA-EDGE-LEVEL-TEST` is identical to \widehat{M} defined in [Eq. \(16\)](#) of [Section 5](#), and obeys the same properties.

If a matching weight incident to a vertex v is at least λ , (recall that λ is the constant from [Corollary 2](#)), then we say v is *heavy*. An edge is incident to a heavy vertex if at least one of its endpoints is heavy. Hence, by [Corollary 2](#), at most $1/2$ of the matching mass is incident to heavy vertices in expectation. Let e be an edge neither of whose endpoints are heavy. The edge e is in the matching if $X_e = 1$ and $X_{e'} = 0$ for each other edge e' incident to e . Recall that $X_{e'} = 1$ with probability `LCA-EDGE-LEVEL-TEST`(e')/(10 λ). Since no endpoint of e is heavy, it implies that $X_{e'} = 0$ for every e' incident to e with probability

$$\begin{aligned} & \prod_{e' \in \delta(e)} \left(1 - \frac{\text{LCA-EDGE-LEVEL-TEST}(e')}{10\lambda} \right) \\ & \geq 1 - \sum_{e' \in \delta(e)} \frac{\text{LCA-EDGE-LEVEL-TEST}(e')}{10\lambda} \\ & \geq 1 - \frac{2\lambda}{10\lambda} = \frac{4}{5}. \end{aligned}$$

Hence, if e is not incident to a heavy vertex, `ORACLE-EDGE`(e) returns TRUE and hence adds e to M with probability at least $\frac{4}{5} \text{LCA-EDGE-LEVEL-TEST}(e)/(10\lambda)$. Also, by [Theorem 5](#) and our discussion about the matching mass of the edges incident to heavy vertices, we have that

$$\sum_{e = \{u, v\} : u \text{ and } v \text{ are not heavy}} \text{LCA-EDGE-LEVEL-TEST}(e) = \Theta(\text{MM}(G)).$$

This together with the fact that λ is a constant implies that $\mathbb{E}[|M|] = \Theta(\text{MM}(G))$, as desired. \square

6.6 Memory Complexity and Consistent Oracles

In this section we discuss about the memory requirement of our LCA algorithms, and also describe how to obtain oracles that provide consistent outputs.

Each of our methods maintains $O(1)$ variables. Observe that by definition the depth of our recursive method `LCA-LEVEL- $(j+1)$ -TEST` is $O(\log d)$. Hence, the total number of variables that our algorithms have to maintain at any point of execution is $O(\log d)$ requiring $O(\log d \cdot \log n)$ bits.

The way we described `ORACLE-EDGE` above, when it is invoked with an edge e two times, it could potentially provide different outputs. This is the case for two reasons: D on [Line 3](#) and e on [Line 5](#) of

LCA-LEVEL- $(j+1)$ -TEST are chosen randomly, and this choice may vary from iteration to iteration; and the random variable X_e drawn by MATCHING-CANDIDATE may be different in different invocations of MATCHING-CANDIDATE(e). It is tempting to resolve this by memorizing the output of ORACLE-EDGE(e) and the corresponding invocations of LCA-LEVEL- $(j+1)$ -TEST (as we will see shortly, not all the invocations of LCA-LEVEL- $(j+1)$ -TEST should be memorized). Then, when ORACLE-EDGE(e) is invoked the next time we simply output the stored value. Unfortunately, in this way our algorithm would potentially require $\Theta(Q)$ memory to execute Q oracle queries, while our goal is to implement the oracles using $O(d \cdot \text{poly log } n)$ memory. To that end, instead of memorizing outputs, we will use k -wise independent hash functions.

Lemma 12. *For $k, b, N \in \mathbb{N}$, there is a hash family \mathcal{H} of k -wise independent hash functions such that all $h \in \mathcal{H}$ maps $\{0, 1\}^N$ to $\{0, 1\}^b$. Any hash function in the family \mathcal{H} can be stored using $O(k \cdot (N + b))$ bits of space.*

Lemma 13 (Nisan's PRG, [Nis90]). *For every $s, R > 0$, there exists a PRG that given a seed of $s \log R$ truly random bits can produce $\Omega(R)$ pseudo random bits such that any algorithm of space at most s requiring $O(R)$ random bits will succeed using the pseudo random bits with probability at least $2^{-\Omega(s)}$. Each bit can be extracted in $O(s \log R)$ time.*

Randomness and consistency in the algorithms. Before we explain how to apply Lemmas 12 and 13 to obtain consistency of our methods, we recall a part of our analysis and recall how LCA-LEVEL- j -TEST is used in our algorithms. Our analysis crucially depends on Corollary 2 (see Lemma 11) that provides a statement about heavy vertices, i.e., about vertices whose incident edges have the matching mass of at least λ . Heavy vertices are defined with respect to \widehat{M} , while \widehat{M} variables are defined with respect to \widehat{L} (see Eqs. (14) and (16)). Finally, $\widehat{L}(v)$ is defined/sampled by repeatedly invoking LCA-LEVEL- j -TEST(v) for $j = 0, 1, 2, \dots$ while the tests return TRUE (see the discussion above Eq. (9)). LCA-EDGE-LEVEL-TEST(e) effectively samples \widehat{L} for its endpoints by Line 3. Since we provide our analysis by assuming that $\widehat{L}(v)$ is defined consistently, the invocations of LCA-LEVEL- j -TEST directly from LCA-EDGE-LEVEL-TEST have to be consistent (Line 3). We call this invocation as *the top invocation* of LCA-LEVEL- j -TEST.

Based on this discussion, the top invocation of LCA-LEVEL- j -TEST(v) will use a fixed sequence $B(v)$ of random bits to execute this call (including all the recursive invocations of LCA-LEVEL- j -TEST performed therein). We emphasize that the output of the top invocation of LCA-LEVEL- j -TEST(v) does not have to be the same as the output of LCA-LEVEL- j -TEST(v) invoked recursively via some other top invocation. That is, for example, if a top level invocation LCA-LEVEL- j -TEST(v) recursively calls LCA-LEVEL- i -TEST(w) (for some w neighbor of v and some $i < j$) then the recursive call LCA-LEVEL- i -TEST(w) continues to use $B(v)$ for its randomness.

Next, recall that by Lemma 7 LCA-LEVEL- j -TEST has query complexity $O(d)$ even at the highest level. Each query is a random edge-sample. Also, recursively via Line 7, each query requires sampling $O(d)$ times variable D on Line 3 of LCA-LEVEL- j -TEST. Hence, the total number of random bits required for the execution of LCA-LEVEL- (j) -TEST is $O(d \log^2 n)$. However, the test itself uses only $\log(d) \cdot \log(n)$ space, therefore, by Lemma 13, a seed of $O(\log^3 n)$ truly random bits suffice, that is $|B(v)| = O(\log^3 n)$. Furthermore, our runtime is only increased by a factor of $\log^3 n$ from using Nisan's PRG.

The rounding of the fractional matching by MATCHING-CANDIDATE(e) should also be consistent across queries and should never depend on where the call to MATCHING-CANDIDATE(e) came from, (unlike with LCA-LEVEL- j -TEST). To that end, each edge e should have its own random seed $B(e)$ to use in MATCHING-CANDIDATE(e). Here $|B(e)| = O(\log n)$ suffices.

Independence. We have shown that our LCA algorithm would work correctly if all seeds, $B(v)$ and $B(e)$ for $v \in V$ and $e \in E$, were truly independent. However, storing $\Omega(m)$ random seeds would be extremely inefficient. Instead we will use an k -wise independent hash family, \mathcal{H} , mapping $V \cup E$ to $\{0, 1\}^{O(\log^3 n)}$. That is we will sample a hash function $h \in \mathcal{H}$ up front and calculate $B(v) = h(v)$ during queries. Consider an edge $e \in E$. Note that whether or not e is in the integral matching depends only on the levels of vertices in the 1-hop neighborhood of e , as well as the rounding of edges in its 1-hop neighborhood. This is $O(d)$ vertices and edges in total, and so an $O(d)$ -wise independent hash family mapping $V \cup E$ to $\{0, 1\}^{O(\log^3 n)}$ with uniform marginal distributions on each vertex and edge would suffice to guarantee that e is in the integral matching with exactly the same probability as in the truly

independent case. This shows that $\mathbb{E}[|M|] = \text{MM}(G)$ would still hold. By [Lemma 12](#) such a family exists, and any element of it can be stored in space $O(d \log^3 n)$ space.

6.7 Proof of [Theorem 2](#)

Proof of [Theorem 2](#). We are now ready to prove the main result of this section. In [Section 6.3](#) we provided two oracles, ORACLE-EDGE and ORACLE-VERTEX. [Lemmas 9](#) and [10](#) show that these oracles have the desired query complexity. [Lemma 11](#) proves that ORACLE-EDGE outputs a $\Theta(1)$ -approximate maximum matching. Observe that ORACLE-VERTEX is consistent with ORACLE-EDGE. That is, for any vertex v , ORACLE-VERTEX(v) output TRUE iff there is an edge incident to v for which ORACLE-EDGE(e) outputs true. This implies that the outputs of ORACLE-VERTEX also correspond to a $\Theta(1)$ -approximate maximum matching. Finally, in [Section 6.6](#) we discussed how these oracles can be implemented by using space $O(d \log^3 n)$. \square

Proof sketch of [Remark 6](#). We observe that the random variable $|M|$ concentrates around its expectation since it is the sum of many bounded variables:

$$|M| = \sum_{e \in E} \mathbb{1}(e \in M).$$

Although these variables are not independent, their dependence graph has bounded degree, as observed in [Section 6.6](#) under the heading **Independence**. Indeed, for any specific edge $e = (u, v) \in E$, the variable $\mathbb{1}(e \in M)$ depends on the levels of the vertices in the one-hop neighborhood of e , as well as the level and rounding of edges in the one-hop neighborhood of e . Overall, random bits that influence the rounding of e include $B(w)$ and $B(f)$ for all vertices w and edges f in the neighborhood of e . If e and e' are at least distance 3 away, they are completely independent (disregarding the dependences introduced by the hash functions we use), therefore the dependence graph of the variables $\mathbb{1}(e \in M)$ has maximum degree $d' = O(d^3)$.

Theorem 1. of [[Pem01](#)] states that when the independence graph of Bernoulli variables X_i has degree bounded by d' , then

$$\mathbb{P} \left[\sum_i X_i \geq (1 - \epsilon)\mu \right] \leq \frac{4(d' + 1)}{\epsilon} \exp(-\mu\epsilon^2/2(d' + 1)),$$

where $\mu = \mathbb{E}[\sum_i X_i]$. By applying this with $\epsilon = 1/2$, $\mu = O(\text{MM}(G)) = O(n/d)$, $d' = O(d^3)$ and $d = O((n/\log n)^{1/4})$, we get that indeed $|M|$ is at least half of its expectation. It is also not hard to see that the same bound follows even if the variables $\mathbb{1}(e \in M)$ for edges more than distance 2 away are merely $\log n$ -wise independent instead of being truly independent.

7 Lower Bound of $\tilde{\Omega}(d^2)$ for Simulation of Randomized Greedy

In this section we analyze the result of Yoshida et al. [[YYI09](#)] for constructing a constant fraction approximate maximum matching in the LOCAL model. It was proven in [[YYI09](#)] that one can return whether some edge is in a maximal matching or not in time only $\Theta(d)$ when expectation is taken over both the randomness of the algorithm *and the choice of edge*. If it could be proven that this (or even a slightly weaker) bound holds for a worst case edge, the algorithm could be simply transformed into an LCA algorithm more efficient than the one we present in [Section 6](#). However, we proceed to prove that this is not the case.

The algorithm we consider is a simulation of the greedy algorithm for maximal matching in LCA. Given a graph $G = (V, E)$ and a permutation π of E , a natural way to define a maximal matching of G with respect to π is as follows: process the edges of E in the ordering as given by π ; when edge e is processed, add e to the matching if none of its incident edges has been already added. Motivated by this greedy approach, Yoshida et al. proposed and analyzed algorithm YYI-MAXIMAL-MATCHING (see [Algorithm 10](#)) that tests whether a given edge e is in the greedy maximal matching defined with respect to π .

Algorithm 10 Implementation of the greedy algorithm for maximal matching in LCA.

```

1: procedure YYI-MAXIMAL-MATCHING ( $e, \pi$ )
2:   for  $f \in \delta(e)$  such that  $f$  precedes  $e$ , in order of  $\pi$  do       $\triangleright \delta(e)$  is the edge-neighbourhood of  $e$ .
3:     if YYI-MAXIMAL-MATCHING( $f, \pi$ ) returns TRUE then
4:       return FALSE
5:   return TRUE

```

Pictorially, YYI-MAXIMAL-MATCHING can be viewed as a process of walking along neighboring edges (as defined via the recursive calls), and hence exploring the graph adaptively based on π . Yoshida et al. showed that the size of this exploration graph of YYI-MAXIMAL-MATCHING(e, π) is in expectation at most d , where the expectation is taken over all the starting edges e and all possible permutations π . It remained an open question whether it was necessary to take the expectation over the starting edge. If the size of the exploration tree could be shown to be $O(d)$ (or $O(d \log n)$) for even the worst case edge, this would yield an extremely efficient LCA algorithm for approximate maximum matching.

However, the main result of the section is that this is not the case:

Theorem 3. *There exists an absolute constant $b > 0$ such that for every n , $d \in [5, \exp(b\sqrt{\log n})]$ and $\epsilon \in [1/d, 1/2]$ there exists a graph G with n vertices and maximum degree $d + 1$, and an edge e such that running YYI-MAXIMAL-MATCHING(e, π) from [Algorithm 10](#) results in an exploration tree of size at least*

$$\frac{1}{8} \cdot \epsilon \cdot \left(\frac{d}{2}\right)^{2-\epsilon},$$

in expectation.

Notice that potentially $d = \exp(c\sqrt{\log n}) \gg \log n$. Therefore the $O(d \log n)$ bound that we achieve in [Theorem 2](#) is a factor $O(\frac{\exp(c\sqrt{\log n})}{\log n})$ better.

Overview of Our Approach We first construct a simple infinite tree ([Definition 1](#)): a tree in which each vertex has exactly d children with one extra special edge connected to the root. Then we prove that the number of queries made by YYI-MAXIMAL-MATCHING is bigger than d for the special edge. Afterwards, we extend the graph by merging the end points of the special edges of ϵd independent copies of these trees which creates another infinite tree. In this tree, beside the root that has ϵd children, the rest of vertices has d children ([Definition 3](#)). We also add an edge to the root and show that YYI-MAXIMAL-MATCHING uses almost d^2 queries for this edge. Infinite trees ease the computations due to the fact that each subtree is isomorphic to the main tree. In [Section 7.2](#), we carefully analyze the probability and variance of YYI-MAXIMAL-MATCHING reaching high depths and show that it is unlikely that it passes depth $O(\log n)$. Later, we use that to truncate the tree after depth $O(\log n)$. This enables us to get the graph with desired bounds in [Section 7.3](#). Notice that, throughout this section for the sake of simplicity, we assume that ϵd is an integer.

7.1 Lower Bound for Infinite Graphs

We begin by analyzing the behavior of YYI-MAXIMAL-MATCHING on infinite d -regular trees. We implement the random permutation π by assigning to each edge e a rank $r(e)$ chosen independently and uniformly at random from the interval $[0, 1]$. Edges are then implicitly ordered by increasing rank. Then, [Line 2](#) of [Algorithm 10](#) can be thought of as being “**for** $f \in \delta(e)$ such that $r(f) < r(e)$, in increasing order of rank **do**”. The behavior of the algorithm on any edge can be described by the two functions $p_e(\lambda)$ and $t_e(\lambda)$, where $\lambda \in [0, 1]$. The function $p_e(\lambda)$ denotes the probability that e is in the matching, given only that $r(e) = \lambda$. Therefore, $p_e(\lambda) \in [0, 1]$ and $p_e(0) = 1$. The function $t_e(\lambda)$ denotes the expected size of the exploration tree when exploring from e , given only that $r(e) = \lambda$. Therefore, we have $t_e(\lambda) \geq 1$, with equality only if $\lambda = 0$.

Definition 1 (Graph H^d , see [Fig. 1](#) for illustration). *For an integer $d \geq 1$, the graph H^d is defined as an infinite d -regular tree rooted in an edge $e_0 = (u_0, v_0)$. Let u_0 have no neighbor other than v_0 , and let v_0 have d neighbors other than u_0 . In general, let all vertices other than u_0 have $d + 1$ neighbors. For an edge e , level of e is defined as its distance from e_0 and denoted by $\ell(e)$. In particular, $\ell(e_0) = 0$ and the level of any other edge adjacent to v_0 is 1. Every edge $e = \{u, v\} \neq e_0$ has exactly $2d$ edge-neighbors (d incident to v and d incident to u); d of these neighbors have a higher level than e , we call them e ’s*

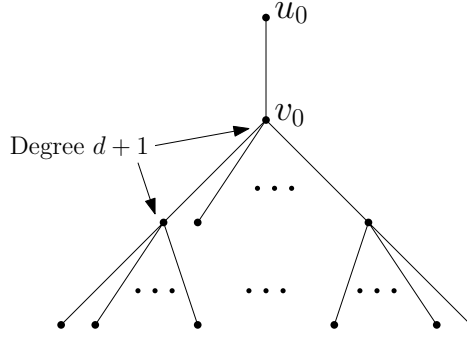


Figure 1: Construction of H^d .

children; $d - 1$ have equal level to e , we call them e 's siblings; exactly 1 has lower level than e , we call it e 's parent.

Definition 2 (Graph H_e). For every edge $e \in H^d$ let H_e be the set of edges whose unique path to e_0 goes through e (including e itself).

In the rest of this section, we analyze the behavior of $\text{YYI-MAXIMAL-MATCHING}(e_0, \cdot)$. Specifically, we calculate the expectation and the variance of its size, and upper-bound its depth over the randomness of the ranks. It will be convenient to consider a slightly more efficient version of [Algorithm 10](#), one in which the algorithm memorizes the results of queries across the recursive tree. That is, if in the tree of recursive calls from $\text{YYI-MAXIMAL-MATCHING}(e_0, \pi)$ some edge e appears multiple times, it is counted only once in the size of the exploration tree. This memorization can lead to a great saving in the query complexity. For instance, suppose that e is the parent of f and g , which are therefore siblings. Let their ranks be $r(e) = \lambda$, $r(f) = \mu$ and $r(g) = \nu$ with $\lambda > \mu > \nu$. If the algorithm queries e , it first explores the subtree H_g . Then, if $\text{YYI-MAXIMAL-MATCHING}(g, \pi)$ returns FALSE, the algorithm proceeds to querying f , only to immediately return to g and explore the same subtree of H_g , as $g \in \delta(f)$. Since the output of $\text{YYI-MAXIMAL-MATCHING}(g, \pi)$ is memorized, the algorithm does not have to explore H_g again.

Thanks to memorization, in [Line 2](#) we can ignore the parent of e , as well as all its siblings. We can ignore the parent p , since e must have been recursively queried from p , therefore $r(p) > r(e)$. As for any sibling f of e , either $r(f) > r(e)$, in which case f can be safely ignored, or $r(f) < r(e)$, in which case f must have already been queried from p and can be ignored due to memorization. This alteration to [Algorithm 10](#) can only reduce the size of the exploration tree $T_{e_0}(\lambda)$. Since in this section we are concerned with a lower-bound on the size of a specific exploration tree, we will analyze this altered version of [Algorithm 10](#).

Lemma 14. Let e_0 be the root edge of the graph H^d , as defined in [Definition 1](#). Then

$$p_{e_0}(\lambda) = x(\lambda)^{\frac{d}{1-d}}$$

$$t_{e_0}(\lambda) = x(\lambda)^{\frac{d}{d-1}},$$

where $x(\lambda) = 1 + (d - 1)\lambda$.

Proof. Let $p(\lambda) = p_{e_0}(\lambda)$ and $t(\lambda) = t_{e_0}(\lambda)$. For ease of notation we denote $\text{YYI-MAXIMAL-MATCHING}(e, \pi)$ by $\text{MM}(e, \pi)$.

A closed form expression for $p(\lambda)$. We first derive a recursive formula for $p(\lambda)$:

$$\begin{aligned} p(\lambda) &= \mathbb{P}[\text{MM}(e_0, \pi) \text{ returns TRUE} | r(e_0) = \lambda] \\ &= \mathbb{P}[\forall e \in \delta(e_0) : r(e) > \lambda \text{ or } \text{MM}(e, \pi) \text{ returns FALSE on } H_e] \\ &= \prod_{e \in \delta(e_0)} \left(1 - \int_0^\lambda \mathbb{P}[\text{MM}(e, \pi) \text{ returns TRUE on } H_e | r(e) = \mu] d\mu \right) \\ &= \left(1 - \int_0^\lambda p(\mu) d\mu \right)^d, \end{aligned} \tag{27}$$

since H_e is isomorphic to H^d (as per [Definitions 1 and 2](#)).

We can now solve this recursion and get a closed form formula for $p(\lambda)$. By raising both sides of [Eq. \(27\)](#) to power $1/d$, we derive that $p^{1/d}(\lambda) = 1 - \int_0^\lambda p(\mu) d\mu$. Differentiating both sides of this equation, we get $\frac{1}{d} \cdot p^{(1/d)-1}(\lambda) \cdot p'(\lambda) = p(\lambda)$, which implies that $p^{(1/d)-2}(\lambda) \cdot p'(\lambda) = d$ and hence

$$p(\lambda) = (C + (d-1)\lambda)^{\frac{d}{1-d}} = x^{\frac{d}{1-d}}(\lambda),$$

as claimed, where $C = 1$ due to the initial condition of $p(0) = 1$.

A closed form expression for $t(\lambda)$. We now derive a recursive formula for $t(\lambda)$. Let T_e be a random variable denoting the size of the exploration tree when running [Algorithm 10](#) from e in H_e . Note that T_e is distributed identically for all e , and $\mathbb{E}(T_e | r(e) = \lambda) = t(\lambda)$, since H_e is always isomorphic to H^d . Let $T = T_{e_0}$. Furthermore, let I_e denote the indicator variable of e being explored through the recursive calls, when the algorithm is originally initiated from e_0 . Then T satisfies

$$T = 1 + \sum_{e \in \delta(e_0)} I_e \cdot T_e,$$

and hence,

$$t(\lambda) = \mathbb{E}[T | r(e_0) = \lambda] = 1 + \sum_{e \in \delta(e_0)} \mathbb{E}[I_e \cdot T_e | r(e_0) = \lambda]$$

The expression $\mathbb{E}[I_e \cdot T_e | r(e_0) = \lambda]$ can be nicely taken apart if we condition on the rank of e , as long as it is less than λ . (If it is more than λ , $I_e = 0$.) Indeed, note that I_e depends only on the ranks and outcomes of the siblings of e , as well as the rank of e itself. Meanwhile, T_e depends only on the ranks in H_e . The only intersection between these is the rank of e , meaning that I_e and T_e are independent conditioned on $r(e)$. These observations lead to

$$t(\lambda) = 1 + \sum_{e \in \delta(e_0)} \int_0^\lambda \mathbb{P}[I_e = 1 | r(e_0) = \lambda, r(e) = \mu] \cdot \mathbb{E}[T_e | r(e) = \mu] d\mu.$$

The expected size of the exploration tree from e is simply $t(\mu)$, again since H_e is isomorphic to H^d . Consider the probability that e is explored at all. This happens exactly when for any sibling of e , f , either $r(f) > r(e)$ or $\text{MM}(f, \pi)$ returns FALSE. This condition is very similar to the condition for $\text{MM}(e_0, \pi)$ returning TRUE (recall [Eq. \(27\)](#)). The only difference is that the condition must hold only for $\delta(e_0) \setminus e$ as opposed to $\delta(e_0)$. Hence we have

$$\mathbb{P}[I_0 = 1 | r(e_0) = \lambda, r(e) = \mu] = \left(1 - \int_0^\mu p(\nu) d\nu\right)^{d-1} = p^{\frac{d}{d-1}}(\mu) = x^{-1}(\mu).$$

In particular, the rhs does not depend on λ . Therefore,

$$t(\lambda) = 1 + d \int_0^\lambda x^{-1}(\mu) t(\mu) d\mu.$$

We can now solve this recursion and get a closed form formula for $t(\lambda)$.

$$\begin{aligned} t(\lambda) &= 1 + d \int_0^\lambda x^{-1}(\mu) t(\mu) d\mu \\ t'(\lambda) &= dx^{-1}(\lambda) t(\lambda) \\ \frac{t'(\lambda)}{t(\lambda)} &= \frac{d}{x(\lambda)} \\ \log(t(\lambda)) &= \frac{d}{d-1} \cdot \log(x(\lambda)) + C_1 \\ t(\lambda) &= C_2 \cdot x^{\frac{d}{d-1}}(\lambda) = x^{\frac{d}{d-1}}(\lambda), \end{aligned}$$

as claimed, due to the initial condition of $t(0) = 1$. □

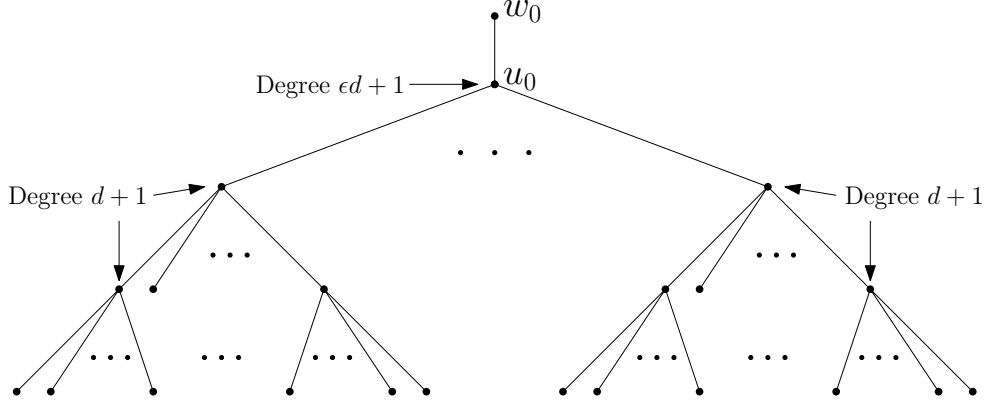


Figure 2: Construction of $H^{d, \epsilon}$.

Corollary 3. Let e_0 be the root edge of the graph H^d , as defined in [Definition 1](#). Then,

$$\mathbb{E}[T_{e_0}] \leq \mathbb{E}_\lambda[t_{e_0}(\lambda)] \leq t_{e_0}(1) = x^{\frac{d}{d-1}}(1) = d^{\frac{d}{d-1}} \leq 2d,$$

for $d \geq 5$.

We next construct an infinite graph with an edge whose expected exploration tree size has nearly quadratic dependence on d .

Definition 3 (Graph $H^{d, \epsilon}$, see [Fig. 2](#) for illustration). Fix some small positive number ϵ . Take ϵd disjoint copies of H^d , call them $H^{(1)}, H^{(2)}, \dots, H^{(\epsilon d)}$. Let the root edge of $H^{(i)}$ be $e^{(i)} = (u^{(i)}, v^{(i)})$. We merge $u^{(1)}, u^{(2)}, \dots, u^{(\epsilon d)}$ into a supernode u_0 , and add a new node w_0 along with an edge $e_0 = (w_0, u_0)$. This creates the infinite graph $H^{d, \epsilon}$; we call the edge e_0 the root edge.

We will now show that querying e_0 in $H^{d, \epsilon}$ with [Algorithm 10](#) produces an exploration tree of nearly quadratic size in expectation.

Lemma 15. For every $\epsilon \in (0, 1)$, every integer $d \geq 5$, in the graph $H^{d, \epsilon}$ (see [Definition 3](#)), where e_0 is the root edge, it holds

$$t_{e_0}(\lambda) \geq \epsilon \cdot x^{2-\epsilon}(\lambda)/2.$$

Proof. Recall the definitions of I_e and T_e from the proof of [Lemma 14](#): Let $I_{e^{(i)}}$ be the indicator variable of $e^{(i)}$ being explored when [Algorithm 10](#) is called from e_0 ; let $T_{e^{(i)}}$ be the size of the exploration tree from $e^{(i)}$ in $H^{(i)}$. For simplicity let $T_i = T_{e^{(i)}}$ and $I_i = I_{e^{(i)}}$. We can derive a formula for $t_{e_0}(\lambda)$ similarly to [Lemma 14](#):

$$t_{e_0}(\lambda) = 1 + \sum_{i=1}^{\epsilon d} \int_0^\lambda \mathbb{P}[I_i | r(e^{(i)}) = \mu] \cdot \mathbb{E}[T_i | r(e^{(i)}) = \mu] d\mu.$$

$\mathbb{E}[T_i | r(e^{(i)}) = \mu]$ is simply $t(\mu) = x^{\frac{d}{d-1}}(\mu)$ by [Lemma 14](#), since the subtree of $e^{(i)}$ in $H^{d, \epsilon}$ is isomorphic to H^d . Similarly to [Lemma 14](#), the probability that $e^{(i)}$ is explored is the probability that for any sibling $e^{(j)}$ of $e^{(i)}$ we have that either $r(e^{(j)}) > r(e^{(i)})$ or $\text{MM}(e^{(j)}, \pi)$ returns FALSE. $e^{(i)}$ has $\epsilon d - 1$ neighbors, each of whose subtree is isomorphic to H^d , hence we have

$$\begin{aligned} \mathbb{P}[I_i | r(e^{(i)}) = \mu] &= \prod_{i=1}^{\epsilon d} \left(1 - \int_0^\mu p_{e^{(i)}}(\nu) d\nu\right) \\ &= \left(1 - \int_0^\mu p(\nu) d\nu\right)^{\epsilon d - 1} \\ &= p^{\frac{\epsilon d - 1}{d-1}}(\mu) \\ &= x^{\frac{\epsilon d - 1}{1-d}}(\mu). \end{aligned}$$

Therefore,

$$\begin{aligned}
t_{e_0}(\lambda) &= 1 + \epsilon d \int_0^\lambda x^{\frac{\epsilon d - 1}{1 - d}} \cdot x^{\frac{d}{d-1}}(\mu) d\mu \\
&\geq 1 + \epsilon d \int_0^\lambda x^{1-\epsilon}(\mu) d\mu \\
&= 1 + \frac{\epsilon d}{(2 - \epsilon)(d - 1)} \cdot (x^{2-\epsilon}(\lambda) - 1) \\
&\geq \epsilon \cdot x^{2-\epsilon}(\lambda)/2,
\end{aligned}$$

as claimed. \square

Corollary 4. *For every $\epsilon \in (0, 1)$, every integer $d \geq 5$, for the root e_0 of the graph $H^{d,\epsilon}$ (see [Definition 3](#)) we have:*

$$\mathbb{E}[T_{e_0}] = \mathbb{E}_\lambda[t_{e_0}(\lambda)] \geq \frac{1}{2} \cdot \epsilon \cdot t_{e_0}(1/2) \geq \epsilon \cdot \frac{1}{4} \cdot \left(\frac{d}{2}\right)^{2-\epsilon}.$$

Therefore this is a construction in which an edge has expected exploration tree size which is nearly quadratic in d . However, the graph $H^{d,\epsilon}$ is infinite, so we proceed to finding a finite graph that has an edge whose exploration tree is also near quadratic. To obtain such a finite graph, we perform the following natural modification of $H^{d,\epsilon}$: we cut off the $H^{d,\epsilon}$ graphs at some depth ℓ , that is we discard all edges e such that $\ell(e) > \ell$, along with vertices that become isolated as a result. (Recall that $\ell(e)$ is the level of the edge e .) We will prove that the exploration tree usually does not explore edges beyond a depth of $O(\log d)$, and so intuitively we should be able to cut the graph at that depth. We make this precise in the next section.

7.2 Depth and Variance Analysis for Infinite Graphs

We will now study the depth of the exploration tree, that is the highest level of any edge in it. Let the depth of the exploration tree from e_0 be D .

Lemma 16. *For every $\ell \geq 1$, if D is the depth of the exploration tree in the graph H^d (see [Definition 1](#)), one has*

$$\mathbb{P}[D \geq \ell] \leq 2^{1-\ell} d^2.$$

Proof. Consider the weight of the exploration tree, defined as follows: For each edge e in the exploration tree we count it with weight $2^{\ell(e)}$. Let the expected weighted size of an exploration tree from e_0 , conditioned on $e_0 = \lambda$ be $T_2(\lambda)$. We can derive a recursive formula for $t_2(\lambda)$ with the same technique as was used to derive a recursive formula for $t(\lambda)$ in [Lemma 14](#), the only difference is the additional factor of 2 on the right hand side (we do not repeat the almost identical proof here). We get

$$t_2(\lambda) = 1 + d \int_0^\mu x^{-1}(\mu)(2t_2(\mu))d\mu.$$

We can then solve this recursion in a similar manner to the one in [Lemma 14](#) (again, the only difference is the extra factor of 2 in the exponent):

$$\begin{aligned}
t_2(\lambda) &= 1 + 2d \int_0^\lambda x^{-1}(\mu)t_2(\mu)d\mu \\
t'_2(\lambda) &= 2dx^{-1}(\lambda)t_2(\lambda) \\
\frac{t'_2(\lambda)}{t_2(\lambda)} &= \frac{2d}{x(\lambda)} \\
\log(t_2(\lambda)) &= \frac{2d}{d-1} \cdot \log(x(\lambda)) + C_1 \\
t_2(\lambda) &= C_2 \cdot x^{\frac{2d}{d-1}}(\lambda) = x^{\frac{2d}{d-1}}(\lambda),
\end{aligned}$$

due to the initial condition of $t_2(0) = 1$. Specifically $t_2(\lambda) < t_2(1) = x^{\frac{2d}{d-1}} \leq 2d^2$ for $d \geq 5$.

We can now complete the proof of the lemma. Note that if the depth D of the exploration tree in H^d satisfies $D \geq \ell$ then the tree contains at least an edge of level ℓ , the weight of the tree must be at least 2^ℓ . Therefore:

$$\begin{aligned} 2d^2 &\geq t_2(1) \geq \mathbb{E}[2^D] \geq 2^\ell \cdot \mathbb{P}[D \geq \ell] \\ \mathbb{P}[D \geq \ell] &\leq 2^{1-\ell} d^2, \end{aligned}$$

as claimed. □

Corollary 5. *In the graph $H^{d,\epsilon}$ (see [Definition 3](#)),*

$$\mathbb{P}[D \geq \ell + 1] \leq 2^{1-\ell} \epsilon d^3.$$

Proof. Indeed, in order for T_{e_0} in $H^{d,\epsilon}$ to have depth $\ell + 1$, $T_{e(i)}$ in $H^{(i)}$ must have depth at least ℓ for at least one of the i 's. We know from [Lemma 16](#) that the probability of this is at most $2^{1-\ell} d^2$ as $H^{(i)}$ is isomorphic to H^d . By simple union bound over all values of i we get that $\mathbb{P}[D \geq \ell + 1] \leq 2^{1-\ell} \epsilon d^3$. □

We have shown in [Corollary 5](#) that the depth of the exploration tree from the root of H^d or $H^{d,\epsilon}$ doesn't exceed $O(\log d)$ with high probability. It would be intuitive to truncate these graphs at depth $\Theta(\log d)$ to get a finite example for a graph with an edge e_0 from which exploration takes quadratic time. However, [Corollary 5](#) does not by itself rule out the possibility that T_{e_0} concentrated extremely badly around its expectation, i.e. the exploration tree is extremely large with very small probability, and most of the work is done beyond the $O(\log d)$ first levels of the tree. We rule this out by exhibiting a $d^{O(1)}$ upper bound on the second moment $\mathbb{E}[T_{e_0}^2]$ in H^d and $H^{d,\epsilon}$. Afterwards, we show that combining these second moment bounds with [Corollary 5](#) and [Lemma 15](#) shows that truncating $H^{d,\epsilon}$ indeed yields a hard instance. Our variance bound is given by:

Lemma 17. *In the graph H^d (see [Definition 1](#)) for the root edge e_0 one has $\mathbb{E}[T_{e_0}^2] \leq 10d^5$.*

Corollary 6. *In the graph $H^{d,\epsilon}$ (see [Definition 3](#)) for the root edge e_0 one has $\mathbb{E}[T_{e_0}^2] \leq 11\epsilon d^6$.*

The proofs follow along the lines of previous analysis, but are more technical and hence both are deferred to [Appendix C](#).

7.3 The Lower Bound Instance (Truncated $H^{d,\epsilon}$)

We are now ready to truncate our graph $H^{d,\epsilon}$:

Definition 4 (Truncated graph $H_\ell^{d,\epsilon}$). *Define $H_\ell^{d,\epsilon}$ as the graph $H^{d,\epsilon}$ reduced to only edges of level at most ℓ .*

Note that $H_\ell^{d,\epsilon}$ is a finite graph, specifically with approximately $n = d^\ell$ vertices. We now show that for some $\ell = O(\log d)$ the size of the exploration tree from the root edge e_0 in the graph $H_\ell^{d,\epsilon}$ is essentially the same as in the graph $H^{d,\epsilon}$ (see [Corollary 4](#) below). This yields our final lower bound instance.

Lemma 18. *For every integer $d \geq 5$, every $\epsilon \in [1/d, 1/2]$ and $\ell \geq 7 \log_2(3d) + 1$, in graph $H_\ell^{d,\epsilon}$ one has $\mathbb{E}[T_\ell] \geq \frac{1}{8} \cdot \epsilon \cdot \left(\frac{d}{2}\right)^{2-\epsilon}$, where T_ℓ is the size of the exploration tree started at the root edge e_0 in $H_\ell^{d,\epsilon}$.*

Proof. Consider a graph $H^{d,\epsilon}$ and its truncated version $H_\ell^{d,\epsilon}$ for $\ell \geq 7 \log_2(3d)$. Let $T = T_{e_0}$ in $H^{d,\epsilon}$ and $T_\ell = T_{e_0}$ in $H_\ell^{d,\epsilon}$. We consider the ranks of edges in $H^{d,\epsilon}$ to be identical to the ranks of the corresponding edges in $H_\ell^{d,\epsilon}$ (this is in a way a coupling of the ranks of the two graphs). Consider the events \mathcal{S} and \mathcal{D} referring to a shallow or a deep exploration tree respectively. Specifically, \mathcal{S} refers to the event that the exploration tree in $H^{d,\epsilon}$ does not exceed a depth of ℓ ; \mathcal{D} is the complement of \mathcal{S} . Note that given \mathcal{S} , $T = T_\ell$ thanks to the coupling of the ranks.

We know from [Corollary 4](#) that

$$\epsilon \cdot \frac{1}{4} \cdot \left(\frac{d}{2}\right)^{2-\epsilon} \leq \mathbb{E}[T] = \mathbb{E}[T \cdot \mathbb{1}(\mathcal{S})] + \mathbb{E}[T \cdot \mathbb{1}(\mathcal{D})] = \mathbb{E}[T_\ell \cdot \mathbb{1}(\mathcal{S})] + \mathbb{E}[T \cdot \mathbb{1}(\mathcal{D})].$$

Thus, in order to prove that $\mathbb{E}[T_\ell \cdot \mathbb{1}(\mathcal{S})] \geq \frac{1}{8} \cdot \epsilon \cdot \left(\frac{d}{2}\right)^{2-\epsilon}$ it suffices to show that

$$\mathbb{E}[T \cdot \mathbb{1}(\mathcal{D})] \leq \frac{1}{8} \cdot \epsilon \cdot \left(\frac{d}{2}\right)^{2-\epsilon}. \quad (28)$$

Let $p = \mathbb{P}(\mathcal{D})$; by [Corollary 5](#) and our choice of $\ell \geq 7 \log_2(3d)$ we know that this is at most $2\epsilon \cdot 3^{-7} \cdot d^{-4} \leq (11 \cdot 64)^{-1} \cdot d^{-4}$, for $\epsilon \leq 1$. Let us upper bound $\mathbb{E}[T \cdot \mathbb{1}(\mathcal{D})] = p \cdot \mathbb{E}[T|\mathcal{D}]$: We know from [Corollary 6](#) that

$$11\epsilon d^6 \geq \mathbb{E}[T^2] \geq \mathbb{E}[\mathbb{1}(\mathcal{D}) \cdot T^2] = p \cdot \mathbb{E}[T^2|\mathcal{D}] \geq p \cdot \mathbb{E}[T|\mathcal{D}]^2 = \frac{(p \cdot \mathbb{E}[T|\mathcal{D}])^2}{p},$$

and thus, rearranging, we get

$$p \cdot \mathbb{E}[T|\mathcal{D}] \leq \sqrt{11\epsilon d^6 p} \leq \frac{1}{8} \cdot \epsilon \cdot \left(\frac{d}{2}\right)^{2-\epsilon},$$

since $d \geq 5$ and $p \leq (11 \cdot 64)^{-1} \cdot d^{-4}$ by assumption of the lemma and setting of parameters. \square

We can now prove the main theorem of this section.

Proof of Theorem 3: Indeed, let $G = H_\ell^{d,\epsilon}$ with e being the root edge and $\ell = \Theta(\log_d n)$. Then the theorem holds by [Lemma 18](#) as long as $d \leq \exp(b\sqrt{\log n})$ for a sufficiently small absolute constant b . \square

8 Lower-bound on the Number of Sampled Edges

8.1 Overview

In this section we prove that our algorithm is nearly optimal with respect to sample complexity, even disregarding the constraint on space. That is, we show that it is impossible to obtain a constant-factor approximation of the maximum matching size with polynomially fewer than n^2 samples.

Theorem 8. *There exists a graph G consisting of $\Theta(n^2)$ edges such that no algorithm can compute a constant-factor approximation of $\text{MM}(G)$ with probability more than $6/10$ while using iid edge stream of length $n^{2-\epsilon}$. More generally, for every constant C , every m between $n^{1+o(1)}$ and $\Omega(n^2)$ it is information theoretically impossible to compute a C -approximation to maximum matching size in a graph with high constant probability using fewer than $m^{1-o(1)}$ iid samples from the edge set of G , even if the algorithm is not space bounded.*

[Theorem 8](#) follows directly from the following result.

Theorem 9. *For any $\epsilon > 0$, any $C > 0$, any m between $n^{1+o(1)}$ and $\Omega(n^2)$, and large enough n there exists a pair of distributions of graphs on n vertices, \mathcal{D}^{YES} and \mathcal{D}^{NO} , such that the sizes of the maximum matchings of all graphs in \mathcal{D}^{NO} are M and the sizes of the maximum matchings of all graphs in \mathcal{D}^{YES} are at least CM . However, the total variation distance between an iid edge stream of length $m^{1-\epsilon}$ of a random graph in \mathcal{D}^{YES} and one in \mathcal{D}^{NO} is at most $1/10$.*

Overview of the approach. Our lower bound is based on a construction of two graphs G and H on n vertices such that for a parameter k **(a)** matching size in G is smaller than matching size in H by a factor of $n^{\Omega(1)/k}$ but **(b)** there exists a bijection from vertices of G to vertices of H that preserves k -depth neighborhoods up to isomorphism. To the best of our knowledge, this construction is novel. Related constructions have been shown in the literature (e.g., cluster trees of [\[KMW16\]](#)), but these constructions would not suffice for our lower bound, since they do not provide a property as strong as **(b)** above. For example, the construction of [\[KMW16\]](#) only produces one graph G with a large matching together with two subsets of vertices S, S' of G whose neighborhoods are isomorphic. This suffices for proving strong lower bounds on finding near-optimal matchings in a distributed setting [\[KMW16\]](#), but not for our purpose. Indeed, it is crucial for us to have a gap (i.e., two graphs G and H) and have the strong indistinguishability property provided by **(b)**.

Our construction proceeds in two steps. We first construct two graphs G' and H' that have identical k -level degrees (see [Section 8.3](#)). This produces two graphs G' and H' that are indistinguishable based

on k -level degrees (but whose neighborhoods are not isomorphic due to cycles) but whose matching size differs by an $n^{\Omega(1/k)}$ factor. These graphs have $n^{2-O(1/k)}$ edges and provide nearly tight instances for peeling algorithms that we hope may be useful in other contexts. We note that a similar step is used in the construction of cluster trees of [KMW16], but, as mentioned above, these graphs provide neither the indistinguishability property for all vertices nor a gap in matching size. Furthermore, the number of edges in the corresponding instances of [KMW16] is $\tilde{O}(n^{3/2})$, i.e., the graphs do not get denser with large k , whereas our construction appears to have the optimal behaviour. The second step of our construction is a lifting map (see [Theorem 11](#)) that relies on high girth Cayley graphs and allows us to convert graphs with identical k -level vertex degrees to graphs with isomorphic depth- k neighborhoods without changing matching size by much. The details are provided in [Section 8.4](#).

Finally, the proof of the sampling lower bound proceeds as follows. To rule out factor C approximation in $m^{1-o(1)}$ space, take a pair of constant (rather, $m^{o(1)}$) size graphs G and H such that **(a)** matching size in G is smaller than matching size in H by a factor of C and **(b)** for some large k one has that k -depth neighborhoods in G are isomorphic to k -depth neighborhoods in H . Then the actual hard input distribution consists of a large number of disjoint copies of G in the **NO** case and a large number of copies of H in the **YES** case, possibly with a small disjoint clique added in both cases to increase the number of edges appropriately. Since the vertices are assigned uniformly random labels in both cases, the only way to distinguish between the **YES** and the **NO** case is to ensure that at least k edge-samples land in one of the small copies of H or G . Since k is small, the result follows.

We now give the details. Formally, our main tool will be a pair of constant sized graphs that are indistinguishable if only some given constant number of edges are sampled from either. This is guaranteed by the following theorem, proved in [Section 8.5](#).

Theorem 10. *For every $\lambda > 1$ and every k , there exist graphs G and H such that $MM(G) \geq \lambda \cdot MM(H)$, but for every graph K with at most k edges, the number of subgraphs of G and H isomorphic to K are equal.*

Defining distributions \mathcal{D}^{YES} and \mathcal{D}^{NO} . All the graphs from our distributions will have the same vertex set $V \stackrel{\text{def}}{=} [n]$. Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be the two graphs provided by [Theorem 10](#) invoked with parameters $\lambda = 2C$ and $k = 2/\epsilon$. Let $q \stackrel{\text{def}}{=} \max(|V_G|, |V_H|)$ (our construction in fact guarantees that $|V_G| = |V_H|$). Let $s \stackrel{\text{def}}{=} MM(H)$, and hence $MM(G) \geq \lambda \cdot s = 2C \cdot s$.

Partition V into the following:

1. $r = \frac{n}{2q}$ sets of size q , denoted by V_1, \dots, V_r ;
2. a set V_K consisting of w vertices, for $w \in [0, ns/q]$; and
3. set I containing the remaining vertices.

The sets V_1, \dots, V_r will serve as the vertex sets of copies of G or H , where V_i equals V_G or equals V_H depending on whether we are constructing \mathcal{D}^{YES} or \mathcal{D}^{NO} . The set V_K will be a clique, while I will be a set of isolated vertices in the construction. The distributions \mathcal{D}^{YES} and \mathcal{D}^{NO} are now defined as follows:

\mathcal{D}^{YES} : Take r independently uniformly random permutations π_1, \dots, π_r on V_1, \dots, V_r respectively, and construct a copy G_i of G embedded into V_i via π_i . Then construct a clique K_w on V_K .

\mathcal{D}^{NO} : Take r independently uniformly random permutations π_1, \dots, π_r on V_1, \dots, V_r respectively and construct a copy H_i of H embedded into V_i via π_i . Then construct a clique K_w on V_K .

We will refer to copies of G and H in the two distributions above as *gadgets*. We now give an outline of the proof of [Theorem 9](#) assuming [Theorem 10](#). The full proof follows the same steps, but is more involved, and is deferred to [Appendix D](#).

Proof outline (of [Theorem 9](#)). Naturally, our distribution-pair will be \mathcal{D}^{YES} and \mathcal{D}^{NO} as defined above. Note that the maximum matching size of any element of the support of \mathcal{D}^{YES} is at least $r \cdot 2Cs$ as it contains r copies of G . On the other hand, any element of the support of \mathcal{D}^{NO} contains r copies of H as well as a clique of size $w \leq ns/q$ which means its maximum matching size is at most $r \cdot s + ns/2q \leq 2r \cdot s$. Hence, the sizes of maximum matchings in \mathcal{D}^{YES} and maximum matchings in \mathcal{D}^{NO} differ by at least factor C , as desired.

Note that the number of edges in the construction is at least $\binom{w}{2}$ and at most $\binom{w}{2} + r \cdot \binom{q}{2} \leq w^2 + qn$. Thus the number of edges can be set to be (within a constant factor of) anything from $n^{1+o(1)}$ to $\Omega(n^2)$.

Next, we compute the total variation distance between iid edge streams of length $m^{1-\epsilon}$ of a graph sampled from \mathcal{D}^{YES} and \mathcal{D}^{NO} respectively. Denote these random iid edge streams by C_1 and C_2 , respectively. Consider the following event, that we call *bad*,

$$\mathcal{E} \stackrel{\text{def}}{=} \{\exists i \in [r] : \text{edges between vertices of } V_i \text{ appear more than } k \text{ times in the stream}\}.$$

We show below that distributions of C_1 conditioned on $\bar{\mathcal{E}}$ is identical to the distribution of C_2 conditioned on $\bar{\mathcal{E}}$. Then the total variation distance is bounded by the probability of \mathcal{E} . We first bound this probability, and then prove the claim above.

Upper bounding the probability of \mathcal{E} . Consider a realization of \mathcal{D}^{YES} or a realization of \mathcal{D}^{NO} . Since we have $m^{1-\epsilon}$ edge samples each chosen uniformly at random from a possible m edges, each specific edge, e , appears exactly $m^{-\epsilon}$ times throughout the stream in expectation. Therefore, by Markov's inequality the probability that e ever appears in the stream is at most $\frac{3m^{-\epsilon}}{\mu^2}$. Also, the event that an edge e_1 appears in the stream and the event that an edge $e_2 \neq e_1$ appears in the stream are negatively associated.

Fix a single gadget of the realized graph; this gadget has at most q^2 edges. Therefore, by union bound and from the negative association outlined above, the probability that at least $k+1$ edges of the gadget will be sampled is at most

$$\binom{q^2}{k+1} \cdot (m^{-\epsilon})^{k+1} \leq \binom{q^2}{k+1} \cdot m^{-2},$$

where we used the assumption that $k = 2/\epsilon$. Thus, again by union bound, the probability that any of the $r = \frac{n}{2q}$ gadgets has at least $k+1$ of its edges occurring in the stream is at most

$$\frac{n}{2q} \cdot \binom{q^2}{k+1} \cdot m^{-2} \leq 1/10,$$

for large enough n and $m > n$. So, $\mathbb{P}[\mathcal{E}] \leq 1/10$ under both distributions.

Analyzing conditional distributions. It remains to show that C_1 and C_2 are identically distributed when conditioned on $\bar{\mathcal{E}}$. We now give an overview of this proof, and defer details to [Appendix D](#). Since the cliques are identical across the two distributions, edges sampled from them are no help in distinguishing between \mathcal{D}^{YES} and \mathcal{D}^{NO} . Consider a single gadget. (As a reminder, a gadget is a copy of G or H .) By conditioning on $\bar{\mathcal{E}}$ we have that only at most k distinct edges of this gadget are observed. Since the gadgets are randomly permuted in both \mathcal{D}^{YES} and \mathcal{D}^{NO} , only the isomorphism-class of the sampled subgraph of the gadget provides any information. However, each isomorphism-class's probability is proportional to the number of times such a subgraph appears in the gadget. This is equal across the YES and NO cases thanks to the guarantee of [Theorem 10](#) on G and H . \square

From here it remains to prove [Theorem 10](#). We organize the rest of this section as follows. In [Section 8.3](#) we define and analyze our main construction, which is a pair of graphs isomorphic with respect to k -level degrees while having greatly different matching numbers. That is to say, we produce two graphs and a bijection between them such that the bijection preserves degree structure up to a depth of k , disregarding cycles. (For the definition of k -level degree see [Section 8.2](#).) This is the main technical result of our lower bound. Then in [Section 8.4](#) we use a graph lifting construction to increase the girths of our graphs, resulting in a pair of graphs that have truly isomorphic k -depth neighborhoods. Finally in [Section 8.5](#) we prove [Theorem 10](#), concluding the lower bound.

8.2 Preliminaries

We begin by stating a few definition which will be used in the rest of [Section 8](#). First, we recall a few basic graph theoretical definitions:

Definition 5 (*c*-star). *We call a graph with a single degree c vertex connected to c degree 1 vertices a *c*-star. We call the degree c vertex the center and degree 1 vertices petals.*

Definition 6 (girth of a graph). *The girth of a graph is the length of its shortest cycle.*

Definition 7 (permutation group of V). Let $G = (V, E)$ be a graph. We call the subgroup of S_V (the permutation group of V) containing all permutations that preserve edges the automorphism group of G . That is, a permutation $\pi \in S_V$ is in the automorphism group if the following holds:

$$\forall v, w \in V : (v, w) \in E \iff (\pi(v), \pi(w)) \in E.$$

We denote it $\text{Aut}(G)$.

We now define two local property of a vertex, i.e., k -hop neighborhood and k -level degree.

Definition 8 (k -hop neighbourhood of a vertex). Let the k -hop neighborhood of a vertex v in graph G be defined as the subgraph induced by vertices of G with distance at most k from v .

Definition 9 (k -level degree of a vertex). Let the k -level degree of a vertex v in a graph G denoted by $d_k^G(v)$ be a multiset defined recursively as follows:

- $d_1^G(v) \stackrel{\text{def}}{=} d(v)$, the degree of v in G .
- For $k > 1$, $d_k^G(v) \stackrel{\text{def}}{=} \{d_{k-1}^G(w) | w \in N(v)\}$, where this is a multiset.

For ease of presentation, in the future we will use the following less intuitive but more explicit formulation:

$$d_k^G(v) = \biguplus_{w \in N(v)} \{d_{k-1}^G(w)\},$$

where \biguplus denotes multiset union. Moreover, if G is clear from context, we will omit the superscript and write $d_k(v)$ instead of $d_k^G(v)$.

Note that the above two definitions are similar but distinct, with the k -hop neighborhood containing the more information of the two. Imagine a vertex v of a C_3 cycle and a vertex w of a C_4 cycle. Their arbitrarily high level degrees are identical, but their 2-hop neighborhoods contain their respective graphs fully, and so they are clearly different.

Observation 3. The following conditions are sufficient for the k -hop neighborhoods of vertices v and w (from graphs G and H respectively) to be isomorphic:

1. $d_k^G(v) = d_k^H(w)$
2. G and H both have girth at least $2k + 2$, that is neither graph contains a cycle shorter than $2k + 2$.

This observation will be crucial to our construction as our main goal will be to construct two graphs G and H such that a bijection between their vertex-sets preserves k -depth neighborhoods. We achieve this by first guaranteeing condition 1. above, then improving our construction to guarantee condition 2. as well.

8.3 Constructing Graphs with Identical k -Level Degrees

As stated before, our first goal is to design a pair of graphs with greatly differing maximum matching sizes such that, nonetheless, there exists a bijection between them preserving k -level degrees for some large constant k . We will later improve this construction so that the bijection also preserves k -hop neighborhoods.

Definition 10 (Degree padding \bar{G} of a graph G and special vertices). For every graph $G = (V, E)$, let d_l and d_h be minimum and maximum vertex degrees in G respectively. Define the degree padding $\bar{G} = (\bar{V}, \bar{E})$ of G as the graph obtained from G by adding a bipartite clique between vertices of degree d_l and a new set of $d_h - d_l$ vertices. More formally, let S be a set of $d_h - d_l$ nodes disjoint from V , let $\bar{V} := V \cup S$, and $\bar{E} := E \cup (S \times \{v \in V | d(v) = d_l\})$ (see [Fig. 4](#)). We refer to the set S in \bar{G} as the set of special vertices.

Definition 11 (Recursive construction of k -depth similar graphs $G^{(k)}$ and $H^{(k)}$). For every integer $k \geq 1$ and integer $c \geq 1$, define graphs $G^{(k)}$ and $H^{(k)}$ recursively as follows. For $k = 1$, let $G^{(1)}$ be a graph that is the disjoint union of a $c + 1$ -clique and $(c + 1)c/2$ isolated edges. Let $H^{(1)}$ be $c + 1$ disjoint copies of c -stars (see [Fig. 3](#)). For $k > 1$, let $\bar{G}^{(k-1)}$ denote a degree padding of $G^{(k-1)}$ as per [Definition 10](#), and let $G^{(k)} = \bar{G}_1^{(k-1)} \cup \dots \cup \bar{G}_c^{(k-1)}$ denote the disjoint union of c copies of $\bar{G}^{(k-1)}$. Similarly let $\bar{H}^{(k)}$ denote the degree padding of $H^{(k-1)}$, and let $H^{(k)} = \bar{H}_1^{(k-1)} \cup \dots \cup \bar{H}_c^{(k-1)}$ denote the disjoint union of c copies of $\bar{H}^{(k-1)}$.

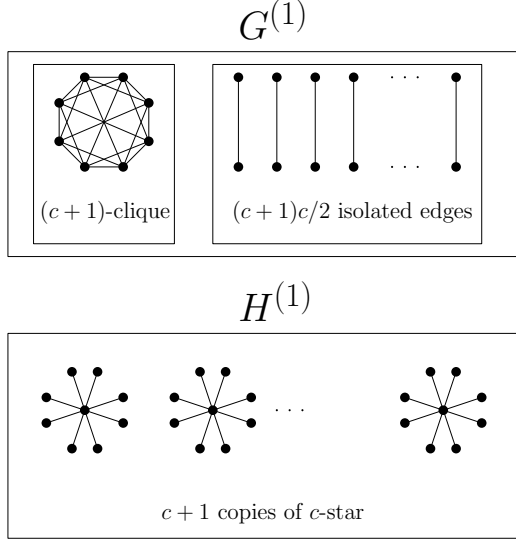


Figure 3: Construction of $G^{(1)}$ and $H^{(1)}$.

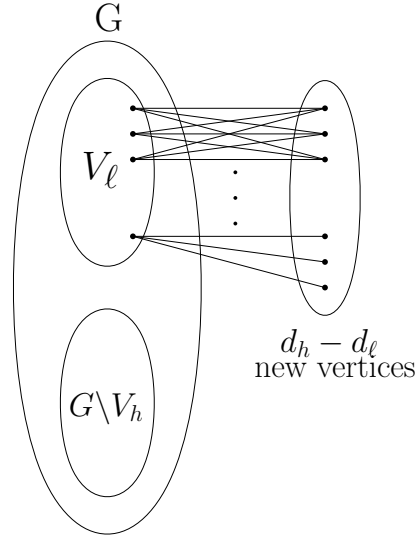


Figure 4: Padding of a graph G .

We first prove some simple structural claims about $G^{(j)}$ and $H^{(j)}$.

Lemma 19. *For graphs $G^{(j)}$ and $H^{(j)}$ as defined in Definition 11 there exist numbers $N_h^{(j)}$, $N_l^{(j)}$ and $d_h^{(j)} > d_l^{(j)}$ such that both $G^{(j)}$ and $H^{(j)}$ contain exactly $N_h^{(j)}$ vertices of degree $d_h^{(j)}$, $N_l^{(j)}$ vertices of degree $d_l^{(j)}$ and no other vertices. Let $V^{(j)}$ and $W^{(j)}$ denote the vertex-sets of $G^{(j)}$ and $H^{(j)}$, respectively. Furthermore, let $V_h^{(j)} \subseteq V^{(j)}$ and $W_h^{(j)} \subseteq W^{(j)}$ be the vertices of degree $d_h^{(j)}$; let $V_l^{(j)} \subseteq V^{(j)}$ and $W_l^{(j)} \subseteq W^{(j)}$ be vertices of degree $d_l^{(j)}$.*

Proof. We prove the lemma by induction on j . Clearly, for $j = 1$, the claim of the lemma is satisfied with $N_h^{(1)} = c + 1$, $N_l^{(1)} = (c + 1)c$, $d_h^{(1)} = c$ and $d_l^{(1)} = 1$.

Inductive step: $j - 1 \rightarrow j$. Consider $\bar{G}^{(j-1)}$ and $\bar{H}^{(j-1)}$. It is clear that the vertices of both of these graphs fall into two categories: the old vertices of $G^{(j-1)}$ and $H^{(j-1)}$ respectively as well as the newly introduced special vertices. The old vertices used to be of degree either $d_h^{(j-1)}$ or $d_l^{(j-1)}$ according to the inductive hypothesis. After the padding, the degree of those vertices is uniform and equal to $d_h^{(j-1)}$. Since the special vertices are connected to all the old vertices that used to have low degree in $G^{(j-1)}$ and $H^{(j-1)}$ respectively, the degrees of each special vertex is $N_l^{(j-1)}$. By definition, the number of special vertices in each of $\bar{G}^{(j-1)}$ and $\bar{H}^{(j-1)}$ is $d_h^{(j-1)} - d_l^{(j-1)}$.

All that remains to be proven is that the special vertices have strictly higher degree than the old vertices in, say $\bar{G}^{(j-1)}$, that is $N_l^{(j-1)} > d_h^{j-1}$. For $j = 2$ one can simply verify that this is true from the base construction of Definition 11. For $j > 2$ consider the structure of $G^{(j-1)}$: $G^{(j-1)} = \bar{G}_1^{(j-2)} \cup \dots \cup \bar{G}_c^{(j-2)}$ where $\bar{G}_i^{(j-2)}$ are disjoint copies of the degree padded version of $G^{(j-2)}$. Hence, any high degree vertex of $G^{(j-1)}$ corresponds to a special vertex used in the padding of $G^{(j-2)}$ and so, no two high degree vertices of $G^{(j-1)}$ are connected to each other. So a high degree vertex of $G^{(j-1)}$ is only connected to its low degree vertices, and not even all of them, since $G^{(j-1)}$ falls into c disjoint copies. Therefore its degree, $d_h^{(j-1)}$ must be smaller than the number of low degree vertices in the same graph, $N_l^{(j-1)}$.

In conclusion, we have proved the equivalent of the lemma's statement for $\bar{G}^{(j-1)}$ and $\bar{H}^{(j-1)}$. It is easy to see that duplicating this c times will not disrupt this. \square

Lemma 20. *For graphs $G^{(j)}$ and $H^{(j)}$ as defined in Definition 11 and $c \geq 2k$, the following inequalities hold for the quantities from Lemma 19:*

- $N_h^{(j)} \leq c^j + 2jc^{j-1}$
- $N_l^{(j)} \leq c^{j+1} + 2jc^j$

- $d_h^{(j)} \leq c^j + 2jc^{j-1}$

Proof. We prove the claims by induction. As mentioned before $N_h^{(1)} = c + 1$, $N_l^{(1)} = (c + 1)c$ and $d_h^{(1)} = c$ which satisfies the inequalities.

Inductive step: $j - 1 \rightarrow j$. Since these quantities can be derived equivalently from either $G^{(j)}$ or $H^{(j)}$ by [Lemma 19](#), we will be looking at $G^{(j)}$ for simplicity. The set of high degree vertices in $G^{(j)}$ are the copies of the special vertices from degree padding $G^{(j-1)}$, which number $d_h^{(j-1)} - d_l^{(j-1)}$. So $N_h^{(j)} = c \left(d_h^{(j-1)} - d_l^{(j-1)} \right) \leq c d_h^{(j-1)} \leq c^j + 2(j-1)c^{j-1} \leq c^j + 2jc^{j-1}$ as claimed.

The set of low degree vertices in $G^{(j)}$ are copies of the old vertices from the unpadding $G^{(j-1)}$, which number $|V^{(j-1)}| = N_h^{(j-1)} + N_l^{(j-1)}$. So $N_l^{(j)} = c \left(N_h^{(j-1)} + N_l^{(j-1)} \right) \leq c^j + 2(j-1)c^{j-1} + c^{j+1} + 2(j-1)c^j \leq c^{j+1} + 2jc^j$ as claimed, since $2(j-1) \leq 2k \leq c$.

The high degree vertices in $G^{(j)}$ are copies of the special vertices added during the degree padding of $G^{(j-1)}$; their degree is $N_l^{(j-1)}$ as prescribed in [Definition 16](#). So $d_h^{(j)} = N_l^{(j-1)} = c^j + 2(j-1)c^{j-1} \leq c^j + 2jc^{j-1}$ as claimed. \square

We will now show that the discrepancy in the matching numbers of $G^{(j)}$ and $H^{(j)}$ persists throughout the recursive construction.

Lemma 21. *For $c \geq 2k$ the graph $G^{(j)}$ as constructed in [Definition 11](#) has maximum matching size at least $(c + 1)c^j/2$ while $H^{(j)}$ from the same construction has maximum matching size at most $2jc^j$.*

Proof. We prove the following claim by induction: $G^{(j)}$ has a feasible matching of size $(c + 1)c^j/2$ while $H^{(j)}$ has a feasible vertex cover of size $2jc^j$. For the case of $j = 1$ this is clear: the set of isolated edges in $G^{(1)}$ constitutes a matching and the centers of the stars in $H^{(1)}$ constitute a vertex cover.

Inductive step: $j - 1 \rightarrow j$. Degree padding does not affect the feasibility of a matching. Hence, when constructing $G^{(j)}$ by duplicating $\bar{G}^{(j-1)}$ c times, to construct a matching in $G^{(j)}$ we can simply duplicate the matchings from $\bar{G}^{(j-1)}$ as well. This results in a matching in $G^{(j)}$ that is c times larger than the one in $\bar{G}^{(j-1)}$. Therefore, by the inductive hypothesis $G^{(j)}$ has a sufficiently large feasible matching.

Now we discuss the size of minimum vertex cover in $H^{(j)}$. Upon degree padding $H^{(j-1)}$ we add the special vertices to the vertex cover, increasing its size by $d_h^{(j-1)} - d_l^{(j-1)} \leq c^{j-1} + 2(j-1)c^{j-2}$. When duplicating $\bar{H}^{(j-1)}$ c times we duplicate the vertex cover as well. This makes the size of our feasible vertex cover of $H^{(j)}$ at most $2(j-1)c^j + c^j + 2(j-1)c^{j-1} \leq 2jc^j$ as claimed, since $2(j-1) \leq 2k \leq c$. \square

Next we define a sequence of bijections between vertex-sets of $G^{(j)}$ and $H^{(j)}$. We begin by giving the following definitions.

Each of the bijections between the vertex-sets of $G^{(j)}$ and $H^{(j)}$ that we define preserves j -level degree.

Lemma 22. *For $c \geq 2k$, let $\tilde{G} \stackrel{\text{def}}{=} G^{(j)}$ and $\tilde{H} \stackrel{\text{def}}{=} H^{(j)}$ be defined as in [Definition 11](#). Then, there exists a bijection $\Phi^{(j)} : V^{(j)} \rightarrow W^{(j)}$ such that for every $v \in W^{(j)}$ it holds $d_j^{\tilde{H}}(v) = d_j^{\tilde{G}}(\Phi^{(j)}(v))$.*

Proof. We prove this lemma by induction on j .

Base case: $j = 1$. Consider the following bijection $\Phi^{(1)}$ that maps the vertices from $G^{(1)}$ to $H^{(1)}$: $\Phi^{(1)}$ maps vertices of degree c (vertices in the clique in $G^{(1)}$) to the centers of stars in $H^{(1)}$, and endpoints of isolated edges in $G^{(1)}$ to petals in $H^{(1)}$. $\Phi^{(1)}$ can be arbitrary as long as it satisfies this constraint (and remains a bijection). Observe that $\Phi^{(1)}$ is a bijection such that the vertices of $G^{(1)}$ are mapped to vertices of the same degree in $H^{(1)}$.

Inductive step: $j - 1 \rightarrow j$. We first define a bijection $\Phi^{(j)}$ and then argue that it maps the vertices of $G^{(j)}$ to the vertices of $H^{(j)}$ with the same j -level degree.

We define $\Phi^{(j)}$ inductively on j as follows. Recall that $G^{(j)}$ is a disjoint union of c copies of the degree padding $\bar{G}^{(j-1)}$ of $G^{(j-1)}$, and $H^{(j)}$ is a disjoint union of c copies of the degree padding $\bar{H}^{(j-1)}$ of $H^{(j-1)}$. To define $\Phi^{(j)}$, we “reuse” $\Phi^{(j-1)}$ (which exists by the inductive hypothesis) and add the mapping for the vertices not included by $\Phi^{(j-1)}$; these vertices are called *special* (see [Definition 16](#)). By

Lemma 19 the number of special vertices in $\bar{G}_i^{(j-1)}$ equals the number of special vertices in $\bar{H}_i^{(j-1)}$, for every $i = 1, \dots, c$. So, we define $\Phi^{(j)}$ to map the special set A_i of vertices in $\bar{G}_i^{(j-1)}$ bijectively (and arbitrarily) to the special set B_i of vertices in $\bar{H}_i^{(j-1)}$.

For the sake of brevity, let $d_\iota(v) \stackrel{\text{def}}{=} d_\iota^{G^{(j-1)}}(v)$ for vertices v of $G^{(j-1)}$ and $d_\iota(v) \stackrel{\text{def}}{=} d_\iota^{H^{(j-1)}}(v)$ for vertices of $H^{(j-1)}$. Similarly, let $\bar{d}_\iota(v) \stackrel{\text{def}}{=} d_\iota^{\bar{G}^{(j-1)}}(v)$ for vertices v of $\bar{G}^{(j-1)}$ and $\bar{d}_\iota(v) \stackrel{\text{def}}{=} d_\iota^{\bar{H}^{(j-1)}}(v)$ for vertices of $\bar{H}^{(j-1)}$. Furthermore, let $N(v)$ and $\bar{N}(v)$ denote the neighborhood of vertex v in $G^{(j-1)} \cup H^{(j-1)}$ and in $\bar{G}^{(j-1)} \cup \bar{H}^{(j-1)}$, respectively.

Since \tilde{G} and \tilde{H} are c disjoint copies of $\bar{G}^{(j-1)}$ and $\bar{H}^{(j-1)}$, respectively, it suffices to show that $\Phi^{(j)}$ maps a vertex of $\bar{G}^{(j-1)}$ to a vertex of $\bar{H}^{(j-1)}$ with the same j -level degree. Recall that $V(\bar{G}^{(j-1)}) = V^{(j-1)} \cup A$, where $V^{(j-1)} = V(G^{(j-1)})$ and A refers to the special vertices added to $G^{(j-1)}$. Similarly, recall that $V(\bar{H}^{(j-1)}) = W^{(j-1)} \cup B$. For the rest of our proof, we show the following claim.

Claim 1. *We have that:*

- (A) *For any $u \in V^{(j-1)}$ and $u' \in W^{(j-1)}$ (not necessarily mapped to each other by the bijection) if $d_{\iota-1}(u) = d_{\iota-1}(u')$, then $\bar{d}_\iota(u) = \bar{d}_\iota(u')$. (For the purposes of this statement, let $d_0^G(v) = \emptyset$ for every v .)*
- (B) *For any $u \in A$ and $u' \in B$ it holds $\bar{d}_\iota(u) = \bar{d}_\iota(u')$.*

Proof. We prove this claim by induction.

Base case: $\iota = 1$.

Proof of (A) By construction of $\bar{G}^{(j-1)}$ and $\bar{H}^{(j-1)}$, all vertices in either $V^{(j-1)}$ or $W^{(j-1)}$ have degree exactly $d_h^{(j-1)}$, so their 1-level degrees are equal.

Proof of (B) By construction of $\bar{G}^{(j-1)}$ and $\bar{H}^{(j-1)}$, all vertices in either A or B have degree exactly $d_l^{(j-1)}$, so their 1-level degrees are equal.

Inductive step: $\iota - 1 \rightarrow \iota$.

Proof of (A) Let u and u' be two vertices satisfying condition (A) for $\iota - 1$, i.e., $d_{\iota-1}(u) = d_{\iota-1}(u')$. Then, by definition

$$\biguplus_{\omega \in N(u)} \{d_{\iota-2}(\omega)\} = \biguplus_{\omega' \in N(u')} \{d_{\iota-2}(\omega')\}.$$

This means that there exists a bijection between the $G_i^{(j-1)}$ -neighborhoods of u and u' , denoted by $\Phi_u^*: N(u) \rightarrow N(u')$ that preserves $\iota - 2$ -level degrees. For any $\omega \in N(u)$ and $\omega' \in N(u')$ such that $\Phi_u^*(\omega) = \omega'$, by (A) of the inductive hypothesis it is also true that $\bar{d}_{\iota-1}(\omega) = \bar{d}_{\iota-1}(\omega')$. Therefore

$$\biguplus_{\omega \in N(u)} \{\bar{d}_{\iota-1}(\omega)\} = \biguplus_{\omega' \in N(u')} \{\bar{d}_{\iota-1}(\omega')\}. \quad (29)$$

To prove the inductive step for (A), we will show that

$$\biguplus_{\omega \in \bar{N}(u)} \{\bar{d}_{\iota-1}(\omega)\} = \biguplus_{\omega' \in \bar{N}(u')} \{\bar{d}_{\iota-1}(\omega')\} \quad (30)$$

as follows. Note that in (30) the neighbors w iterate over \bar{N} , while in (29) they iterate over N . Now we consider two cases.

Case $u \in V_h^{(j-1)}$: It follows that $u' \in W_h^{(j-1)}$. Then we are done, since u is not connected to A and its neighborhoods in $G_1^{(j-1)}$ and $\bar{G}_1^{(j)}$ are identical. Similarly for u' .

Case $u \in V_l^{(j-1)}$: It follows that $u' \in W_l^{(j-1)}$. Then $\bar{N}(u) \setminus N(u) = A$. Similarly $\bar{N}(u') \setminus N(u') = B$. It holds that $|A| = |B|$ and, by (B) of the inductive hypothesis, any vertex of A and any vertex of B have identical $\bar{d}_{\iota-1}$ -degrees. Therefore, extending the multiset-union from $N(u)$ and $N(u')$ to $\bar{N}(u)$ and $\bar{N}(u')$, respectively, preserves the equality of \bar{d}_ι -degrees.

Hence, in both cases it holds that $\bar{d}_\iota(u) = \bar{d}_\iota(u')$, as claimed.

Proof of (B) Consider vertices $u \in A$ and $u' \in B$. In this case $\bar{N}(u) = V_l^{(j-1)}$ and $\bar{N}(u') = W_l^{(j-1)}$, so our goal is to prove that

$$\biguplus_{\omega \in V_l^{(j-1)}} \{\bar{d}_{l-1}(\omega)\} = \biguplus_{\omega' \in W_l^{(j-1)}} \{\bar{d}_{l-1}(\omega')\}$$

or, equivalently, we aim to show that there exists a bijection between $V_l^{(j-1)}$ and $W_l^{(j-1)}$ that preserves \bar{d}_{l-1} -degree.

By the claim of the *outer* inductive hypothesis, i.e., by Lemma 22, there exists a bijection $\Phi^{(j-1)}$ that preserves the d_{j-1} -degree between $V^{(j-1)}$ and $W^{(j-1)}$. Since $\Phi^{(j-1)}|_{V_l^{(j-1)}}$ preserves d_{j-1} -degree it also preserves d_{l-2} -degree and by (A) it also preserves \bar{d}_{l-1} -degree. Thus, (B) holds. \square

To conclude the proof of Lemma 22 consider again a vertex $v \in V^{(j-1)} \cup A$ to which $\Phi^{(j)}$ can be applied. If $v \in V^{(j-1)}$, then the claim holds by Claim 1 (A); if it is in A , then the claim holds by Claim 1 (B). \square

Corollary 7. *For large enough $c \geq 2k$ there exists a bijection $\Phi^{(k)} : V^{(k)} \rightarrow W^{(k)}$ such that for any $v \in V^{(k)}$ $d_k(v) = d_k(\Phi^{(k)}(v))$. However, $MM(G_1)$ and $MM(G_2)$ differ by at least a factor $\frac{c+1}{4k}$.*

Proof. By Lemma 21 and Lemma 22 the graphs $G^{(k)}$ and $H^{(k)}$ constructed in Definition 11 satisfy these requirements when $c \geq 2k$. \square

8.4 Increasing Girth via Graph Lifting

Let us start this section by introducing Cayley graphs, we will later use them in order to increase the girth. Girth refers to the minimum length of a cycle within a graph.

Definition 12. *Let \mathcal{G} be a group and S a generating set of elements. The Cayley graph associated with \mathcal{G} and S is defined as follows: Let the vertex set of the graph be \mathcal{G} . Let any two elements of the vertex set, g_1 and g_2 be connected by an edge if and only if $g_1 \cdot s = g_2$ or $g_1 \cdot s^{-1} = g_2$ for some element $s \in S$. We can think of the edges of a Cayley graph as being directed and labeled by generator elements from S .*

Remark 7. *Consider traversing a (undirected) walk in a Cayley graph. Let the sequence of edge-labels of the walk be s_1, s_2, \dots, s_l . Further, let $\epsilon_1, \epsilon_2, \dots, \epsilon_l$ be ± 1 variables, where ϵ_i corresponds to whether we have crossed the i^{th} edge in the direction associated with right-multiplication by s_i ($\epsilon_i = 1$) or in the direction associated with right-multiplication by s_i^{-1} ($\epsilon_i = -1$). Then traversing the walk corresponds to multiplication by $s_1^{\epsilon_1} \cdot s_2^{\epsilon_2} \cdot \dots \cdot s_l^{\epsilon_l}$, that is the final vertex of the walk corresponds to the starting vertex multiplied by this sequence.*

Definition 13. *Let \mathcal{G} be a group. A sequence of elements $s_1^{\epsilon_1} \cdot s_2^{\epsilon_2} \cdot \dots \cdot s_l^{\epsilon_l}$, from some generator set S , is considered irreducible if no two consecutive elements cancel out. That is*

$$\nexists i : s_i^{\epsilon_i} \cdot s_{i+1}^{\epsilon_{i+1}} = \mathbb{1}.$$

Remark 8. *By the previous remark, circuits of a Cayley graph associated with \mathcal{G} and S correspond to irreducible sequences from S multiplying to $\mathbb{1}$. A circuit is a closed walk with no repeating edges. (Note that the other direction is not true: Not all irreducible sequences from S multiplying to $\mathbb{1}$ correspond to circuits in the Cayley graph, as they could have repeating edges.)*

Taking $(G_1^{(k)}, G_2^{(k)})$ from the previous section we now have a pair of graphs that differ greatly in their maximum matching size but are identical with respect to their k -level degree composition. In order to turn this result into a hard instance for approximating the maximum matching size, we need the additional property that both graphs are high girth (particularly $\geq 2k + 2$).

Theorem 11. *For every graph $G = (V, E)$, $|V| = n$, every integer $g \geq 1$, there exists an integer $R = R(n, g) = n^{O(g)}$ and a graph $L = (V_L, E_L)$, $V_L = V \times [R]$, such that the following conditions hold.*

- (1) *Size of the maximum matching of L is multiplicatively close to that of G : $R \cdot MM(G) \leq MM(L) \leq 2R \cdot MM(G)$;*

(2) L contains no cycle shorter than g (i.e., L has high girth);

(3) For every $k \in \mathbb{N}$ and every $v \in V$ one has, for all $r \in R$ that $d_k^H((v, r)) = d_k^G(v)$.

We refer to L as the lift of G .

Before proving this theorem, let us state a key lemma that we use in proving it. The proof of this lemma is deferred to [Appendix D.2](#).

Lemma 23. For any parameters g and l , there exists a group \mathcal{G} of size $l^{O(g)}$ along with a set of generator elements S of size at least l , such that the associated Cayley graph ([Definition 12](#)) has girth at least g .

Equivalently, this means that no irreducible sequence of elements from S and their inverses, shorter than g , equates to the identity. We are now ready to prove [Theorem 11](#).

Proof of Theorem 11. Let \mathcal{G} be a group according to [Lemma 23](#) with parameter $l = |E|$, and let $S \subseteq \mathcal{G}$ denote the set of elements of \mathcal{G} whose Cayley graph has girth at least g , as guaranteed by [Lemma 23](#). We think of the elements of S as indexed by the edges of the graph G , and write $S = (s_e)_{e \in E}$. We direct the edges of G arbitrarily, and define the edge-set E_L of L as

$$E_L \stackrel{\text{def}}{=} \{((v_1, g_1), (v_2, g_2)) \in V_L \times V_L \mid e = (v_1, v_2) \in E \text{ and } g_1 \cdot s_e = g_2\}.$$

That is we connect vertices $(v_1, g_1), (v_2, g_2) \in V_L$ by an edge if and only if $e = (v_1, v_2)$ is an (directed) edge in E and $g_1 \cdot s_e = g_2$ in \mathcal{G} . In this construction $R = |\mathcal{G}| = m^{O(g)} = n^{O(g)}$ as stated in the theorem.

Note that every vertex v in the original graph G corresponds to an independent set of R vertices $v \times \mathcal{G}$ in L . For any pair of vertices v_1 and v_2 in the original graph if $(v_1, v_2) \in E$, then the subgraph induced by $(v_1 \times \mathcal{G}) \cup (v_2 \times \mathcal{G})$ is a perfect matching; if not, the union $(v_1 \times \mathcal{G}) \cup (v_2 \times \mathcal{G})$ forms an independent set. Overall, every edge $e = (v_1, v_2) \in E$ can be naturally mapped to R edges among the edges of L , namely

$$\{((v_1, g_1), (v_2, g_1 \cdot s_e)) \in V_L \times V_L \mid g_1 \in \mathcal{G}\}.$$

Property (1). Any matching M of G can be converted into a matching M^L of size $R|M|$ in L , for instance $M^L = \{((v_1, g_1), (v_2, g_2)) \in E_L \mid (v_1, v_2) \in M \text{ and } g_1 \in \mathcal{G}\}$. As mentioned above, g_2 is uniquely defined here. The same statement is true for vertex covers: If V is a vertex cover in G , then $V \times \mathcal{G}$ is a vertex cover in L . Therefore,

$$R \cdot \text{MM}(G) \leq \text{MM}(L) \leq \text{VC}(L) \leq R \cdot \text{VC}(G) \leq 2R \cdot \text{MM}(G),$$

where the last inequality is due to the fact that the minimum vertex cover of a graph is at most twice the size of its maximum matching.

Property (2). Toward contradiction, suppose C is a short cycle in L , that is it has length $f < g$. Let C be

$$((v_1, g_1)(v_2, g_2), \dots, (v_h, g_h)(v_{f+1}, g_{f+1}) = (v_1, g_1)).$$

Let $e_i := ((v_i, g_i), (v_{i+1}, g_{i+1})) \in E_L$. Let ϵ_i be a ± 1 variable indicating whether the direction of the edge e_i is towards v_{i+1} ($\epsilon_i = 1$) or towards v_i ($\epsilon_i = -1$). (Recall that the edges of G were arbitrarily directed during the construction of L .)

If C as described above is indeed a cycle, this means that $g_1 \cdot s_{e_1}^{\epsilon_1} \cdot s_{e_2}^{\epsilon_2} \cdot \dots \cdot s_{e_f}^{\epsilon_f} = g_1$. Therefore $s_{e_1}^{\epsilon_1} \cdot s_{e_2}^{\epsilon_2} \cdot \dots \cdot s_{e_f}^{\epsilon_f}$ is an irreducible sequence of elements from S and their inverses shorter than g that equates to unity. Indeed, if it was not irreducible, that is an element and its inverse appeared consecutively, then that would mean C crossed an edge twice consecutively ($e_i = e_{i+1}$ for some i) and therefore it would not be a true cycle. This is a contradiction of theorem [Lemma 23](#), so L must have girth at least g .

Property (3). The third statement is proven by induction on k . The **base case** is provided by $k = 1$. Fix v and $h \in \mathcal{G}$. Every neighbor of v in G corresponds to exactly one neighbor of (v, h) in L . Indeed, $w \in N(v)$ (such that $e = (v, w) \in E$) corresponds to $(w, h \cdot s_e^\epsilon)$ where ϵ indicates the direction of e . Therefore $d^G(v) = d^L((v, h))$.

We now show the **inductive step** ($k-1 \rightarrow k$). Again, fix v and h . Similarly to the base case, every neighbor $w \in N(v)$ corresponds to a single neighbor of (v, h) in L : (w, h_w) for some $h_w \in \mathcal{G}$. By the inductive hypothesis $d_{k-1}^G(w) = d_{k-1}^L((w, h_w))$. So

$$d_k^G(v) = \biguplus_{w \in N(v)} \{d_{k-1}^G(w)\} = \biguplus_{w \in N(v)} \{d_{k-1}^L((w, h_w))\} = d_k^L((v, h))$$

□

Thus, there exists a pair of graphs G_1 and G_2 such that there is a bijection between their vertex-sets that preserves high level degrees up to level k and such that neither G_1 nor G_2 contains a cycle shorter than $2k+2$. Furthermore, $\text{MM}(G_1)$ and $\text{MM}(G_2)$ differ by a factor of at least $\frac{c+1}{8k}$, which can be set to be arbitrarily high by the choice of c .

Corollary 8. *For $c \geq 2k$, there exists a pair of graphs G and H with vertex sets V and W , respectively, such that there is a bijection $\Phi : V \rightarrow W$ with the property that the k -depth neighborhoods of v and $\Phi(v)$ are isomorphic. Also, $\text{MM}(G) \geq \frac{c+1}{8k} \text{MM}(H)$.*

Proof. This follows directly from [Corollary 7](#), [Theorem 11](#) and [Observation 3](#). Indeed, consider graphs G' and H' guaranteed by [Corollary 7](#) with the same parameters. Apply to each [Theorem 11](#) to get the lifted graphs G and H respectively. The bijection Φ guaranteed in [Corollary 7](#) extends naturally to G and H . By the guarantee of [Theorem 11](#) this bijection still preserves k -level degrees, and furthermore, both G and H have girth at least $2k+2$. By [Observation 3](#) this is sufficient to show [Corollary 8](#). □

8.5 k -Edge Subgraph Statistics in G and H

The main result of this section is the equality of numbers of subgraphs in G and H . This will result in proving [Theorem 10](#).

Definition 14 (Subgraph counts). *For a graph $G = (V, E)$ and any graph K we let*

$$\#(K : G) = |\{U \subseteq E \mid U \cong K\}|,$$

where we write $U \cong K$ to denote the condition that U is isomorphic to K .

Lemma 24. *Let $k \geq 1$ be an integer and let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ be two graphs such that a bijection $\Phi : V_G \rightarrow V_H$ between their vertex-sets preserves the k -depth neighborhoods. That is, for every $v \in V_G$, the k -depth neighborhood of v is isomorphic to that of $\Phi(v)$. Then for any graph $K = (V_K, E_K)$ of at most k edges $\#(K : G) = \#(K : H)$.*

Proof. We prove below that if

$$\alpha_K := |\{\Psi : V_K \hookrightarrow V_G \mid \forall (u, w) \in E_K : (\Psi(u), \Psi(w)) \in E_G\}|$$

and

$$\beta_K := |\{\Psi : V_K \hookrightarrow V_H \mid \forall (u, w) \in E_K : (\Psi(u), \Psi(w)) \in E_H\}|,$$

then $\alpha_K = \beta_K$. Since $\alpha_K = \#(K : G) \cdot |\text{Aut}(K)|$ and $\beta_K = \#(K : H) \cdot |\text{Aut}(K)|$, where $|\text{Aut}(K)|$ is the number of automorphisms of K , then result then follows.

We proceed by induction on the number of connected components q in K . We start with the **base case** ($q = 1$), which is when K is connected, i.e. K has one connected component. Select arbitrarily a root $r \in V_K$ of K . Define further, for all $v \in V_G$ and $v \in V_H$ respectively

$$\#(K : G|v) = |\{\Psi : V_K \hookrightarrow V_G \mid \forall (u, w) \in E_K : (\Psi(u), \Psi(w)) \in E_G \wedge \Psi(r) = v\}|$$

and

$$\#(K : H|v) = |\{\Psi : V_K \hookrightarrow V_H \mid \forall (u, w) \in E_K : (\Psi(u), \Psi(w)) \in E_H \wedge \Psi(r) = v\}|$$

so that $\#(K : G) = \sum_{v \in V_G} \#(K : G|v)$ and $\#(K : H) = \sum_{v \in V_H} \#(K : H|v)$. Recall that there exists a bijection $\Phi : V_G \rightarrow V_H$ such that for every $v \in V_G$ the k -depth neighborhood of v in G is identical to the k -depth neighborhood of $\Phi(v)$ in H . Since the number of edges in K is at most k , and K is connected, we get that $\#(K : G|v) = \#(K : H|\Phi(v))$, and hence

$$\#(K : G) = \sum_{v \in V_G} \#(K : G|v) = \sum_{v \in V_G} \#(K : H|\Phi(v)) = \sum_{v \in V_H} \#(K : H|v) = \#(K : H),$$

as required.

We now provide the **inductive step** ($q - 1 \rightarrow q$). Since $q \geq 2$, we let $K_1 = (V_{K_1}, E_{K_1})$ and $K_2 = (V_{K_2}, E_{K_2})$ be a bipartition of K into two disjoint non-empty subgraphs, i.e., V_K is the disjoint union of V_{K_1} and V_{K_2} and E_K is the disjoint union of E_{K_1} and E_{K_2} . Since the number of components of K_1 and K_2 are both smaller than q , we have by the inductive hypothesis that $\alpha_{K_1} = \beta_{K_1}$ and $\alpha_{K_2} = \beta_{K_2}$.

We will write the number of embeddings of K into G (resp. H) in terms of the number of embeddings of K_1, K_2 into G (resp. H), as well as embeddings of natural derived other graphs. Indeed, every pair of embeddings (Ψ_1, Ψ_2) , where $\Psi_i : V_{K_i} \hookrightarrow V_G, i \in \{1, 2\}$, naturally defines a mapping $\Psi : V_F \rightarrow V_G$. However, this mapping is not necessarily injective. Indeed $\Psi_1(v_1)$ might clash with $\Psi_2(v_2)$ for some pairs $(v_1, v_2) \in V_{K_1} \times V_{K_2}$. In this case Φ defines different graph K' which we get by merging all clashing pairs from $V_{K_1} \times V_{K_2}$, along with an embedding of K' into G . Note that K' has strictly fewer than q components. We call such a pair (Ψ_1, Ψ_2) an K' -clashing pair. We now get

$$\begin{aligned}
\alpha_K &= |\{\text{embedding } \Psi \text{ of } K \text{ into } G\}| \\
&= |\{(\Psi_1, \Psi_2) | \text{embeddings of } K_i \text{ into } G, i \in \{1, 2\}\}| \\
&\quad - \sum_{\substack{K' \text{ graph with} \\ < q \text{ components}}} |\{(\Psi_1, \Psi_2) | K' - \text{clashing embeddings of } K_i, i \in \{1, 2\} \text{ into } G\}| \\
&= |\{(\Psi_1, \Psi_2) | \text{embeddings of } K_i \text{ into } H, i \in \{1, 2\}\}| \\
&\quad - \sum_{\substack{K' \text{ graph with} \\ < q \text{ components}}} |\{(\Psi_1, \Psi_2) | K' - \text{clashing embeddings of } K_i, i \in \{1, 2\} \text{ into } H\}| \\
&= \beta_K,
\end{aligned}$$

where in the third transition above we used the fact that for any K' with $< q$ connected components one has

$$\begin{aligned}
&|\{(\Psi_1, \Psi_2) | K' - \text{clashing embeddings of } K_i, i \in \{1, 2\} \text{ into } G\}| \\
&= |\{\Psi' | \text{embeddings of } K' \text{ into } G\}| \\
&= |\{\Psi' | \text{embeddings of } K' \text{ into } H\}| \quad (\text{by inductive hypothesis}) \\
&|\{(\Psi_1, \Psi_2) | K' - \text{clashing embeddings of } K_i, i \in \{1, 2\} \text{ into } H\}|.
\end{aligned}$$

This completes the proof. \square

Corollary 9 ([Theorem 10](#)). *For every $\lambda > 1$ and every k , there exist graphs G and H such that $MM(G) \geq \lambda \cdot MM(H)$, but for every graph K with at most k edges, the number of subgraphs of G and H isomorphic to K are equal.*

Proof. By [Lemma 24](#) the pair of graphs satisfying the guarantee of [Corollary 8](#) will satisfy the guarantees of [Theorem 10](#) as well. We just need to set c such that $c \geq 2k$ and $\frac{c+1}{8(k+1)} \geq \lambda$. \square

9 Analysis of the algorithm on a random permutation stream

9.1 Introduction and Technical Overview

In this section we focus on the setting in which the set of elements, e.g., edges, is presented as a random permutation. This has been a popular model of computation for graph algorithms in recent years with many results, including for matching size approximation [[KMM12](#), [KKS14](#), [MMPS17](#), [PS18](#), [ABB⁺19](#)]. As mentioned before, our goal is to show that [Algorithm 2](#) is robust to the correlations introduced by replacing independent samples with a random permutation. This results in algorithm for approximating matching size to within a factor of $O(\log^2 n)$, in polylogarithmic space.

Theorem 4. *There exists an algorithm that given access to a random permutation edge stream of a graph $G = (V, E)$, with n vertices and $m \geq 3n$ edges, produces an $O(\log^2 n)$ factor approximation to maximum matching size in G using $O(\log^2 n)$ bits of memory in a single pass over the stream with probability at least $3/4$.*

Recall that this improves the previous best-known approximation ratio ([KKS14]) by at least a factor $O(\log^6 n)$.

Our overall strategy consists of showing that the algorithm behaves identically when applied to iid samples or a permutation. That is, we show that distribution of the state of the algorithm at each moment is similar under these two settings. To this end, we use total variation distance and KL-divergence as a measure of similarity of distributions. Furthermore, we break down the algorithm to the level of LEVEL- j -TEST tests to show that these behave similarly.

More specifically, consider an invocation of LEVEL- j -TEST(v) in either the iid or the permutation stream. In the permutation stream, we have already seen edges pass and therefore know that they will not reappear during the test; this biases the output of the test compared to the iid version which is oblivious to the prefix of the stream. However, we are able to prove in Section 9.4 that KL-divergence between the output-distribution of these two versions is proportional to the number of samples used, with a factor of $O(\log^2 n/m)$, see Lemma 30. Since this is true for all tests, intuitively, the algorithm should still work in the permutation setting as long as it uses $O(m/\log^2 n)$ samples. Conversely, our original algorithm uses $\Theta(m)$ samples; however, we can reduce the number of samples used by slightly altering it, at the expense of a $O(\log^2 n)$ factor in the approximation ratio.

In fact the algorithm we use in permutation stream setting is nearly identical to the one defined in Section 3; the one difference is that in the subroutine EDGE-LEVEL-TEST we truncate the number of tests to $J - \Omega(\log^2 n)$ to reduce the number of edges used, (see Algorithm 16 in Algorithm 16).

One technical issue that arises in carrying out this approach is the fact that KL-divergence does not satisfy the triangle inequality, not even an approximate one, when the distributions of the random variables in question can be very concentrated. (Or, equivalently, we can assume very small values, since we are thinking of Bernoulli variables). Essentially triangle inequality is not satisfied even approximately when some of the random variables in question are nearly deterministic (see Remark 2 in Section 9.2).

We circumvent this problem using a mixture of total variation and KL-divergence bounds. Furthermore, we develop a weaker version of triangle inequality for KL-divergence, see Lemma 27. This both loses a constant factor in the inequality and is only true under the condition that the variables involved are bounded away from deterministic. However, it suffices for our proofs in Sections 9.3 and 9.4.

Our proof strategy consists of two steps: we first modify the tests somewhat to ensure that all relevant variables are not too close to deterministic (see the padded tests presented in Section 9.3), at the same time ensuring that the total variation distance to the original tests is very small (see Lemma 28 in Section 9.3). We then bound the KL-divergence between the padded tests on the iid and permutation stream in Section 9.4. An application of Pinsker's inequality then completes the proof.

9.2 Preliminaries

Throughout this section, for sake of simplicity, we use n as an upperbound on the maximum degree d of G . This does not affect the guarantees that we provide.

Next we provide some notations that we will use in the sequel. Let Π be the random variable describing the permutation stream. Let the residual graph at time t be G^t , that is the original graph, but with the edges that have already appeared in the stream up until time t deleted, for $0 \leq t \leq m$. In this case G^0 is simply the original input graph and G^m is the empty graph of n isolated vertices. Let the residual degree of v at time T be

$$d^t \stackrel{\text{def}}{=} \frac{d^{G^t}}{1 - t/m}.$$

Let the outcome of a j^{th} level test on vertex v (recall LEVEL- j -TEST from Algorithm 3) be $T_j^{\text{IID}}(v)$. Let of the same test on the permutation stream, performed at time t will be denoted $T_j^{\pi,t}(v)$.

Definition 15 (KL-divergence). *For two distributions P and Q over \mathcal{X} , the KL-divergence of P and Q is*

$$D_{KL}(P\|Q) = \sum_{x \in \mathcal{X}} -P(x) \log \left(\frac{Q(x)}{P(x)} \right).$$

Remark 1. *Throughout this document, \log is used to denote the natural logarithm in the definition of KL-divergence, even though it is conventionally the base 2 logarithm. This only scales down $D_{KL}(P\|Q)$ by a factor of $\log 2$ and does not affect any of the proofs.*

Lemma 25 (Chain rule). For random vectors $P = (P_i)_i$ and $Q = (Q_i)_i$,

$$D_{KL}(P\|Q) = \sum_i \mathbb{E}_{x_1, \dots, x_{i-1}} [D_{KL}(P_i\|Q_i|x_1, \dots, x_{i-1})]$$

Lemma 26. For every $p, \epsilon \in [0, 1]$ such that $p + \epsilon \in [0, 1]$ one has

$$D_{KL}(\text{Ber}(p + \epsilon)\|\text{Ber}(p)) = \frac{16\epsilon^2}{p(1-p)}$$

A proof of [Lemma 26](#) is provided in [Appendix E](#).

Remark 2. The triangle inequality does not hold for KL-divergence, not even with a constant factor loss, as the following example shows. For sufficiently small ϵ , let $A = \text{Ber}(\epsilon)$, $B = \text{Ber}(\epsilon^2)$ and $C = \text{Ber}(\epsilon^{1/\epsilon})$. One can verify that $D_{KL}(A\|B) \leq \epsilon$, $D_{KL}(B\|C) \leq \epsilon$, but $D_{KL}(A\|C) \geq \omega(\epsilon)$. Nevertheless, we provide a restricted version of triangle inequality in [Lemma 27](#) that forms the basis of our analysis.

Definition 16 (θ -padding of a Bernoulli random variable). We define the padding operation as follows. Given a Bernoulli random variable X and a threshold $\theta \in (0, 1)$ we let

$$\text{PADDING}(X, \theta) \stackrel{\text{def}}{=} \begin{cases} X & \text{if } \mathbb{E}[X] \in [\theta, 1 - \theta] \\ \text{Ber}(\theta) & \text{if } \mathbb{E}[X] < \theta \\ \text{Ber}(1 - \theta) & \text{if } \mathbb{E}[X] > 1 - \theta \end{cases}$$

Lemma 27 (Triangle inequality for padded KL-divergence). Suppose $p, q, r \in [0, 1]$ and consider the KL-divergence between Bernoulli variables $\text{Ber}(p)$, $\text{Ber}(q)$ and $\text{Ber}(r)$. Then for some absolute constant C , any $\epsilon \in [0, 1/32]$, if $D_{KL}(\text{Ber}(p)\|\text{Ber}(q)) \leq \epsilon$ and $D_{KL}(\text{Ber}(q)\|\text{Ber}(r)) \leq \epsilon$, then

$$D_{KL}(\text{Ber}(p)\|\text{Ber}(\text{PADDING}(r, \epsilon))) \leq C\epsilon.$$

The proof is provided in [Appendix E](#).

Remark 9. We note that the padding is crucial, due to the example in [Remark 2](#).

9.3 Padding and total variation distance

In this and the next section we compare the behavior of the LEVEL- j -TEST's on the iid and permutation streams. Recall the definition of LEVEL- $j + 1$ -TEST from [Algorithm 4](#) in [Section 3](#). We call this the T_{j+1}^{IID} test and restate it here for completeness:

Algorithm 11 Vertex test in the i.i.d. stream

```

1: procedure  $T_{j+1}^{\text{IID}}(v)$ 
2:    $S \leftarrow 0$ 
3:   for  $k = 1$  to  $c^j \cdot \frac{m}{n}$  do
4:      $e \leftarrow$  next edge in the iid stream ▷ Equivalent to sampling and iid edge
5:     if  $e$  is adjacent to  $v$  then
6:        $w \leftarrow$  the other endpoint of  $e$ 
7:        $i \leftarrow 0$ 
8:       while  $i \leq j$  and  $T_i^{\text{IID}}(w)$  do
9:          $S \leftarrow S + c^{i-j}$ 
10:      if  $S \geq \delta$  then
11:        return FALSE
12:       $i \leftarrow i + 1$ 

```

Similarly, define the version of the T_{j+1} tests on the permutation stream, starting at position t , which we call $T_{j+1}^{\pi, t}$ as follows:

Algorithm 12 Vertex test in the permutation stream

```
1: procedure  $T_{j+1}^{\pi,t}(v)$ 
2:    $S \leftarrow 0$ 
3:   for  $k = 1$  to  $c^j \cdot \frac{m}{n}$  do
4:      $e \leftarrow$  next edge in the permutation stream  $\triangleright$  We start using the stream from the  $t + 1^{\text{th}}$  edge
5:     if  $e$  is adjacent to  $v$  then
6:        $w \leftarrow$  the other endpoint of  $e$ 
7:        $i \leftarrow 0$ 
8:       while  $i \leq j$  and  $T_i^{\pi}(w)$  do
9:          $S \leftarrow S + c^{i-j}$ 
10:        if  $S \geq \delta$  then
11:          return FALSE
12:         $i \leftarrow i + 1$ 
13:   return True
```

We also define recursively padded versions of T_{j+1}^{IID} define recursively

Algorithm 13 Padded T_1 test

```
1: procedure  $\tilde{T}_1^{\text{IID}}(v)$ 
2:   return  $T_1^{\text{IID}}(v)$ .
```

Algorithm 14 Recursively padded T_{j+1} tests

```
1: procedure  $\bar{T}_{j+1}^{\text{IID}}(v)$ 
2:    $S \leftarrow 0$ 
3:   for  $c^j m/n$  edges  $e$  of the iid stream do
4:     if  $e$  is adjacent to  $v$  then
5:        $w \leftarrow$  the other endpoint of  $e$ .
6:        $i \leftarrow 0$ 
7:       while  $i \leq j$  and  $\tilde{T}_i^{\text{IID}}(w)$  do
8:          $S \leftarrow S + c^{i-j}$ 
9:         if  $S \geq \delta$  then
10:          return FALSE
11:        $i \leftarrow i + 1$ 
12:   return True
```

and, using [Definition 16](#),

Algorithm 15 Recursively padded T_{j+1} tests

```
1: procedure  $\tilde{T}_{j+1}^{\text{IID}}(v)$ 
2:   return PADDING( $\bar{T}_{j+1}^{\text{IID}}(v)$ ,  $\frac{200c^j \log^2 n}{n}$ )
```

Note that these alternate tests are not implementable and merely serve as a tool to proving that the T^{π} tests work similarly to the T^{IID} tests.

We begin by proving that padding the T^{IID} tests, even recursively, only changes the output with probability proportional to the fraction of the stream (of length m) consumed by the test.

Lemma 28. *For sufficiently small $\delta > 0$ and large enough c the following holds. For all $v \in V$ and $j = [0, J]$ one has*

$$\left\| T_{j+1}^{\text{IID}}(v) - \tilde{T}_{j+1}^{\text{IID}}(v) \right\|_{TV} \leq \frac{400 \cdot c^j \log^2 n}{n}.$$

Proof. The proof follows by triangle inequality of total variation distance: We will prove the following

bounds:

$$\left\| T_{j+1}^{\text{IID}}(v) - \bar{T}_{j+1}^{\text{IID}}(v) \right\|_{\text{TV}} \leq \frac{200 \cdot c^j \log^2 n}{n} \quad (31)$$

$$\left\| \bar{T}_{j+1}^{\text{IID}}(v) - \tilde{T}_{j+1}^{\text{IID}}(v) \right\|_{\text{TV}} \leq \frac{200 \cdot c^j \log^2 n}{n}. \quad (32)$$

Eq. (32) follows easily from definitions, since $\tilde{T}_{j+1}^{\text{IID}}(v)$ is $200c^j \log^2(n)/n$ -padded version of $\bar{T}_{j+1}^{\text{IID}}(v)$. We proceed to proving Eq. (31).

Consider the following $0, 1$ vector $W_{j+1}(v)$ describing the process of a $T_{j+1}(v)$ test: The first $c^j m/n$ coordinates denote the search phase; specifically we write a 1 if a neighbor was found and a 0 if not. The following coordinates denote the outcomes of the recursive tests, 1 for pass 0 for fail, in the order they were performed. There could be at most c^j recursive tests performed (if they were all T_1 's) so $W_{j+1}(v)$ has length $c^j m/n + c^j$ in total. However, often much fewer recursive tests are performed due to the early stopping rule, or simply because too few neighbors of v were found. In this case, tests not performed are represented by a 0 in $W_{j+1}(v)$.

Like with $T_{j+1}(v)$, we will define different versions of $W_{j+1}(v)$, namely $W_{j+1}^{\text{IID}}(v)$ and $\bar{W}_{j+1}^{\text{IID}}(v)$. Since $W_{j+1}(v)$ determines $T_{j+1}(v)$ we can simply bound $\left\| W_{j+1}^{\text{IID}}(v) - \bar{W}_{j+1}^{\text{IID}}(v) \right\|_{\text{TV}}$.

The first $c^j m/n$ coordinates of W^{IID} and \bar{W}^{IID} are distributed identically and contribute nothing to the divergence. Consider the test corresponding to the i^{th} coordinate of the recursive phase of $W_{j+1}(v)$. Let the level of the test be $\ell_i \in [0, j]$ (with 0 representing no test) and let the vertex of the test be u_i . These are random variables determined by Π . Furthermore, let t_i be the position in the stream where the recursive T_{ℓ_i} test is called on u_i . Then we have

$$\begin{aligned} \left\| T_{j+1}^{\text{IID}}(v) - \bar{T}_{j+1}^{\text{IID}}(v) \right\|_{\text{TV}} &\leq \left\| W_{j+1}^{\text{IID}}(v) - \bar{W}_{j+1}^{\text{IID}}(v) \right\|_{\text{TV}} \\ &= \sum_i \mathbb{E}_{\ell_i, u_i} \left\| (W_{j+1, i}^{\text{IID}}(v) | \ell_i, u_i) - (\bar{W}_{j+1, i}^{\text{IID}}(v) | \ell_i, u_i) \right\|_{\text{TV}} \\ &= \sum_i \mathbb{E}_{\ell_i, u_i} \left\| T_{\ell_i}^{\text{IID}}(u_i) - \tilde{T}_{\ell_i}^{\text{IID}}(u_i) \right\|_{\text{TV}} \\ &\leq \sum_i \mathbb{E}_{\ell_i} \left(\frac{400 c^{\ell_i-1} \log^2 n}{n} \right), \end{aligned}$$

by the inductive hypothesis. Here $(W_{j+1, i}(v) | \ell_i, u_i)$ denotes conditional distribution of $W_{j+1, i}(v)$ on ℓ_i and u_i .

Note that the term in the sum is proportional to the number of edges used by the corresponding recursive test. Indeed, recall by Lemma 2 that a T_{ℓ_i} test runs for at most $c^{\ell_i-1} \cdot \frac{m}{n} \cdot (1 + 2\delta)$ samples with probability one. It is crucial that this result holds with probability one, and therefore extends to not just the iid stream it was originally proven on, but any arbitrary stream of edges.

Therefore, the term in the sum above is at most $200C \cdot c \log^2(n)/m$ times the number of edges used by the corresponding recursive test. So the entire sum is at most $400 \log^2(n)/m$ times the length of the recursive phase of the original T_{j+1} test. It was also derived in Lemma 2 that the recursive phase itself takes at most $c^j \cdot \frac{m}{n} \cdot 2\delta$ edges. (Again, the result holds with probability one.) Therefore,

$$\left\| T_{j+1}^{\text{IID}}(v) - \tilde{T}_{j+1}^{\text{IID}}(v) \right\|_{\text{TV}} \leq \frac{400 \log^2 n}{m} \cdot \frac{2\delta c^j m}{n} \leq \frac{400 \cdot c^j \log^2 n}{n},$$

for small enough absolute constant δ . This shows Eq. (31) and concludes the proof. \square

9.4 Bounding KL-divergence

In this section we will bound the KL-divergence between tests on the permutation stream and padded tests on the iid stream. We will use the padded triangle inequality of Lemma 27 as well as the data processing inequality for kl-divergence:

Lemma 29. (Data Processing Inequality) *For any random variables X, Y and any function f one has $D_{KL}(f(X) || f(Y)) \leq D(X || Y)$.*

Lemma 30. For sufficiently small $\delta > 0$ and large enough c the following holds. With high probability over Π , for all $v \in V$, $j \leq J$, $t \leq m/2 - 2c^j \cdot \frac{m}{n}$,

$$D_{KL} \left(T_{j+1}^{\pi,t}(v) \middle\| \tilde{T}_{j+1}^{\text{IID}}(v) \middle| G^t \right) \leq \frac{200C \cdot c^j \log^2 n}{n}, \quad (33)$$

where C is the constant from [Lemma 27](#).

Note that the choice of t is such that we guarantee the T_{j+1} tests finishing before half of the stream is up, by [Lemma 2](#).

Naturally, we will prove the above lemma by induction on j . Specifically, our inductive hypothesis will be that [Eq. \(33\)](#) holds up to some threshold j . Furthermore, recall that [Algorithm 15](#) satisfies

$$\tilde{T}_{j+1}^{\text{IID}}(v) = \text{PADDING} \left(\bar{T}_{j+1}^{\text{IID}}(v), \frac{200 \cdot c^j \log^2 n}{n} \right).$$

Thus, to bound $D_{KL} \left(T_{j+1}^{\pi,t}(v) \middle\| \tilde{T}_{j+1}^{\text{IID}}(v) \middle| G^t \right)$ we can apply [Lemma 27](#) directly. By setting $\text{Ber}(p)$ to $(T_{j+1}^{\pi,t}(v) | G^t)$ and $\text{Ber}(r)$ to $\bar{T}_{j+1}^{\text{IID}}(v)$ we get that $\tilde{T}_{j+1}^{\text{IID}}(v) \sim \text{Ber}(\tilde{r})$. So in order to bound $D_{KL}(\text{Ber}(p) \| \text{Ber}(\tilde{r}))$ as required by the lemma, we need only to bound $D_{KL}(\text{Ber}(p) \| \text{Ber}(q))$ and $D_{KL}(\text{Ber}(q) \| \text{Ber}(r))$.

Our q , the midpoint of our triangle inequality, will be the outcome of a newly defined hybrid test using both the permutation and iid streams.

```

1: procedure  $T_{j+1}^{\chi,t}(v)$ 
2:    $S \leftarrow 0$ 
3:   for  $k = 1$  to  $c^j \cdot \frac{m}{n}$  do
4:      $e \leftarrow$  next edge in the permutation stream  $\triangleright$  We start using the stream from the  $t + 1^{\text{th}}$  edge
5:     if  $e$  is adjacent to  $v$  then
6:        $w \leftarrow$  the other endpoint of  $e$ 
7:        $i \leftarrow 0$   $\triangleright$  Represents the last level that  $w$  passes
8:       while  $i \leq j$  and  $\tilde{T}_i^{\text{IID}}(w)$  do
9:          $S \leftarrow S + c^{i-j}$ 
10:        if  $S \geq \delta$  then
11:          return FALSE
12:         $i \leftarrow i + 1$ 
13:   return True

```

We will begin by bounding $D_{KL} \left(T_{j+1}^{\chi,t}(v) \middle\| \bar{T}_{j+1}^{\text{IID}}(v) \middle| G^t \right)$

Lemma 31. For sufficiently small $\delta > 0$ and large enough c the following holds. With probability $1 - n^{-5}$, for all, $v \in V$, $j \leq J$, $t \leq m/2 - 2c^j \cdot \frac{m}{n}$

$$D_{KL} \left(T_{j+1}^{\chi,t}(v) \middle\| \bar{T}_{j+1}^{\text{IID}}(v) \middle| G^t \right) \leq \frac{200 \cdot c^j \cdot \log^2 n}{n}. \quad (34)$$

Proof. We will deconstruct $T_{j+1}(v)$ a little differently than before, in the proof of [Lemma 28](#). Consider the following integer vector, $Y_{j+1}(v)$, describing the process of $T_{j+1}(v)$: There are $c^j m/n$ coordinates in total, with the i^{th} coordinate corresponding to the i^{th} edge sampled from the stream. If this edge is not adjacent to v the coordinate is 0. If it is adjacent, with its other endpoint being u_i , the algorithm performs higher and higher level tests on u_i until it fails or the level exceeds j . So let the i^{th} coordinate of $Y_{j+1}(v)$ simply denote the level ℓ_i at which $T_{\ell_i}(u_i)$ failed, or $j + 1$ if the vertex never failed.

Like with $T_{j+1}(v)$, we will define different versions of $Y_{j+1}(v)$, namely $Y_{j+1}^{\chi,t}(v)$ and $\bar{Y}_{j+1}^{\text{IID}}(v)$. Note that $Y_{j+1}(v)$ determines $T_{j+1}(v)$ and it suffices to bound $D_{KL} \left(Y_{j+1}^{\chi,t}(v) \middle\| \bar{Y}_{j+1}^{\text{IID}}(v) \middle| G^t \right)$ by the data processing inequality ([Lemma 29](#)).

$$\begin{aligned} D_{KL} \left(T_{j+1}^{\chi,t}(v) \middle\| \bar{T}_{j+1}^{\text{IID}}(v) \middle| G^t \right) &\leq D_{KL} \left(Y_{j+1}^{\chi,t}(v) \middle\| \bar{Y}_{j+1}^{\text{IID}}(v) \middle| G^t \right) \\ &= \sum_i \mathbb{E}_{G^t} D_{KL} \left(Y_{j+1,i}^{\chi,t}(v) \middle\| \bar{Y}_{j+1,i}^{\text{IID}}(v) \middle| G^t \right), \end{aligned}$$

where G^{t_i} represents the residual graph right after we sample the i^{th} edge for the T_{j+1} test (that is i^{th} excluding edges sampled during recursion). Consider the distribution of $Y_{j+1,i}^{\chi,t}(v)$ and $\bar{Y}_{j+1,i}^{\text{IID}}(v)$. In both cases the recursive tests run would use the iid stream. Consider for each neighbor of v running all iid tests $\bar{T}_\ell^{\text{IID}}$ for ℓ from 1 to j .

Let $\psi : E \rightarrow [0, j+1]$ be the following random mapping: If e is not adjacent on v then $\psi(e) = 0$. If $e = (u, v)$, $\psi(e)$ is distributed as the smallest level ℓ at which u would fail when running $\bar{T}_1^{\text{IID}}(u), \dots, \bar{T}_j^{\text{IID}}(u)$, and $j+1$ if it never fails. As described above, $\bar{Y}_{j+1,i}^{\text{IID}}(v)$ is distributed as $\psi(e) : e \sim U(E)$ and $Y_{j+1,i}^{\chi,t}(v)$ is distributed as $\psi(e) : e \sim U(E^{t_i})$. Here U represents the uniform distribution and E^{t_i} represents the residual edge-set.

$$\mathbb{E}_{G^{t_i}} D_{KL} \left(Y_{j+1,i}^{\chi,t}(v) \parallel \bar{Y}_{j+1,i}^{\text{IID}}(v) \mid G^{t_i} \right) = \mathbb{E}_{G^{t_i}, \psi} D_{KL} \left(\psi(e) : e \sim U(E^{t_i}) \parallel \psi(e) : e \sim U(E) \mid G^{t_i} \right)$$

We will now bound the right hand side with high probability over G^{t_i} . Fix v and j , thereby fixing the distribution of ψ . Define for every $k \in [0, j+1]$

$$\begin{aligned} p_k^{\text{IID}}(v) &:= \mathbb{P}_{e \sim U(E)}[\psi(e) = k] \\ p_k^{\chi,t_i}(v) &:= \mathbb{P}_{e \sim U(E^{t_i})}[\psi(e) = k]. \end{aligned} \tag{35}$$

We know by Chernoff bound that for every fixed choice of $v \in V$, $k \in [0, j+1]$ and $t_i \leq m/2 - 2c^{k-1} \cdot \frac{m}{n}$

$$|p_k^{\text{IID}}(v) - p_k^{\chi,t_i}(v)| \leq \sqrt{\frac{100p_k \log n}{m}} \tag{36}$$

with probability at least $1 - n^{-10}$ over the randomness of Π . Indeed

$$p_k^{\chi,t_i}(v) = \sum_{e \in E} \frac{\mathbb{1}(e \in G^{t_i}) \cdot \mathbb{P}(\psi(e) = k)}{m - t_i},$$

so $\mathbb{E} p_k^{\chi,t_i}(v) = p_k^{\text{IID}}(v)$ and p_k^{χ,t_i} is a sum of independent variables bounded by $2/m$. Therefore, by Chernoff bounds, specifically [Items a and c of Theorem 7](#),

$$\mathbb{P} \left[|p_k^{\chi,t_i}(v) - p_k^{\text{IID}}(v)| \geq \sqrt{\frac{100p_k^{\text{IID}} \log n}{m}} \right] \leq 2 \exp \left(\frac{100p_k^{\text{IID}} \log n}{3mp_k^{\text{IID}} \cdot (2/m)} \right) \leq n^{-10}.$$

Therefore, with probability at least $1 - n^{-5}$ this bound holds for all choices of v , k and t_i simultaneously. Let us restrict our analysis to this event from now on.

In the following calculation we denote $p_k^{\chi,t_i}(v)$ by \tilde{p}_k and $p_k^{\text{IID}}(v)$ by p_k for simplicity of notation. We have

$$D_{KL}(\psi(e) : e \sim U(E^{t_i}) \parallel \psi(e) : e \sim U(E) \mid \psi, G^{t_i}) = \sum_{k=0}^{j+1} -\tilde{p}_k \log \left(\frac{p_k}{\tilde{p}_k} \right) = \sum_{k=0}^{j+1} -\tilde{p}_k \log \left(1 + \frac{p_k - \tilde{p}_k}{\tilde{p}_k} \right).$$

Note that because $t_i \leq m/2$, $\tilde{p} \in [0, 2p_k]$, so $(p_k - \tilde{p}_k)/\tilde{p}_k$ is in the range $[-1/2, \infty]$. In the range $x \in [-1/2, \infty]$, $\log(1+x)$ can be lower bounded by $x - x^2$. Therefore, the above sum can be upper bounded as follows:

$$\begin{aligned} D_{KL}(\psi(e) : e \sim U(E^{t_i}) \parallel \psi(e) : e \sim U(E) \mid \psi, G^{t_i}) &\leq \sum_{k=0}^{j+1} -\tilde{p}_k \cdot \left(\frac{p_k - \tilde{p}_k}{\tilde{p}_k} - \frac{(p_k - \tilde{p}_k)^2}{\tilde{p}_k^2} \right) \\ &= \sum_{k=0}^{j+1} \left(\tilde{p}_k - p_k + \frac{(p_k - \tilde{p}_k)^2}{\tilde{p}_k} \right) \\ &\leq \sum_{k=0}^{j+1} \frac{\left(\sqrt{100\tilde{p}_k \log n/m} \right)^2}{\tilde{p}_k} && \text{due to Eq. (36),} \\ &\leq \sum_{k=0}^{j+1} \frac{100 \log n}{m} \\ &\leq \frac{200 \log^2 n}{m}. \end{aligned}$$

With this we can bound the whole sum, and thus $D_{KL} \left(T_{j+1}^X(v) \parallel \bar{T}_{j+1}^{\text{IID}}(v) \right)$.

$$\begin{aligned} D_{KL} \left(T_{j+1}^X(v) \parallel \bar{T}_{j+1}^{\text{IID}}(v) \right) &= \frac{c^j m}{n} \cdot \frac{200 \log^2 n}{m} \\ &= \frac{200 \cdot c^j \log^2 n}{n}, \end{aligned}$$

as claimed. \square

We will now proceed to bound the divergence between $T_1^{\pi,t}(v)$ and $T_1^{\text{IID}}(v)$. This will serve as the base case of the induction in the proof of [Lemma 30](#).

Lemma 32. *For sufficiently small $\delta > 0$ and large enough c the following holds. With probability $1 - n^{-5}$, for all $v \in V$, $t \leq m/2 - 2m/n$*

$$D_{KL} \left(T_1^{\pi,t}(v) \parallel T_1^{\text{IID}}(v) \mid G^t \right) \leq \frac{200C \cdot \log^2 n}{n}, \quad (37)$$

where C is the constant from [Lemma 27](#).

Proof. We will deconstruct the tests similarly to the previous proof. For both types of $T_1(v)$ test consider the random boolean vector $Y \in \{0, 1\}^{m/n}$ denoting whether the next m/n edges in the appropriate stream are adjacent to v or not. That is $Y_i = 1$ if and only if the i^{th} edge in the appropriate stream is adjacent to v . Let Y^π and Y^{IID} denote such random variables for $T_1^{\pi,t}(v)$ and $T_1^{\text{IID}}(v)$ respectively.

Because Y determines the outcome of $T_1(v)$, by data processing inequality ([Lemma 29](#)) one has

$$D_{KL} \left(T_1^{\pi,t}(v) \parallel T_1^{\text{IID}}(v) \mid G^t \right) \leq D_{KL} \left(Y^\pi \parallel Y^{\text{IID}} \mid G^t \right). \quad (38)$$

$$\begin{aligned} D_{KL} \left(Y^\pi \parallel Y^{\text{IID}} \mid G^t \right) &= \sum_{i=1}^{m/n} D_{KL} \left(Y_i^\pi \parallel Y_i^{\text{IID}} \mid G^t, Y_1^\pi, \dots, Y_{i-1}^\pi \right) \\ &\leq \sum_{i=1}^{m/n} D_{KL} \left(Y_i^\pi \parallel Y_i^{\text{IID}} \mid G^{t+i-1} \right) \\ &= \sum_{i=1}^{m/n} D_{KL} \left(\text{Ber} \left(d^{t+i-1}(v)/m \right) \parallel \text{Ber} \left(d(v)/m \right) \right). \end{aligned} \quad (39)$$

Recall that $d^t(v)$ is the residual degree of v at time t . By [Lemma 26](#) one has

$$D_{KL} \left(\text{Ber} \left(d^{t+i-1}(v)/m \right) \parallel \text{Ber} \left(d(v)/m \right) \mid G^{t+i-1} \right) \leq \frac{16 \left(\frac{d^{t+i-1}(v)}{m} - \frac{d(v)}{m} \right)^2}{\frac{d(v)}{m} \left(1 - \frac{d(v)}{m} \right)}, \quad (40)$$

and hence it suffices to upper bound the squared deviation of the residual degree $d^{t+i-1}(v)$ from $d(v)$. Note that we have $\mathbb{E}[d^t(v)] = d(v)$ for every v and t , and by Chernoff bounds the residual degree concentrates around its expectation. More specifically:

$$\mathbb{P} \left(|d^t(v) - d(v)| \geq \sqrt{100d(v) \log(n)} \right) \leq n^{-10}. \quad (41)$$

Let us constrain ourselves to the event of probability at least $1 - n^{-5}$ where this bound is satisfied for every t and v . That is one has $|d^t(v) - d(v)| \leq \sqrt{100d(v) \log(n)}$.

We thus have, using [Eqs. \(40\)](#) and [\(41\)](#) together with the fact that $1 \leq d(v) \leq m/3$

$$\begin{aligned} D_{KL} \left(\text{Ber} \left(d^{t+i-1}(v)/m \right) \parallel \text{Ber} \left(d(v)/m \right) \mid G^{t+i-1} \right) &\leq \frac{16 \left(\sqrt{100d(v) \log(n)} \right)^2}{m \cdot d(v) \left(1 - \frac{d(v)}{m} \right)} \\ &\leq \frac{1600 \cdot d(v) \log n}{m \cdot d(v) \cdot 2/3} \\ &= \frac{2400 \log n}{m}. \end{aligned} \quad (42)$$

(Note that we assumed $d(v) \leq n \leq m/3$.)

Therefore,

$$D_{KL}(Y^\pi \| Y^{\text{IID}} | G^t) \leq \frac{2400 \log n}{m} \cdot \frac{m}{n} = \frac{2400 \log n}{n},$$

which is stronger than the desired bound of Eq. (37). \square

We are finally ready to prove Lemma 30.

Proof of Lemma 30. First, let us constrain ourselves to the high probability event where Eqs. (34) and (37) hold from Lemmas 31 and 32. As mentioned before, we will proceed by induction on j . Our base case is simply Lemma 32 with a slight modification. Indeed, Lemma 32 bounds the divergence of $T_1^{\pi,t}(v)$ and $T_1^{\text{IID}}(v)$, whereas we need to bound the divergence of $T_1^{\pi,t}(v)$ and $\tilde{T}_1^{\text{IID}}(v)$. Note however, that $\tilde{T}_1^{\text{IID}}(v)$ is simply the padding of $\bar{T}_1^{\text{IID}}(v)$ by $200 \log^2(n)/n$, which is itself identical to $T_1^{\text{IID}}(v)$ by definition.

If $\bar{T}_1^{\text{IID}}(v)$ is not close to deterministic, the padding does nothing and the base case holds. If $T_1^{\text{IID}}(v)$ is close enough to deterministic that it gets padded, the divergence from $T_1^{\pi,t}(v)$ either decreases or increases to at most $D_{KL}(\text{Ber}(0) \| \text{Ber}(\epsilon))$, where $\epsilon = 200 \log^2(n)/n$ is the padding parameter. In this case

$$\begin{aligned} D_{KL}(T_1^{\pi,t}(v) \| \tilde{T}_1^{\text{IID}}(v) | G^t) &\leq D_{KL}\left(\text{Ber}(0) \| \text{Ber}\left(\frac{200 \log^2 n}{n}\right)\right) \\ &= -\log\left(1 - \frac{200 \log^2 n}{n}\right) \\ &\leq \frac{400 \log^2 n}{n}, \end{aligned}$$

which suffices. (See Fact 3 from the proof of Lemma 27 in Appendix E.)

For, the inductive step, we will use the triangle inequality of Lemma 27, pivoting on $T_{j+1}^{\chi,t}(v)$. Specifically, we invoke Lemma 27 with

$$\begin{aligned} p &= \mathbb{P}(T_{j+1}^{\pi,t}(v) \text{ succeeds}) \\ q &= \mathbb{P}(T_{j+1}^{\chi,t}(v) \text{ succeeds}) \\ r &= \mathbb{P}(\bar{T}_{j+1}^{\text{IID}}(v) \text{ succeeds}). \end{aligned}$$

and $\epsilon = 200c^j \log^2 n/n$. Since the ϵ -padding of r , denoted by $\tilde{r} = \text{PADDING}(r, \epsilon)$ is exactly $\mathbb{P}(\tilde{T}_{j+1}^{\pi,t}(v) \text{ succeeds})$, by Lemma 27 it suffices to prove

$$D_{KL}(T_{j+1}^{\pi,t}(v) \| T_{j+1}^{\chi,t}(v) | G^t) \leq \frac{200c^j \log^2 n}{n} \quad (43)$$

and

$$D_{KL}(T_{j+1}^{\chi,t}(v) \| \bar{T}_{j+1}^{\text{IID}}(v) | G^t) \leq \frac{200c^j \log^2 n}{n}. \quad (44)$$

Eq. (44) holds by Lemma 31.

Comparing $T_{j+1}^{\pi,t}(v)$ and $T_{j+1}^{\chi,t}(v)$ (establishing Eq. (43)): We will deconstruct T_{j+1} identically to what was done in the proof of Lemma 28, and we will use techniques for bounding the divergence similar to the ones used in the proof of Lemma 28 for bounding the total variation distance. Recall the 0,1 vector $W_{j+1}(v)$ describing the process of a $T_{j+1}(v)$ test: The first $c^j m/n$ coordinates denote the search phase; specifically we write a 1 if a neighbor was found and a 0 if not. The following coordinates denote the outcomes of the recursive tests, 1 for pass 0 for fail, in the order they were performed. There could be at most c^j recursive tests performed (if they were all T_1 's) so $W_{j+1}(v)$ has length $c^j m/n + c^j$ in total. However, often much fewer recursive tests are performed due to the early stopping rule, or simply because too few neighbors of v were found. In this case, tests not performed are represented by a 0 in $W_{j+1}(v)$.

Since $W_{j+1}(v)$ determines $T_{j+1}(v)$ we can simply bound $D_{KL}(W_{j+1}^{\pi,t}(v) \| W_{j+1}^{\chi,t}(v) | G^t)$ by the data processing inequality of Lemma 29.

The first $c^j m/n$ coordinates of W^π and W^χ are distributed identically and contribute nothing to the divergence. Consider the test corresponding to the i^{th} coordinate of the recursive phase of $W_{j+1}(v)$. Let the level of the test be $\ell_i \in [0, j]$ (with 0 representing no test) and let the vertex of the test be u_i . These are random variable determined by Π . Furthermore, let t_i be the position in the stream where the recursive T_{ℓ_i} test is called on u_i . Then we have

$$\begin{aligned}
D_{KL}(T_{j+1}^{\pi,t}(v) \| T_{j+1}^{\chi,t}(v) | G^t) &\leq D_{KL}(W_{j+1}^{\pi,t}(v) \| W_{j+1}^{\chi,t}(v) | G^t) \\
&= \sum_i D_{KL}(W_{j+1,i}^{\pi,t}(v) \| W_{j+1,i}^{\chi,t}(v) | G^t, W_{j+1,1}^{\pi,t}(v), \dots, W_{j+1,i-1}^{\pi,t}(v)) \\
&\leq \sum_i \mathbb{E}_{\ell_i, u_i} D_{KL}(W_{j+1,i}^{\pi,t}(v) \| W_{j+1,i}^{\chi,t}(v) | G^{t_i}, \ell_i, u_i) \\
&= \sum_i \mathbb{E}_{\ell_i, u_i} D_{KL}(T_{\ell_i}^{\pi, t_i}(u_i) \| \tilde{T}_{\ell_i}^{\text{IID}}(u_i) | G^{t_i}) \\
&\leq \sum_i \mathbb{E}_{\ell_i} \left(\frac{200C \cdot c^{\ell_i-1} \log^2 n}{n} \right),
\end{aligned}$$

by the inductive hypothesis. Here in the third line we condition on ℓ_i and u_i , thereby only increasing the divergence.

Note that the term in the sum is proportional to the number of edges used by the corresponding recursive test. Indeed, recall by [Lemma 2](#) that a T_{ℓ_i} test runs for at most $c^{\ell_i-1} \cdot \frac{m}{n} \cdot (1 + 2\delta)$ samples with probability one. It is crucial that this result holds with probability one, and therefore extends to not just the iid stream it was originally proven on, but any arbitrary stream of edges.

Therefore, the term in the sum above is at most $200C \log^2(n)/m$ times the number of edges used by the corresponding recursive test. So the entire sum is at most $200C \log^2(n)/m$ times the length of the recursive phase of the original T_{j+1} test. It was also derived in [Lemma 2](#) that the recursive phase itself takes at most $c^j \cdot \frac{m}{n} \cdot 2\delta$ edges. (Again, the result holds with probability one.) Therefore,

$$D_{KL}(T_{j+1}^{\pi,t}(v) \| T_{j+1}^{\chi,t}(v) | G^t) \leq \frac{200C \log^2(n)}{m} \cdot \frac{\delta c^j m}{n} \leq \frac{200 \cdot c^j \log^2 n}{n},$$

for small enough absolute constant δ . This shows [Eq. \(43\)](#) and concludes the proof. \square

9.5 The full algorithm

We proceed to derive an algorithm for approximating the matching size of a graph in a random permutation stream. The following well-known theorem will be useful for proving correctness:

Theorem 12 (Pinsker's Inequality). *For two distributions P and Q ,*

$$\|P - Q\|_{TV} \leq \sqrt{2 \log_2 e \cdot D_{KL}(P \| Q)}.$$

Recall the EDGE-LEVEL-TEST from [Algorithm 3](#) in [Section 3](#). We will run a similar edge test on the permutation stream, with one crucial difference: Our original algorithm from [Section 3](#) uses $\Theta(m)$ samples from the stream. Since our bound on the divergence between the iid and permutation variants of LEVEL- j -TEST scales with $\log^2 n$ times the sample complexity, it would grow past constant if we were to adapt our iid algorithm as is. Therefore, we will constrain the algorithm to only use a $\log^2 n$ fraction of the available stream. Specifically EDGE-LEVEL-TEST will only calculate the level of an edge up to $J - 2 \log_c(\log n)$, as opposed to J . We will call this edge test $E^{\text{IID}}(e)$.

Algorithm 16 Given an edge e , this algorithm returns a fractional matching-weight of e .

```

1: procedure  $E^{\text{IID}}(e = (u, v))$ 
2:    $w \leftarrow 1/n$ 
3:   for  $i = 1$  to  $J - 2 \log_c(\log n)$  do ▷ Recall that  $J = \lfloor \log_c n \rfloor - 1$ 
4:     if LEVEL- $i$ -TEST( $u$ ) and LEVEL- $i$ -TEST( $v$ ) then
5:        $w \leftarrow w + c^i/n$ 
6:     else
7:       return  $w$ 
8:   return  $w$ 

```

As in the previous section we define the edge level test $E(e)$ for the permutation stream as well, namely $E^{\pi,t}(e)$. We also define \tilde{E}^{IID} using the padded \tilde{T}_i^{IID} tests in place of LEVEL- i -TEST.

Although the truncated edge test E^{IID} is not as powerful as the untruncated version, it is a $\Theta(\log^2 n)$ -factor estimator for the matching number of the input graph, $\text{MM}(G)$.

Corollary 10. *For all c large enough, there exists $\delta > 0$ such that the following holds. For all $G = (V, E)$ and an edge $e \in E$, let $E^{\text{IID}}(e)$ denote the value returned by [Algorithm 16](#). Then,*

$$\sum_{e \in E} E^{\text{IID}}(e) = O(\text{MM}(G))$$

$$\sum_{e \in E} E^{\text{IID}}(e) = \Omega\left(\frac{\text{MM}(G)}{\log^2 n}\right).$$

Proof. Recall that [Theorem 5](#) guarantees that

$$\sum_{e \in E} M_e = \Theta(\text{MM}(G)),$$

where M_e is the output of the untruncated edge level test EDGE-LEVEL-TEST from [Algorithm 3](#). Note that, by stopping $2 \log_c(\log n)$ levels early, E^{IID} may misclassify edges by assigning them to levels up to $2 \log_c(\log n)$ levels lower, but never misclassifies them by putting them higher. Therefore, $E^{\text{IID}}(e)$ is no greater than M_e and at most $\Theta(\log^2 n)$ factor lower. \square

Finally, recall [Algorithm 2](#), our main algorithm for estimating the matching size.

Algorithm 17 [Algorithm 2](#) estimating the value of $\text{MM}(G)$.

```

1: procedure IID-PEELING( $G = (V, E)$ )
2:    $M' \leftarrow 0$ 
3:    $s \leftarrow 1$ 
4:   while samples lasts do
5:      $M' \leftarrow \text{SAMPLE}(G, s)$ 
6:      $s \leftarrow 2s$ 
7:   return  $M'$ 
8:
9: procedure SAMPLE( $G = (V, E), s$ )
10:   $M' \leftarrow 0$ 
11:  for  $k = 1$  to  $s$  do
12:     $e \leftarrow$  iid edge from the stream
13:     $M' \leftarrow M' + \text{EDGE-LEVEL-TEST}(e)$ 
14:  return  $m \cdot \frac{M'}{s}$ 

```

Note that the variable denoting the number of edges sampled in SAMPLE, originally t , has been changed to s here so as to not be confused with the variable used to denote our position in the stream.

We will now describe the permutation variant, which can be used to approximate the maximum matching size to an $O(\log^2 n)$ factor, in a random permutation stream with $O(\log^2 n)$ bits of space.

Definition 17. Let PERMUTATION-PEELING($G = (V, E)$) be a variant of IID-PEELING that uses the permutation stream in [Line 12](#) and the subroutine $E^{\pi,t}(e)$ in [Line 13](#). Furthermore, it should only continue until a $\Theta(1/\log^2 n)$ fraction of the stream is exhausted, as opposed to [Algorithm 2](#) which exhausts the entire $\Theta(m)$ sized stream in [Line 4](#).

We define further variants of [Algorithm 17](#) that will be useful in the proof of correctness of PERMUTATION-PEELING.

Definition 18. We define the following three variants of [Algorithm 17](#):

- TRUNCATED-IID-PEELING($G = (V, E)$) uses the iid stream in [Line 12](#) and E^{IID} in [Line 13](#).
- HYBRID-PEELING($G = (V, E)$) uses the permutation stream in [Line 12](#) and E^{IID} in [Line 13](#).

- PADDED-PEELING($G = (V, E)$) uses the permutation stream in [Line 12](#) and \tilde{E}^{IID} in [Line 13](#).

All three algorithms terminate after exhausting a $\Theta(1/\log^2 n)$ fraction of the stream, like PERMUTATION-PEELING.

Theorem 13. For sufficiently small $\delta > 0$ and large enough c the following holds. For any graph $G = (V, E)$, PERMUTATION-PEELING ([Algorithm 17](#)) with $3/4$ probability outputs a $O(\log^2 n)$ factor approximation of $\text{MM}(G)$ by using a single pass over a randomly permuted stream of edges.

Proof. Correctness of Truncated-IID-Peeling. We will first prove the same claim for TRUNCATED-IID-PEELING. Recall first the proof of [Theorem 6](#) from [Section 4](#). Let M_e denote the outcome of $\text{EDGE-LEVEL-TEST}(e)$ and $\mu = \mathbb{E}_{e \sim U(E)}[M_e]$. Recall that the proof hinges on showing that μ is within a constant factor of $\text{MM}(G)$. Also $M_e : e \sim U(E)$ has variance at most $2\mu/c$, so $\Theta(1/\mu c)$ independent edge tests suffice to be able to bound the deviation from the mean powerfully enough using Chebyshev's inequality. Furthermore, we know that in expectation, $\text{EDGE-LEVEL-TEST}(e) : e \sim U(E)$ takes samples proportional to the number $\mu \cdot m$ (see [Lemma 3](#)). Putting all this together we get that with constant probability the output of $\text{SAMPLE}(G, s)$ is within a constant factor of $\text{MM}(G)$ for large enough s , and such a call of SAMPLE fits into $O(m)$ edges from the stream.

We will have a very similar proof for the correctness of TRUNCATED-IID-PEELING. Indeed, consider the random variable $S = E^{\text{IID}}(e) : e \sim U(E)$. Let $\mathbb{E}S = \bar{\mu}$. In [Corollary 10](#) we have shown that $\bar{\mu}$ is a $\Theta(\log^2 n)$ -factor approximation of $\text{MM}(G)$. Furthermore, S is bounded by $\frac{2}{c \log^2 n}$, so its variance is at most $\frac{2\bar{\mu}}{c \log^2 n}$. By Chebyshev's inequality $s = \Theta(\frac{1}{c\bar{\mu} \log^2 n})$ edge tests suffice to get an accurate enough empirical mean. This many edge tests take $s \cdot \bar{\mu} \cdot m = \Theta(\frac{m}{c \log^2 n})$ samples, as desired. If the algorithm doesn't terminate within $\frac{mR}{c \log^2 n}$ we consider it to have failed. This happens with probability less than $1/10$ for some large absolute constant R .

Correctness of Hybrid-Peeling. The only difference between TRUNCATED-IID-PEELING and HYBRID-PEELING is that the latter we use the permutation stream for sampling edges to run E^{IID} on. This guarantees no repetitions which only improves our variance bound and does not hurt the previous proof.

Comparing Hybrid-Peeling and Padded-Peeling. Note that these two algorithms differ only in the type of vertex level tests they use: Specifically, HYBRID-PEELING uses T^{IID} while PADDED-PEELING uses \tilde{T}^{IID} . By [Lemma 28](#) for all $v \in V$ and $j \in [0, J]$

$$\left\| T_{j+1}^{\text{IID}}(v) - \tilde{T}_{j+1}^{\text{IID}}(v) \right\|_{\text{TV}} \leq \frac{400 \cdot c^j \log^2 n}{n}.$$

Note that this is proportional (with multiplicative factor $\frac{200 \log^2 n}{m}$) to the sample complexity of the corresponding test. The output of the main peeling algorithm depends only on the outputs of vertex level tests called directly from edge level tests, which are disjoint (in the samples they use). Furthermore, the entire algorithm takes at most $\frac{mR}{c \log^2 n}$ samples, so the total variation distance between the outputs of HYBRID-PEELING and PADDED-PEELING is at most

$$\frac{200 \log^2 n}{m} \cdot \frac{mR}{c \log^2 n} = \frac{200R}{c}.$$

For more details on this proof technique see the proofs of [Lemmas 28](#) and [30](#).

Comparing Padded-Peeling and Permutation-Peeling. Note that these two algorithms again differ only in the type of vertex level test they use: Specifically, PADDED-PEELING uses \tilde{T}^{IID} while PERMUTATION-PEELING uses $T^{\pi, t}$. By [Lemma 30](#), with high probability, for all $v \in V$, $j \leq J$, $t \leq m/2 - 2c^j \cdot \frac{m}{n}$,

$$D_{KL} \left(T_{j+1}^{\pi, t}(v) \left\| \tilde{T}_{j+1}^{\text{IID}}(v) \right| G^t \right) \leq \frac{200C \cdot c^j \log^2 n}{n}, \quad (45)$$

where C is the constant from [Lemma 27](#). Note that this is proportional (with multiplicative factor $\frac{100C \log^2 n}{m}$) to the sample complexity of the corresponding test. Again we note that the output of the main peeling algorithm is a function of the outputs of the vertex level tests directly called from edge level tests, which are disjoint (in the samples they use). Furthermore, since the entire algorithm takes at most $\frac{mR}{c \log^2 n}$ samples, the divergence between the outputs of PADDED-PEELING and PERMUTATION-PEELING is at most

$$\frac{100 \log^2 n}{m} \cdot \frac{mR}{c \log^2 n} = \frac{100R}{c}.$$

This translates to a total variation distance of at most $\sqrt{\frac{200R \log_2 e}{c}}$ by Pinsker’s inequality. Again, for more details on this proof technique see the proofs of [Lemmas 28](#) and [30](#).

In conclusion, PERMUTATION-PEELING returns a $\log^2 n$ -factor approximation of $\text{MM}(G)$ with probability at least $4/5 - \frac{200R}{c} - \sqrt{\frac{200R \log_2 e}{c}} \geq 3/4$ for large enough c . \square

From here the proof of the main theorem follows.

Proof of [Theorem 4](#). The proof follows from [Theorem 13](#) and the fact that [Algorithm 17](#) has recursion depth of $O(\log n)$, where each procedure in the recursion maintains $O(1)$ variables, hence requiring $O(\log n)$ bits of space. Therefore, the total memory is $O(\log^2 n)$. \square

Acknowledgments

Michael Kapralov is supported in part by ERC Starting Grant 759471. Slobodan Mitrović was supported by the Swiss NSF grant P2ELP2_181772 and MIT-IBM Watson AI Lab. Jakab Tardos is supported by ERC Starting Grant 759471.

References

- [AB19] Sepehr Assadi and Aaron Bernstein. Towards a unified theory of sparsification for matching problems. In Fineman and Mitzenmacher [[FM18](#)], pages 11:1–11:20.
- [ABB⁺19] Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab S. Mirrokni, and Cliff Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1616–1635. SIAM, 2019.
- [AG11a] Kook Jin Ahn and Sudipto Guha. Laminar families and metric embeddings: Non-bipartite maximum matching problem in the semi-streaming model. *arXiv preprint arXiv:1104.4058*, 2011.
- [AG11b] Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. In *International Colloquium on Automata, Languages, and Programming*, pages 526–538. Springer, 2011.
- [AK17] Sepehr Assadi and Sanjeev Khanna. Randomized composable coresets for matching and vertex cover. In Christian Scheideler and Mohammad Taghi Hajiaghayi, editors, *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, pages 3–12. ACM, 2017.
- [AKL17] Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In Klein [[Kle17](#)], pages 1723–1742.
- [AKLY16] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavltssev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In Krauthgamer [[Kra16](#)], pages 1345–1364.
- [ARVX12] 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*, pages 1132–1139. Society for Industrial and Applied Mathematics, 2012.
- [BGM⁺19] Marc Bury, Elena Grigorescu, Andrew McGregor, Morteza Monemizadeh, Chris Schwiegelshohn, Sofya Vorotnikova, and Samson Zhou. Structural results on matching estimation with applications to streaming. *Algorithmica*, 81(1):367–392, 2019.
- [BGY18] Paul Beame, Shayan Oveis Gharan, and Xin Yang. Time-space tradeoffs for learning finite functions from random evaluations, with applications to polynomials. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018.*, volume 75 of *Proceedings of Machine Learning Research*, pages 843–856. PMLR, 2018.

- [BS15] Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 263–274. Springer, 2015.
- [CCE⁺16] Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In Krauthgamer [Kra16], pages 1326–1344.
- [CH12] John Cullinan and Farshid Hajir. Primes of prescribed congruence class in short intervals. *Integers*, 12:A56, 2012.
- [CJMM17] Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and S. Muthukrishnan. The sparse awakens: Streaming algorithms for matching size estimation in sparse graphs. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 29:1–29:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
- [EHL⁺15] Hossein Esfandiari, Mohammad Taghi Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1217–1233. SIAM, 2015.
- [EHM16] Hossein Esfandiari, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Finding large matchings in semi-streaming. In Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo A. Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain.*, pages 608–614. IEEE Computer Society, 2016.
- [EKS09] Sebastian Eggert, Lasse Kliemann, and Anand Srivastav. Bipartite graph matchings in the semi-streaming model. In *European Symposium on Algorithms*, pages 492–503. Springer, 2009.
- [ELMS11] Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM Journal on Discrete Mathematics*, 25(3):1251–1265, 2011.
- [EMR14] Guy Even, Moti Medina, and Dana Ron. Deterministic stateless centralized local algorithms for bounded degree graphs. In *European Symposium on Algorithms*, pages 394–405. Springer, 2014.
- [FKM⁺05] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005.
- [FM18] Jeremy T. Fineman and Michael Mitzenmacher, editors. *2nd Symposium on Simplicity in Algorithms, SOSA@SODA 2019, January 8-9, 2019 - San Diego, CA, USA*, volume 69 of *OASICS*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.
- [Gha16] Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 270–277. Society for Industrial and Applied Mathematics, 2016.
- [GKK12] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485. SIAM, 2012.
- [GKMS18] Buddhima Gamblath, Sagar Kale, Slobodan Mitrović, and Ola Svensson. Weighted matchings via unweighted augmentations. *arXiv preprint arXiv:1811.02760*, 2018.

- [GRT18] Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 990–1002. ACM, 2018.
- [GU19] Mohsen Ghaffari and Jara Uitto. Sparsifying distributed algorithms with ramifications in massively parallel computation and centralized local computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1636–1653. SIAM, 2019.
- [GW19] Mohsen Ghaffari and David Wajc. Simplified and space-optimal semi-streaming $(2+\epsilon)$ -approximate matching. In Fineman and Mitzenmacher [FM18], pages 13:1–13:8.
- [Kap13] Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1679–1697. SIAM, 2013.
- [KKS14] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 734–751. SIAM, 2014.
- [Kle17] Philip N. Klein, editor. *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*. SIAM, 2017.
- [KMM12] Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 231–242. Springer, 2012.
- [KMW16] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *Journal of the ACM (JACM)*, 63(2):17, 2016.
- [Kon18] Christian Konrad. A simple augmentation method for matchings with applications to streaming algorithms. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
- [Kra16] Robert Krauthgamer, editor. *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. SIAM, 2016.
- [KRT17] Gillat Kol, Ran Raz, and Avishay Tal. Time-space hardness of learning sparse parities. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1067–1080. ACM, 2017.
- [LPS88] Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.
- [LRY17] Reut Levi, Ronitt Rubinfeld, and Anak Yodpinyanee. Local computation algorithms for graphs of non-constant degrees. *Algorithmica*, 77(4):971–994, 2017.
- [McG05] Andrew McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 170–181. Springer, 2005.
- [MMPS17] Morteza Monemizadeh, S. Muthukrishnan, Pan Peng, and Christian Sohler. Testable bounded degree graph properties are random order streamable. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 131:1–131:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

- [MRVX12] Yishay Mansour, Aviad Rubinfeld, Shai Vardi, and Ning Xie. Converting online algorithms to local computation algorithms. In *International Colloquium on Automata, Languages, and Programming*, pages 653–664. Springer, 2012.
- [MV13] Yishay Mansour and Shai Vardi. A local computation approximation scheme to maximum matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 260–273. Springer, 2013.
- [MV16] Andrew McGregor and Sofya Vorotnikova. Planar matching in streams revisited. In Klaus Jansen, Claire Mathieu, José D. P. Rolim, and Chris Umans, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, volume 60 of *LIPIcs*, pages 17:1–17:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.
- [MV18] Andrew McGregor and Sofya Vorotnikova. A simple, space-efficient, streaming algorithm for matchings in low arboricity graphs. In Raimund Seidel, editor, *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, volume 61 of *OASICS*, pages 14:1–14:4. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.
- [Nis90] Noam Nisan. Pseudorandom generators for space-bounded computation. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 204–212, 1990.
- [NO08] Huy N Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 327–336. IEEE, 2008.
- [ORRR12] Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1123–1131. Society for Industrial and Applied Mathematics, 2012.
- [Pem01] Sriram V Pemmaraju. Equitable coloring extends chernoff-hoeffding bounds. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pages 285–296. Springer, 2001.
- [PR07] Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1-3):183–196, 2007.
- [PS17] Ami Paz and Gregory Schwartzman. A $(2 + \epsilon)$ -approximation for maximum weight matching in the semi-streaming model. In Klein [Kle17], pages 2153–2161.
- [PS18] Pan Peng and Christian Sohler. Estimating graph parameters from random order streams. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2449–2466. SIAM, 2018.
- [Raz16] Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 266–275. IEEE Computer Society, 2016.
- [Raz17] Ran Raz. A time-space lower bound for a large class of learning problems. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 732–742. IEEE Computer Society, 2017.
- [RTVX11] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. *arXiv preprint arXiv:1104.1377*, 2011.
- [YYI09] Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 225–234. ACM, 2009.
- [Zel12] Mariano Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62(1-2):1–20, 2012.

A Proof of Lemma 5

In this section we prove the following result, stated in [Section 5.2](#).

Lemma 5 (Concentration on $\widehat{M}(v)$). *For any vertex v , any $j > 0$, and constants c and x such that $c \geq 20$ and $x \geq 100c \log c$, we have:*

$$\mathbb{P} \left[\widehat{M}(v) \geq x \wedge \widehat{L}(v) = j + 1 \right] \leq \frac{10c^2}{x^2} \cdot \mathbb{E} \left[\widehat{M}(v) \cdot \mathbb{1} \left[\widehat{L}(v) = j \right] \right]. \quad (18)$$

Where $\widehat{L}(v)$ and $\widehat{M}(v)$ are defined in [Eq. \(10\)](#) and [Eq. \(16\)](#), respectively.

Proof. We begin by rewriting the LHS and the RHS of [Eq. \(18\)](#).

Rewriting the LHS of [Eq. \(18\)](#) Observe that

$$\mathbb{P} \left[\widehat{M}(v) \geq x \mid \widehat{L}(v) = j + 1 \right] \stackrel{\text{Observation 1 (b)}}{=} \mathbb{P} \left[\widehat{M}_{j+1}(v) \geq x \mid \widehat{L}(v) = j + 1 \right].$$

First observe that $\mathbb{P} \left[\widehat{M}(v) \geq x \mid \widehat{L}(v) = j + 1 \right] = \mathbb{P} \left[\widehat{M}_{j+1}(v) \geq x \mid \widehat{L}(v) = j + 1 \right]$, since $\widehat{M}_{j+1}(v) = \widehat{M}(v)$ conditioned on $\widehat{L}(v) = j + 1$ by definition. We thus have

$$\begin{aligned} \mathbb{P} \left[\widehat{M}(v) \geq x \wedge \widehat{L}(v) = j + 1 \right] &= \mathbb{P} \left[\widehat{M}_{j+1}(v) \geq x \wedge \widehat{L}(v) = j + 1 \right] \\ &\stackrel{\text{Observation 1 (a)}}{=} \mathbb{P} \left[\widehat{M}_{j+1}(v) \geq x \right] \mathbb{P} \left[\widehat{L}(v) = j + 1 \right] \\ &= \mathbb{P} \left[\widehat{M}_{j+1}(v) \geq x \right] \mathbb{P} \left[v \in \widehat{V}_j \right] \mathbb{P} \left[v \in \widehat{V}_{j+1} \mid v \in \widehat{V}_j \right] \mathbb{P} \left[v \notin \widehat{V}_{j+2} \mid v \in \widehat{V}_{j+1} \right] \\ &= \mathbb{P} \left[\widehat{M}_{j+1}(v) \geq x \right] \mathbb{P} \left[v \in \widehat{V}_j \right] \mathbb{P} \left[S_j(v) < \delta \right] \mathbb{P} \left[S_{j+1}(v) \geq \delta \right], \end{aligned} \quad (46)$$

where $S_j(v)$ is as defined in [Eq. \(13\)](#). (When $j = J$, $\mathbb{P} \left[v \notin \widehat{V}_{j+2} \right]$ is simply 1.)

Rewriting the RHS of [Eq. \(18\)](#) We have

$$\mathbb{E} \left[\widehat{M}(v) \cdot \mathbb{1} \left[\widehat{L}(v) = j \right] \right] = \mathbb{P} \left[v \in \widehat{V}_j \right] \mathbb{P} \left[\widehat{L}(v) = j \mid v \in \widehat{V}_j \right] \mathbb{E} \left[\widehat{M}_j(v) \right].$$

By definition $\mathbb{P} \left[\widehat{L}(v) = j \mid v \in \widehat{V}_j \right] = \mathbb{P} \left[S_j \geq \delta \right]$, and hence

$$\mathbb{E} \left[\widehat{M}(v) \cdot \mathbb{1} \left[\widehat{L}(v) = j \right] \right] = \mathbb{P} \left[v \in \widehat{V}_j \right] \mathbb{P} \left[S_j(v) \geq \delta \right] \mathbb{E} \left[\widehat{M}_j(v) \right]. \quad (47)$$

The proof strategy In the rest of the proof, to establish [Eq. \(18\)](#) we upper-bound the ratio of the LHS of [Eq. \(46\)](#) and the RHS of [Eq. \(47\)](#) by $10c^2/x^2$. At a high level, the RHS of [Eq. \(46\)](#) is small when $x \gg \delta$. This is the case since the random variables $\widehat{M}_{j+1}(v)$ and $S_j(v)$ (see [Eq. \(17\)](#) and [Eq. \(13\)](#) respectively) have similar expectations and they both concentrate well around their expectations. Hence, it is unlikely that at the same time $\widehat{M}_{j+1}(v)$ is large and $S_j(v)$ is small.

To implement this intuition, we consider two cases with respect to $\mathbb{E} \left[\widehat{M}_{j+1}(v) \right]$. First, when $\mathbb{E} \left[\widehat{M}_{j+1}(v) \right]$ is relatively large, i.e., at least $x/2$, we show that $\mathbb{P} \left[S_j(v) < \delta \right]$ is small. On the other hand, for the terms appearing on the RHS of [Eq. \(47\)](#) we have: large $\mathbb{E} \left[\widehat{M}_{j+1}(v) \right]$ implies large $\mathbb{E} \left[\widehat{M}_j(v) \right]$, and small $\mathbb{P} \left[S_j(v) < \delta \right]$ implies that $\mathbb{P} \left[S_j(v) \geq \delta \right] \geq 1/2$.

Second, when $\mathbb{E} \left[\widehat{M}_{j+1}(v) \right] < x/2$, we show that $\mathbb{P} \left[\widehat{M}_{j+1}(v) \geq x \right]$ is very small.

We complete the proof by balancing the two cases.

Case 1: $\mathbb{E} [\widehat{M}_{j+1}(v)] \geq x/2$. In this case we have

$$\mathbb{E} [S_j(v)] \stackrel{\text{Observation 2}}{\geq} \frac{\mathbb{E} [S_{j+1}(v)]}{c+1} \stackrel{\text{Observation 1 (c)}}{=} \frac{\mathbb{E} [\widehat{M}_{j+1}(v)]}{c+1} \geq \frac{x}{2(c+1)}. \quad (48)$$

This further implies

$$\mathbb{E} [\widehat{M}_j(v)] \stackrel{\text{Observation 1 (c)}}{=} \mathbb{E} [S_j(v)] \stackrel{\text{Eq. (48)}}{\geq} \frac{x}{2(c+1)}. \quad (49)$$

Let us define a random Z_k for iid edge e_k as follows

$$Z_k = \mathbb{1} [e_k \text{ equals } \{w, v\}] \sum_{i=0}^{\min\{L(w), j\}} c^{i-j}.$$

Notice that $S_j(v) = \sum_{k=1}^{c^j m/n} Z_k$ and $Z_k \in [0, 2]$, therefore by applying Chernoff bound [Theorem 7 \(c\)](#) to $S_j(v)$:

$$\begin{aligned} \mathbb{P} [S_j(v) < \delta] &\leq \mathbb{P} [S_j(v) \leq (1 - 1/2)\mathbb{E} [S_j(v)]] \\ &\leq \exp \left(-\frac{(1/2)^2 \mathbb{E} [S_j(v)]}{2 \cdot 2} \right) \\ &\stackrel{\text{Eq. (48)}}{\leq} \exp \left(-\frac{(1/2)^2 \frac{x}{2(c+1)}}{2 \cdot 2} \right) \\ &\leq \exp \left(-\frac{x}{32(c+1)} \right) \\ &\leq \exp \left(-\frac{x}{40c} \right), \end{aligned} \quad (50)$$

where the first inequality follows as $\delta \leq 1/2$ and $\mathbb{E} [S_j(v)] \geq 1$ from [Eq. \(48\)](#) and the definition of x . The last inequality of [Eq. \(50\)](#) follows since $c \geq 20$ by the assumption of the lemma. Therefore, substituting [Eq. \(50\)](#) into [Eq. \(46\)](#), we get

$$\mathbb{P} [\widehat{M}(v) \geq x \wedge \widehat{L}(v) = j+1] \leq \mathbb{P} [v \in \widehat{V}_j] \exp \left(-\frac{x}{40c} \right). \quad (51)$$

Furthermore, from [Eq. \(50\)](#) and for $x \geq 100c \log c$ we have $\mathbb{P} [S_j \geq \delta] \geq 1/2$. Substituting this bound and [Eq. \(49\)](#) into [Eq. \(47\)](#) leads to

$$\mathbb{E} [\widehat{M}(v) \cdot \mathbb{1} [\widehat{L}(v) = j]] \geq \frac{1}{2} \mathbb{P} [v \in \widehat{V}_j] \cdot \frac{x}{2(c+1)}.$$

Combining the last inequality with [Eq. \(51\)](#) leads to

$$\frac{\mathbb{P} [\widehat{M}(v) \geq x \wedge \widehat{L}(v) = j+1]}{\mathbb{E} [\widehat{M}(v) \cdot \mathbb{1} [\widehat{L}(v) = j]]} \leq \frac{4(c+1) \cdot \exp(-\frac{x}{40c})}{x}. \quad (52)$$

Case 2: $\mathbb{E} [\widehat{M}_{j+1}(v)] < x/2$. Let $\mathbb{E} [\widehat{M}_{j+1}(v)] = tx$ for some $t < 1/2$. To apply Chernoff bound, similar to previous case we define Z_w for any vertex $w \in N(v)$ as follows

$$Z_w \stackrel{\text{def}}{=} \sum_{i=0}^{\min\{\widehat{L}(w), j\}} c^i/n.$$

Therefore we have $\widehat{M}_j(v) = \sum_{w \in N(v)} Z_w$. Observe that $Z_w \in [0, 2]$. Since $t < 1/2$, by applying

Chernoff bound [Theorem 7 \(b\)](#) we get

$$\begin{aligned}\mathbb{P}\left[\widehat{M}_{j+1}(v) \geq x\right] &= \mathbb{P}\left[\widehat{M}_{j+1}(v) \geq (1 + (1/t - 1)) \cdot \mathbb{E}\left[\widehat{M}_{j+1}(v)\right]\right] \\ &\leq \exp\left(-\frac{1/t \cdot \log 1/t \cdot \mathbb{E}\left[\widehat{M}_{j+1}(v)\right]}{3 \cdot 2}\right) \\ &\leq \exp\left(-\frac{x \log 1/t}{6}\right).\end{aligned}$$

Substituting this bound into [Eq. \(46\)](#) we obtain

$$\mathbb{P}\left[\widehat{M}(v) \geq x \wedge \widehat{L}(v) = j+1\right] \leq \mathbb{P}\left[v \in \widehat{V}_j\right] \mathbb{P}[S_{j+1} \geq \delta] \exp\left(-\frac{x \log 1/t}{6}\right). \quad (53)$$

[Eq. \(47\)](#) and [Eq. \(53\)](#) imply

$$\begin{aligned}\frac{\mathbb{P}\left[\widehat{M}(v) \geq x \wedge \widehat{L}(v) = j+1\right]}{\mathbb{E}\left[\widehat{M}(v) \cdot \mathbb{1}\left[\widehat{L}(v) = j\right]\right]} &\leq \frac{2(c+1) \cdot \exp\left(-\frac{x \log 1/t}{6}\right)}{tx} \cdot \frac{\mathbb{P}[S_{j+1}(v) \geq \delta]}{\mathbb{P}[S_j(v) \geq \delta]} \\ &= \frac{2(c+1)}{x} \cdot t^{x/6-1} \cdot \frac{\mathbb{P}[S_{j+1}(v) \geq \delta]}{\mathbb{P}[S_j(v) \geq \delta]} \\ &\leq \frac{2(c+1)}{x} \cdot 2^{-x/6+1} \cdot \frac{\mathbb{P}[S_{j+1}(v) \geq \delta]}{\mathbb{P}[S_j(v) \geq \delta]}\end{aligned}$$

Now we upper bound $\frac{\mathbb{P}[S_{j+1}(v) \geq \delta]}{\mathbb{P}[S_j(v) \geq \delta]}$. Consider the definition of $S_j(v)$ from [Eq. \(13\)](#)

$$S_j(v) \stackrel{\text{def}}{=} \sum_{\substack{k=1 \\ e_k \sim U_E}}^{c^j m/n} \mathbb{1}[e_k \text{ equals } \{w, v\}] \sum_{i=0}^{\min(L(w), j)} c^{i-j}$$

Let us split this definition into two parts: one corresponding to the last term of the second sum and one corresponding to all other terms:

$$\begin{aligned}S_j(v) &= A_j(v) + B_j(v) \\ A_j(v) &= \sum_{\substack{k=1 \\ e_k \sim U_E}}^{c^j m/n} \mathbb{1}[e_k \text{ equals } \{w, v\}] \sum_{i=0}^{\min(L(w), j-1)} c^{i-j} \\ B_j(v) &= \sum_{\substack{k=1 \\ e_k \sim U_E}}^{c^j m/n} \mathbb{1}[e_k \text{ equals } \{w, v\}] \mathbb{1}\left[w \in \widehat{V}_j\right]\end{aligned}$$

Note that $\mathbb{P}[S_j(v) \geq \delta] = \mathbb{P}[A_j(v) \geq \delta \vee B_j \geq 1] \leq \mathbb{P}[A_j(v) \geq \delta] + \mathbb{P}[B_j(v) \geq 1]$. We must bound

$$\frac{\mathbb{P}[A_{j+1} \geq \delta] + \mathbb{P}[B_{j+1}(v) \geq 1]}{\mathbb{P}[S_j(v) \geq \delta]}.$$

Notice that $A_{j+1}(v)$ is the average of c independently sampled copies of $S_j(v)$, say $S_j^{(i)}(v)$. In order for $A_{j+1}(v)$ to be greater than δ at least one of the $S_j^{(i)}(v)$'s must be greater than δ , therefore by union bound $\mathbb{P}[A_{j+1} \geq \delta] \leq c\mathbb{P}[S_j(v) \geq \delta]$. Notice now that $B_{j+1}(v)$ is *at most* the sum of c independent copies of $B_j(v)$, say $B_j^{(i)}(v)$. Since $B_j(v)$ is integral, in order for $B_{j+1}(v)$ to be greater than 1 at least one of the $B_j^{(i)}(v)$'s need to be greater than 1, therefore by union bound $\mathbb{P}[B_{j+1}(v) \geq 1] \leq c\mathbb{P}[B_j(v) \geq 1] \leq c\mathbb{P}[S_j(v) \geq \delta]$. So in conclusion

$$\frac{\mathbb{P}[S_{j+1}(v) \geq \delta]}{\mathbb{P}[S_j(v) \geq \delta]} \leq 2c.$$

This finalizes the bound of this case as well

$$\frac{\mathbb{P} \left[\widehat{M}(v) \geq x \wedge \widehat{L}(v) = j+1 \right]}{\mathbb{E} \left[\widehat{M}(v) \mathbb{1}(\widehat{L}(v) = j) \right]} \leq \frac{2(c+1)}{x} \cdot 2^{-x/6+1} \cdot 2c. \quad (54)$$

Finalizing Combining the two cases, from Eq. (52) and Eq. (54) we conclude

$$\frac{\mathbb{P} \left[\widehat{M}(v) \geq x \wedge \widehat{L}(v) = j+1 \right]}{\mathbb{E} \left[\widehat{M}(v) \mathbb{1}(\widehat{L}(v) = j) \right]} \leq \max \left(\frac{4c(c+1)}{x} \cdot 2^{-x/6+1}, \frac{4(c+1) \cdot \exp(-\frac{x}{40c})}{x} \right)$$

The RHS of the inequality above is upper-bounded by $10c^2/x^2$ for $c \geq 20$ and $x \geq 100c \log c$. \square

B Oversampling Lemma

In this section we formally proof the following lemma.

Lemma 1 (Oversampling lemma). *For sufficiently small $\delta > 0$ and large enough c the following holds. Let $X = \sum_{k=1}^K Y_k$ be a sum of independent random variables Y_k taking values in $[0, 1]$, and $\overline{X} \stackrel{\text{def}}{=} \frac{1}{c} \sum_{i=1}^c X_i$ where X_i are iid copies of X . If $\mathbb{E}[X] \leq \delta/3$ and $\mathbb{P}[X \geq \delta] = p$, then $\mathbb{P}[\overline{X} \geq \delta] \leq p/2$.*

Proof. Let $Z = \sum_{i=1}^c X_i$. Notice that Z is a sum of independent random variables each in the range $[0, 1]$. Also, $\mathbb{P}[\overline{X} \geq \delta] = \mathbb{P}[Z \geq c\delta]$. From the definition of X_i and the linearity of expectation, we have $\mathbb{E}[Z] \leq c\delta/3$. This, in compination with Chernoff bound (Theorem 7(b)), further implies

$$\mathbb{P}[\overline{X} \geq \delta] = \mathbb{P}[Z \geq c\delta] \leq \exp \left(-\frac{2 \cdot c\delta/3}{3} \right) \leq \exp \left(-\frac{c\delta}{9} \right). \quad (55)$$

We now consider two cases depending on the value of p .

Case 1: $p \geq 2 \exp(-\frac{c\delta}{9})$. The proof follows directly from Eq. (55).

Case 2: $p < 2 \exp(-\frac{c\delta}{9})$. In this case we consider the following three events which we call *bad*.

- Event \mathcal{E}_1 : At least two of X_i 's have value at least δ .
- Event \mathcal{E}_2 : At least one X_i has value more than t , for a threshold $t := \delta c/30 \gg \delta$.
- Event \mathcal{E}_3 : At least one X_i has value more than δ and less than $0.1 \cdot c$ of the X_i 's have value below $2\delta/3$.

If none of the bad events happen, then $\overline{X} \leq \delta$. To see that, observe that $\bar{\mathcal{E}}_1$ and $\bar{\mathcal{E}}_2$ imply that at most one X_i has value more than δ , and that the same X_i has value at most t . Note that $\bar{\mathcal{E}}_3$ is the event that either none of the X_i 's has value more than δ or more than $0.1c$ of the X_i 's have value below $2\delta/3$. In the former case, $\overline{X} \leq \delta$ is clearly satisfied. Consider now the latter case intersected with $\bar{\mathcal{E}}_1$ and $\bar{\mathcal{E}}_2$; denote the X_i larger than δ by X_{large} . At least $0.1 \cdot c$ values of X_i are less than $2\delta/3$ and the rest, excluding X_{large} , are less than δ . Therefore, the average of X_{large} and the elements having value less than $2\delta/3$ is at most $\frac{0.1 \cdot c \cdot 2\delta/3 + t}{0.1 \cdot c} = 2\delta/3 + 10t/c$. This is less than δ as long as $t \leq \delta c/30$. All other elements are below δ as well.

In the rest of the proof we upper-bound the probability that each of the bad events occurs. Then, by taking union bound we will upper-bound $\mathbb{P}[\overline{X} \geq \delta]$.

Upper-bound on $\mathbb{P}[\mathcal{E}_1]$. By union bound we have

$$\mathbb{P}[\mathcal{E}_1] = \mathbb{P}[\exists i_1 \neq i_2 : X_{i_1} \geq \delta \wedge X_{i_2} \geq \delta] \leq \binom{c}{2} p^2 \leq c^2 p \exp \left(-\frac{c\delta}{9} \right) \quad (56)$$

Upper-bound on $\mathbb{P}[\mathcal{E}_2]$. Again by union bound we derive

$$\begin{aligned}\mathbb{P}[\mathcal{E}_2] &= \mathbb{P}[\exists i : X_i \geq t] \\ &\leq c \cdot \mathbb{P}[X \geq \delta] \cdot \mathbb{P}[X \geq t | X \geq \delta] \\ &= cp \cdot \mathbb{P}[X \geq t | X \geq \delta].\end{aligned}\tag{57}$$

To upper-bound $\mathbb{P}[X \geq t | X \geq \delta]$, consider the random variable L defined as the lowest integer such that the partial sum $\sum_{k=1}^L Y_k$ is already at least δ . Then

$$\begin{aligned}\mathbb{P}[X \geq t | X \geq \delta] &= \sum_{l=1}^K \mathbb{P}[X \geq t | L = l] \mathbb{P}[L = l | X \geq \delta] \\ &\leq \max_l \mathbb{P}[X \geq t | L = l] \\ &= \max_l \mathbb{P}\left[\sum_{k=1}^{l-1} Y_k + Y_l + \sum_{k=l+1}^K Y_k \geq t | L = l\right].\end{aligned}\tag{58}$$

Recall that each $Y_k \in [0, 1]$. Also, for $L = l$, by the definition we have $\sum_{k=1}^{l-1} Y_k < \delta < 1$. Hence,

$$\sum_{k=1}^{l-1} Y_k + Y_l \leq 2.\tag{59}$$

This together with [Eq. \(58\)](#) implies

$$\begin{aligned}\mathbb{P}[X \geq t | X \geq \delta] &\stackrel{\text{from Eq. (58)}}{\leq} \max_l \mathbb{P}\left[\sum_{k=l+1}^K Y_k \geq t - \sum_{k=1}^{l-1} Y_k - Y_l | L = l\right] \\ &\stackrel{\text{from Eq. (59)}}{\leq} \max_l \mathbb{P}\left[\sum_{k=l+1}^K Y_k \geq t - 2 | L = l\right] \\ &\leq \mathbb{P}[X \geq t - 2].\end{aligned}\tag{60}$$

From the assumption given in the statement of the lemma, it holds that $\mathbb{E}[X] \leq \delta/3 < 1$ (we may constrain δ to be less than 3). By Chernoff bound ([Theorem 7\(b\)](#)) and taking into account that X is a sum of random variables in $[0, 1]$, we obtain

$$\mathbb{P}[X \geq t - 2] \stackrel{\text{from } \mathbb{E}[X] < 1}{\leq} \mathbb{P}[X \geq \mathbb{E}[X] + t - 3] \leq \exp\left(-\frac{t-3}{3}\right).$$

From the last chain of inequalities and [Eq. \(57\)](#) we derive

$$\mathbb{P}[\mathcal{E}_2] \leq cp \cdot \exp\left(-\frac{t-3}{3}\right).\tag{61}$$

Upper-bound on $\mathbb{P}[\mathcal{E}_3]$. Consider \mathcal{E}_3 as the union of the subevents $\mathcal{E}_3(i^*)$ when X_{i^*} is specifically greater than δ and less than $0.1 \cdot c$ of the rest of the X_i 's are below $2\delta/3$.

$$\mathbb{P}[\mathcal{E}_3] \leq \sum_{i^*=1}^c \mathbb{P}[\mathcal{E}_3(i^*)] = c\mathbb{P}[\mathcal{E}_3(1)] = cp\mathbb{P}[|\{i > 1 : X_i \leq 2\delta/3\}| < 0.1 \cdot c].\tag{62}$$

Note that by Markov's inequality we have $\mathbb{P}[X_i \leq 2\delta/3] \geq 1/2$ (since $\mathbb{P}[X_i \geq 2\delta/3] \leq 1/2$). Therefore,

$$\mathbb{E}[|\{i > 1 : X_i \leq 2\delta/3\}|] \geq (c-1)/2.$$

Hence, by Chernoff bound ([Theorem 7\(c\)](#)) we derive

$$\mathbb{P}[|\{i > 1 : X_i \leq 2\delta/3\}| \leq 0.1 \cdot c] \leq \exp\left(\frac{(3/4)^2(c-1)/2}{2}\right) \leq \exp\left(-\frac{c}{8}\right).$$

assuming that $c \geq 10$. This bound together with [Eq. \(62\)](#) implies

$$\mathbb{P}[\mathcal{E}_3] \leq cp \exp\left(-\frac{c}{8}\right).\tag{63}$$

Combining all the bounds. From Eq. (56), Eq. (61) and Eq. (63) we conclude

$$\begin{aligned}\mathbb{P}[\bar{X} \geq \delta] &\leq \mathbb{P}[\mathcal{E}_1] + \mathbb{P}[\mathcal{E}_2] + \mathbb{P}[\mathcal{E}_3] \\ &\leq c^2 p \exp\left(-\frac{c\delta}{9}\right) + cp \exp\left(-\frac{t-3}{3}\right) + cp \exp\left(-\frac{c}{8}\right) \\ &\leq p/2,\end{aligned}$$

when δ and c are set appropriately. Indeed recalling that $t = \frac{c\delta}{30}$ and set $c \geq 2000 \log(1/\delta)/\delta$ to achieve this goal. Notice that these bounds are not tight. \square

C Proofs omitted from Section 7

Proof of Lemma 17: Recall the definitions of I_e and T_e from the proof of Lemma 14: Let I_e be the indicator variable of e being explored when Algorithm 10 is called from e_0 ; let T_e be the size of the exploration tree from e in H_i . Let $T = T_{e_0}$, $t(\lambda) = t_{e_0}(\lambda)$. Let $v(\lambda) = \mathbb{E}(T^2 | r(e_0) = \lambda)$; we will derive a recursive formula for $v(\lambda)$ and prove that $\sup_\lambda v(\lambda) \leq 10d^5$, thus proving the lemma. Recall further from the proof of Lemma 14 our formula for T

$$T = 1 + \sum_{e \in \delta(e_0)} I_e \cdot T_e.$$

Therefore,

$$\begin{aligned}v(\lambda) &= \mathbb{E} \left(1 + \sum_{e \in \delta(e_0)} I_e \cdot T_e \middle| r(e_0) = \lambda \right)^2 \\ &= 1 + 2\mathbb{E} \left(\sum_{e \in \delta(e_0)} I_e \cdot T_e \middle| r(e_0) = \lambda \right) + \mathbb{E} \left(\sum_{e \in \delta(e_0)} \sum_{f \in \delta(e_0)} I_e \cdot I_f \cdot T_e \cdot T_f \middle| r(e_0) = \lambda \right) \\ &\leq 2\mathbb{E} \left(1 + \sum_{e \in \delta(e_0)} I_e \cdot T_e \middle| r(e_0) = \lambda \right) + \mathbb{E} \left(\sum_{e \neq f} I_e \cdot I_f \cdot T_e \cdot T_f \middle| r(e_0) = \lambda \right) + \mathbb{E} \left(\sum_{e \in \delta(e_0)} I_e \cdot T_e^2 \middle| r(e_0) = \lambda \right).\end{aligned}$$

The first term is simply $2\mathbb{E}(T | r(e_0) = \lambda) = 2t(\lambda)$ and is therefore bounded by $4d$, due to Corollary 3.

To bound the second term, we drop the I_e and I_f . Then we note that T_e and T_f are independent, as they depend only on H_e and H_f respectively.

$$\begin{aligned}\mathbb{E} \left(\sum_{e \neq f} I_e \cdot I_f \cdot T_e \cdot T_f \middle| r(e_0) = \lambda \right) &\leq \sum_{e \neq f} \mathbb{E}(T_e \cdot T_f | r(e_0) = \lambda) \\ &= \sum_{e \neq f} \mathbb{E}T_e \cdot \mathbb{E}T_f \\ &\leq d(d-1) \cdot (\mathbb{E}T)^2 \\ &\leq 4d^4,\end{aligned}$$

again by Corollary 3.

The third term does not admit to an outright bound. However we can express it recursively in terms of $v(\mu)$. Note, as in the proof of Lemma 14, that I_e and T_e are independent when conditioned on the rank of e .

$$\begin{aligned}\mathbb{E} \left(\sum_{e \in \delta(e_0)} I_e \cdot T_e^2 \middle| r(e_0) = \lambda \right) &= \sum_{e \in \delta(e_0)} \int_0^\lambda \mathbb{E}(I_e \cdot T_e^2 | r(e) = \mu) d\mu \\ &= \sum_{e \in \delta(e_0)} \int_0^\lambda \mathbb{E}(I_e | r(e) = \mu) \cdot \mathbb{E}(T_e^2 | r(e) = \mu) d\mu \\ &= d \int_0^\lambda x^{-1}(\mu) v(\mu) d\mu.\end{aligned}$$

Therefore, the full recursive inequality for $v(\lambda)$ is

$$v(\lambda) \leq 4d + 4d^4 + d \int_0^\lambda x^{-1}(\mu) v(\mu) d\mu \leq 5d^4 + d \int_0^\lambda x^{-1}(\mu) v(\mu) d\mu,$$

for $d \geq 5$. This is very similar for to the recursive formula for $t(\lambda)$ seen in the proof of Lemma 14. Let $\tilde{v}(\lambda) = v(\lambda)/(5d^4)$. Then

$$v(\lambda) \leq 1 + d \int_0^\lambda x^{-1}(\mu) v(\mu) d\mu.$$

This is now identical to the formula for $t(\lambda)$ (with an inequality instead of the equality), so $\tilde{v}(\lambda) \leq t(\lambda)$ by Grönwall's inequality, since $x^{-1}(\mu) \geq 0$. So $v(\lambda) \leq 5d^4 t(\lambda) \leq 10d^5$ as claimed. \square

Proof of Corollary 6: Let $T = T_{e_0}$ and $T_i = T_{e^{(i)}}$.

$$\begin{aligned} \mathbb{E}[T^2] &\leq \mathbb{E}\left[\left(1 + \sum_{i=1}^{\epsilon d} T_i\right)^2\right] \\ &= 1 + 2\mathbb{E}\left[\sum_{i=1}^{\epsilon d} T_i\right] + \mathbb{E}\left[\sum_{i \neq j}^{\epsilon d} T_i \cdot T_j\right] + \mathbb{E}\left[\sum_{i=1}^{\epsilon d} T_i^2\right] \\ &\leq 1 + 2\epsilon d \cdot \mathbb{E}[T_1] + (\epsilon d)^2 \mathbb{E}[T_1]^2 + \epsilon d \cdot \mathbb{E}[T_1^2] \\ &\leq 1 + 4\epsilon d^2 + 4\epsilon^2 d^4 + 10\epsilon d^6, \end{aligned}$$

by Corollary 3 and Lemma 17. This can then be upper bounded by $11\epsilon d^6$ for $d \geq 5$. \square

D Details omitted from Section 8

D.1 Proof of Theorem 9

We now provide the formal analysis of the total variation distance between $m^{1-\epsilon}$ edge-samples from graphs sampled from our hard distributions \mathcal{D}^{YES} and \mathcal{D}^{NO} .

Proof of Theorem 9: We begin by defining random variables A_1, A_2, B_1 , and B_2 that contain partial information about the iid stream under the YES and NO cases respectively. Let A_i be a random variable in $\mathcal{A} \stackrel{\text{def}}{=} ([r] \cup \{\star\})^{m^{1-\epsilon}} \times \mathbb{N}^r$, where the j^{th} coordinate of the first half of A_i (the part in $([r] \cup \{\star\})^{m^{1-\epsilon}}$) signifies which gadget (if any) the j^{th} edge of the stream belongs to, the coordinate being \star if it belongs to the clique. The j^{th} coordinate of the second half of A_i (the part in \mathbb{N}^r) signifies the number of *distinct* edges from $V_j \times V_j$ sampled throughout the stream. Furthermore, let B_i be a vector of length $r+1$, where the j^{th} coordinate signifies the isomorphism class of sampled edges of the j^{th} gadget and the last coordinate signifies the isomorphism class of the subsampled clique. Let the support of B_i be \mathcal{B}

With slight abuse of notation, for $i \in \{1, 2\}$ let

$$\begin{aligned} p_i(a, b, c) &:= \mathbb{P}[A_i = a \wedge B_i = b \wedge C_i = c] \\ p_i(a) &:= \mathbb{P}[A_i = a] \\ p_i(b) &:= \mathbb{P}[B_i = b] \\ p_i(c) &:= \mathbb{P}[C_i = c] \\ p_i(b|a) &:= \mathbb{P}[B_i = b | A_i = a] \\ p_i(c|a, b) &:= \mathbb{P}[C_i = c | A_i = a \wedge B_i = b]. \end{aligned}$$

Again, we are interested in the total variation distance between C_1 and C_2 , which satisfies

$$\begin{aligned} \|C_1 - C_2\|_{\text{TV}} &\leq \|(A_1, B_1, C_1) - (A_2, B_2, C_2)\|_{\text{TV}} \\ &= \frac{1}{2} \sum_{(a, b, c) \in \mathcal{A} \times \mathcal{B} \times \mathcal{C}} |p_1(a, b, c) - p_2(a, b, c)| \\ &= \frac{1}{2} \sum_{(a, b, c) \in \mathcal{A} \times \mathcal{B} \times \mathcal{C}} |p_1(a)p_1(b|a)p_1(c|a, b) - p_2(a)p_2(b|a)p_2(c|a, b)| \end{aligned}$$

First, observe that there is no discrepancy between $p_1(a)$ and $p_2(a)$ as the distributions of A_1 and A_2 are identical. Notice that the probability of a given iid edge being in a specific gadget or in the clique depends only on the number of edges of that gadget or the number of edges of the cliques. The clique contains $\binom{w}{2}$ edges in both the YES and NO cases. Also G and H have the same number of edges (simply apply the guarantee of [Theorem 10](#) with K being a single edge), so all gadgets have the same number of edges as well. $p_1(a) = p_2(a) =: p(a)$.

$$\begin{aligned} \|C_1 - C_2\|_{\text{TV}} &= \frac{1}{2} \sum_{a \in \mathcal{A}} p(a) \sum_{(b,c) \in \mathcal{B} \times \mathcal{C}} |p_1(b|a)p_1(c|a,b) - p_2(b|a)p_2(c|a,b)| \\ &\leq \mathbb{P}[\mathcal{E}] + \frac{1}{2} \sum_{a \in \mathcal{A}'} p(a) \sum_{(b,c) \in \mathcal{B} \times \mathcal{C}} |p_1(b|a)p_1(c|a,b) - p_2(b|a)p_2(c|a,b)| \end{aligned}$$

where \mathcal{A}' is the set of outcomes of A_i in accordance with $\bar{\mathcal{E}}$. Recall that

$$\mathcal{E} \stackrel{\text{def}}{=} \{\exists i \in [r] : \text{edges between vertices of } V_i \text{ appear more than } k \text{ times in the stream}\}.$$

Consider now the discrepancy between $p_1(b|a)$ and $p_2(b|a)$. Again, we will prove that the two distributions are equivalent, as long as the value of A_i being conditioned on is in \mathcal{A}' . Consider B_i to be $(B_i^{(1)}, B_i^{(2)}, \dots, B_i^{(r)}, B_i^*)$, where $B_i^{(j)}$ represents the isomorphism class of the sampled version of the j^{th} gadget and B_i^* is the isomorphism class of the sampled version of the clique. Note that the coordinates of B_i are independent conditioned on an outcome of A_i . Clearly, the distributions of B_1^* and B_2^* are identical. Consider now the distributions of $B_1^{(j)}$ and $B_2^{(j)}$ conditioned on $A_1 = A_2 = a \in \mathcal{A}'$. Conditioning on an outcome in \mathcal{A}' fixes the size of the sampled subgraph to some $l \leq k$, which means the support of $p_i(b|a)$ is some set of graphs of size l . For any specific graph K in the support, we know that the number of subgraphs of G and H isomorphic to K are equal (by the guarantee of [Theorem 10](#)); let this number be X . Also let the number of edges in a gadget be Y . Then $\mathbb{P}[B_1^{(j)} = [K] | A_1 = a] = \mathbb{P}[B_2^{(j)} = [K] | A_2 = a] = X / \binom{Y}{l}$. Thus $p_1(b|a) = p_2(b|a) =: p(b|a)$ for every $a \in \mathcal{A}'$.

$$\|C_1 - C_2\|_{\text{TV}} \leq \frac{1}{10} + \frac{1}{2} \sum_{(a,b) \in \mathcal{A}' \times \mathcal{B}} p(a)p(b|a) \sum_{c \in \mathcal{C}} |p_1(c|a,b) - p_2(c|a,b)|$$

Finally, consider the discrepancy between $p_1(c|a,b)$ and $p_2(c|a,b)$. We will, yet again, prove that the two distributions are identical when conditioned on any $(a,b) \in \mathcal{A}' \times \mathcal{B}$. Having conditioned on $A_i = a$ and $B_i = b$ the following are set about the stream: for every gadget, as well as the clique, we know the placement and number of the edges in the stream, and we know the isomorphism class of the subsampled gadget (or clique). For every gadget (or clique) with subsampled isomorphism-class $[K]$, we don't know the particular embedding of K into V_j (or V_K) that produces the subsampled gadget (or clique), and we also don't know the order and multiplicity with which these edges arrive. Thanks to the fact that all gadgets were uniformly randomly permuted in their embedding into V in the construction of \mathcal{D}^{YES} and \mathcal{D}^{NO} , the embedding of K into V_j is also uniformly random. (The clique is completely symmetric and need not be permuted.) Furthermore, since the stream is iid, conditioned on the set of edges in $V_j \times V_j$ that must appear, their order and multiplicity is drawn from the same distribution, regardless of whether we are in the YES or NO case. Therefore, for any $(a,b,c) \in \mathcal{A}' \times \mathcal{B} \times \mathcal{C}$, $p_1(c|a,b) = p_2(c|a,b) =: p(c|a,b)$.

$$\|C_1 - C_2\|_{\text{TV}} \leq \frac{1}{10} + \frac{1}{2} \sum_{(a,b) \in \mathcal{A}' \times \mathcal{B}} p(a)p(b|a) \sum_{c \in \mathcal{C}} |p(c|a,b) - p(c|a,b)| = \frac{1}{10}$$

□

D.2 Proof of [Lemma 23](#)

Our proof of [Lemma 23](#) is built on Theorem 3.4. of [\[LPS88\]](#) and a result from [\[CH12\]](#). We next restate the first result.

Theorem 14 ([\[LPS88\]](#)). *For any distinct primes p and q congruent to 1 modulo 4, there exists a group $\mathcal{G}^{p,q}$ with a set S of generator elements with the following properties: $|\mathcal{G}^{p,q}| \in [q(q^2 - 1)/2, q(q^2 - 1)]$; $|S| = p + 1$; and, $\mathcal{G}^{p,q}$ has girth at least $2 \log_p(q/4)$.*

Theorem 15 ([CH12]). *For any $x \geq 7$, the interval $(x, 2x]$ contains a prime number congruent 1 modulo 4.*

Lemma 23. *For any parameters g and l , there exists a group \mathcal{G} of size $l^{O(g)}$ along with a set of generator elements S of size at least l , such that the associated Cayley graph (Definition 12) has girth at least g .*

Proof. If $l < 7$, let $p = 13$. Otherwise, if $l \geq 7$, let p be a prime number congruent 1 modulo 4 from the interval $[l, 2l]$. By Theorem 15, such p exists. Let q be a prime number congruent 1 modulo 4 from the interval $[4p^g, 8p^g]$. Again by Theorem 15 and recalling that $p \geq 2$, such q exists. The statement now follows by Theorem 14. \square

E Proofs omitted from Section 9

Proof of Lemma 26: By symmetry we may assume that $p \leq 1/2$. We consider 3 cases:

Case 1: $\epsilon \leq -p/3$. With this constraint

$$D_{KL}(\text{Ber}(p + \epsilon) \parallel \text{Ber}(p)) \leq D_{KL}(\text{Ber}(0) \parallel \text{Ber}(p)) = \log\left(\frac{1}{1-p}\right) \leq \frac{p}{1-p}.$$

Therefore the lemma statement is always satisfied.

Case 2: $\epsilon \geq 1/4$. With this constraint $D_{KL}(\text{Ber}(p + \epsilon) \parallel \text{Ber}(p)) \leq D_{KL}(\text{Ber}(1) \parallel \text{Ber}(p)) = \log\left(\frac{1}{p}\right) \leq \frac{1}{p}$. Therefore, the lemma statement is always satisfied.

Case 3: $\epsilon \in [-p/3, 1/4]$. In that case we have

$$D_{KL}(\text{Ber}(p + \epsilon) \parallel \text{Ber}(p)) = -(p + \epsilon) \log\left(\frac{p}{p + \epsilon}\right) - (1 - p - \epsilon) \log\left(\frac{1 - p}{1 - p - \epsilon}\right) \quad (64)$$

$$= -(p + \epsilon) \log\left(1 - \frac{\epsilon}{p + \epsilon}\right) - (1 - p - \epsilon) \log\left(1 + \frac{\epsilon}{1 - p - \epsilon}\right) \quad (65)$$

$$\leq -(p + \epsilon) \left(-\frac{\epsilon}{p + \epsilon} - \frac{4\epsilon^2}{(p + \epsilon)^2}\right) - (1 - p - \epsilon) \left(\frac{\epsilon}{1 - p - \epsilon} - \frac{4\epsilon^2}{(1 - p - \epsilon)^2}\right) \quad (66)$$

$$= \frac{4\epsilon^2}{(p + \epsilon)(1 - p - \epsilon)} \quad (67)$$

$$\leq \frac{16}{p(1 - p)} \quad (68)$$

Here Eq. (66) follows from Taylor's theorem. Indeed, By the restriction on the range of ϵ , both $-\epsilon/(p + \epsilon)$ and $\epsilon/(1 - p - \epsilon)$ are in the interval $[-1/2, \infty)$. On this interval the function $\log(1 + x)$ is twice differentiable and the absolute value of its second derivative is bounded by 4, therefore

$$x - 4x^2 \leq \log(1 + x) \leq x + 4x^2.$$

\square

Proof of Lemma 27: We assume without loss of generality that $r \leq 1/2$. Let $\tilde{r} := \text{Padding}(r, \epsilon)$. Then \tilde{r} is also less than half and in fact $\tilde{r} = \max(r, \epsilon)$. Let $\eta_1 = |p - q|$, $\eta_2 = |q - \tilde{r}|$ and $\eta_3 = |p - \tilde{r}|$. For simplicity we will denote $D_{KL}(\text{Ber}(x) \parallel \text{Ber}(y))$ as $D_{KL}(x \parallel y)$ during this proof. By Lemma 26, in order to establish the result of the lemma it suffices to show that

$$\eta_3^2 \leq O(\epsilon) \tilde{r}(1 - \tilde{r}). \quad (69)$$

Note that the term $(1 - \tilde{r})$ is in $[1/2, 1]$ and can be disregarded.

We will use the following facts throughout the proof:

Fact 1. *For all $x \in \mathbb{R}$,*

$$\log(1 + x) \leq x.$$

Fact 2. For all $x \leq 1$,

$$\log(1+x) \leq x - \frac{x^2}{4}.$$

Fact 3. For all $x \in [0, 1/2]$,

$$D_{KL}(0\|x) \leq 2x.$$

Fact 4. For all $x \in [0, 1/2]$,

$$D_{KL}(x\|2x) \geq \frac{x}{4}.$$

Indeed,

$$D_{KL}(x\|2x) = -x \log\left(\frac{2x}{x}\right) - (1-x) \log\left(1 - \frac{x}{1-x}\right) \geq -x \log 2 + (1-x) \cdot \frac{x}{1-x} = x \cdot (1 - \log 2) \geq \frac{x}{4},$$

by **Fact 1**.

We will differentiate six cases depending on the ordering of p , q and \tilde{r} . However, four of these, the ones where q is not in the middle, will be very simple.

Case 1: $p \leq \tilde{r} \leq q$. Then,

$$D_{KL}(p\|\tilde{r}) \leq D_{KL}(p\|q) \leq \epsilon.$$

Case 2: $\tilde{r} \leq p \leq q$. Then,

$$D_{KL}(p\|\tilde{r}) \leq D_{KL}(q\|\tilde{r}) \leq D_{KL}(q\|r) \leq \epsilon.$$

Case 3: $\tilde{q} \leq p \leq \tilde{r}$. Then, if $\tilde{r} = r$,

$$D_{KL}(p\|\tilde{r}) \leq D_{KL}(q\|\tilde{r}) = D_{KL}(q\|r) \leq \epsilon.$$

On the other hand, if $\tilde{r} = \epsilon$,

$$D_{KL}(p\|\tilde{r}) \leq D_{KL}(0\|\epsilon) = -\log(1-\epsilon) \leq 2\epsilon,$$

by **Fact 3** since $\epsilon \leq 1/2$.

Case 4: $\tilde{q} \leq \tilde{r} \leq p$. Then,

$$D_{KL}(p\|\tilde{r}) \leq D_{KL}(p\|q) \leq \epsilon.$$

Case 5: $p \leq q \leq \tilde{r}$. We consider two subcases.

(a.) $p \leq 4\epsilon$. Then q cannot be greater than 8ϵ . Indeed this would mean by **Fact 4** that

$$D_{KL}(p\|q) > D_{KL}(4\epsilon\|8\epsilon) \geq \epsilon,$$

which is a contradiction. Similarly, r cannot be greater than 16ϵ . Indeed this would mean by **Fact 4** that

$$D_{KL}(q\|r) > D_{KL}(8\epsilon\|16\epsilon) \geq 2\epsilon,$$

which is also a contradiction. Ultimately, $\tilde{r} \leq 16\epsilon$, so

$$D_{KL}(p\|\tilde{r}) \leq D_{KL}(0\|16\epsilon) \leq 32\epsilon$$

by **Fact 3** since $16\epsilon \leq 1/2$.

(b.) $p \geq 4\epsilon$. Note that $\tilde{r} = r$. Let us bound η_1 . First note that η_1 cannot be greater than p due to [Fact 4](#). We will further show that η_1 in fact cannot be greater than $2\sqrt{p\epsilon}$.

$$\begin{aligned}
\epsilon &\geq D_{KL}(p\|q) \\
&= -p \log\left(1 + \frac{\eta_1}{p}\right) - (1-p) \log\left(1 - \frac{\eta_1}{1-p}\right) \\
&\geq -p \left(\frac{\eta_1}{p} - \frac{\eta_1^2}{4p^2}\right) - (1-p) \left(-\frac{\eta_1}{1-p}\right) && \text{By Facts 1 and 2, since } \eta_1/p \leq 1, \\
&= \frac{\eta_1^2}{4p}.
\end{aligned}$$

An identical calculation shows that $\eta_2 \leq 2\sqrt{q\epsilon}$. Ultimately,

$$\begin{aligned}
\eta_3 &= \eta_1 + \eta_2 \\
&\leq 2\sqrt{p\epsilon} + 2\sqrt{q\epsilon} \\
&\leq 2\sqrt{p\epsilon} + 2\sqrt{(p + 2\sqrt{p\epsilon}) \cdot \epsilon} \\
&\leq 6\sqrt{p\epsilon} && \text{Since } p \geq \epsilon, \\
&\leq 6\sqrt{\tilde{r}\epsilon}.
\end{aligned}$$

From here [\(69\)](#) follows immediately.

Case 6: $\tilde{r} \leq q \leq p$. In this case, let us first bound η_2 . We will show that η_2 cannot be greater than $2\sqrt{q\epsilon}$.

$$\begin{aligned}
\epsilon &\geq D_{KL}(q\|r) \\
&\geq D_{KL}(q\|\tilde{r}) \\
&= -q \log\left(1 - \frac{\eta_2}{q}\right) - (1-q) \log\left(1 + \frac{\eta_2}{1-q}\right) \\
&\geq -q \left(-\frac{\eta_2}{q} - \frac{\eta_2^2}{4q^2}\right) - (1-q) \left(\frac{\eta_2}{1-q}\right) && \text{By Facts 1 and 2,} \\
&= \frac{\eta_2^2}{4q}.
\end{aligned}$$

This also implies that q is at most $6\tilde{r}$. Indeed, suppose $q = \gamma\tilde{r}$. Then

$$\begin{aligned}
\tilde{r} &= q - \eta_2 \\
&\geq q - 2\sqrt{q\epsilon} \\
&= \gamma\tilde{r} - 2\sqrt{\gamma\tilde{r}\epsilon} \\
&\geq (\gamma - 2\sqrt{\gamma}) \cdot \tilde{r},
\end{aligned}$$

since $\tilde{r} \geq \epsilon$. Therefore, $1 \geq \gamma - 2\sqrt{\gamma}$, so $\gamma \leq 6$. We conclude that $\eta_2 \leq 2\sqrt{6\tilde{r}\epsilon}$. An identical calculation shows that $\eta_1 \leq 2\sqrt{6q\epsilon} \leq 12\sqrt{\tilde{r}\epsilon}$. Ultimately,

$$\begin{aligned}
\eta_3 &= \eta_1 + \eta_2 \\
&\leq 2\sqrt{6\tilde{r}\epsilon} + 12\sqrt{\tilde{r}\epsilon} \\
&\leq 18\sqrt{\tilde{r}\epsilon}.
\end{aligned}$$

From here [\(69\)](#) follows immediately.

This concludes the proof of the lemma under all possible orderings of p , q and \tilde{r} . \square