Graph Streaming Lower Bounds for Parameter Estimation and Property Testing via a Streaming XOR Lemma*

Sepehr Assadi
Department of Computer Science
Rutgers University
New Brunswick, NJ, USA
sepehr.assadi@rutgers.edu

Vishvajeet N
Department of Computer Science
Rutgers University
New Brunswick, NJ, USA
vishva.jeet@rutgers.edu

ABSTRACT

We study **space-pass tradeoffs** in graph streaming algorithms for parameter estimation and property testing problems such as estimating the size of maximum matchings and maximum cuts, weight of minimum spanning trees, or testing if a graph is connected or cycle-free versus being far from these properties. We develop a new lower bound technique that proves that for many problems of interest, including all the above, obtaining a $(1+\varepsilon)$-approximation requires either $n^{\Omega(1)}$ space or $\Omega(1/\varepsilon)$ passes, even on highly restricted families of graphs such as bounded-degree planar graphs. For multiple of these problems, this bound matches those of existing algorithms and is thus (asymptotically) optimal.

Our results considerably strengthen prior lower bounds even for arbitrary graphs: starting from the influential work of [Verbin, Yu; SODA 2011], there has been a plethora of lower bounds for single-pass algorithms for these problems; however, the only multi-pass lower bounds proven very recently in [Assadi, Kol, Saxena, Yu; FOCS 2020] rules out sublinear-space algorithms with exponentially smaller $o(\log(1/\varepsilon))$ passes for these problems.

One key ingredient of our proofs is a simple **streaming XOR Lemma**, a generic hardness amplification result, that we prove: informally speaking, if a p-pass s-space streaming algorithm can only solve a decision problem with advantage $\delta > 0$ over random guessing, then it cannot solve XOR of ℓ independent copies of the problem with advantage much better than δ^{ℓ}. This result can be of independent interest and useful for other streaming lower bounds as well.

CCS CONCEPTS

• **Theory of computation → Streaming, sublinear and near linear time algorithms**; *Lower bounds and information complexity.*

KEYWORDS

Graph Streaming, Communication Complexity, XOR lemma, Property Testing, MAXCUT, Pointer Chasing

ACM Reference Format:

Sepehr Assadi and Vishvajeet N. 2021. Graph Streaming Lower Bounds for Parameter Estimation and Property Testing via a Streaming XOR Lemma. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC '21), June 21–25, 2021, Virtual, Italy.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3406325.3451110

1 INTRODUCTION

Consider an n-vertex undirected graph $G = (V, E)$ whose edges are arriving one by one in a stream. Suppose we want to process G with a streaming algorithm using small space (e.g., polylog(n) bits), and in a few passes (e.g., a small constant). How well can we *estimate* parameters of G such as size of maximum cuts and maximum matchings, weight of minimum spanning trees, or number of short cycles? How well can we perform *property testing* on G, say, decide whether it is connected or cycle-free versus being far from having these properties? These questions are highly motivated by the growing need in processing massive graphs and have witnessed a flurry of results in recent years: see, e.g., [8, 43–45, 47] on maximum cut, [2, 17, 20, 22, 42, 51, 52] on maximum matching size, [6, 7, 11, 13, 19, 41, 53] on subgraph counting, [18, 33, 34] on CSPs, and [21, 35, 55, 57] on property testing, among others (see also [3, 21, 64]).

Despite this extensive attention, the answer to these questions have remained elusive; except for a handful of problems and almost exclusively for single-pass algorithms, we have not yet found the "right" answers. For instance, consider property testing of connectivity: given a sparse graph G and a constant $\varepsilon > 0$, find if G is connected or requires at least $\varepsilon \cdot n$ more edges to become so. Huang and Peng [35] proved that for single-pass algorithms, $n^{1-\Theta(\varepsilon)}$ space is sufficient and necessary for this problem. But until very recently, it was even open if one could solve this problem in $O(\log n)$ space and two passes. This question was partially addressed by the first author, Kol, Saxena, and Yu [3] who proved that any algorithm for this problem requires $n^{\Omega(1)}$ space or $\Omega(\log(1/\varepsilon))$ passes. But this is still far from the only known upper bound of polylog(n) space and $O(1/\varepsilon)$ passes obtained via a streaming implementation of the algorithm of [16] (see [57]).

Our goal in this paper is to make further progress on understanding the limits of multi-pass graph streaming algorithms for

*A full version of the paper that includes all technical proofs is available on arXiv: https://arxiv.org/abs/2104.04908.

STOC '21, June 21–25, 2021, Virtual, Italy

ACM ISBN 978-1-4503-8053-9/21/06…$15.00
https://doi.org/10.1145/3406325.3451110

parameter estimation and property testing. We present a host of new multi-pass streaming lower bounds that in multiple cases such as property testing of connectivity, **achieve optimal lower bounds on the space-pass tradeoffs** for the given problems. At the core of our results, similar to [3, 64], is a new lower bound for a *"gap cycle counting"* problem, wherein the goal is to distinguish between graphs consisting of only "short" cycles or only "long" cycles. Our other streaming lower bounds follow by easy reductions from this problem.

Already a decade ago, Verbin and Yu [64] identified a gap cycle counting problem as an excellent intermediate problem for studying the limitations of graph streaming algorithms for estimation problems: Given a graph G and an integer k, decide if G is a disjoint union of k-cycles or $2k$-cycles. By building on [25], they proved that this problem requires $n^{1-O(1/k)}$ space in a single pass and used this to establish lower bounds for several other problems. This work has since been a source of insights and inspirations for numerous other streaming lower bounds, e.g. [2, 10, 14, 18, 22, 33–35, 40, 43–45, 47, 49]. These lower bounds were all for single-pass algorithms. Very recently, [3] proved that any p-pass streaming algorithm for gap cycle counting—and even a variant wherein the goal is to distinguish union of k-cycles from a Hamiltonian cycle—requires $n^{1-O(k^{-1/2p})}$ space; in particular, $\Omega(\log k)$ passes are needed to solve this problem with polylog(n) space. The work of [3] showed that a large body of graph streaming lower bounds for estimation problems can now be extended to multi-pass algorithms using simple reductions from these gap cycle counting problems.

A main question that was left explicitly open by both [3, 64] was to determine the *tight* space-pass tradeoff for these gap cycle counting problems (and by extension other streaming problems obtained via reductions). We partially resolve this question by proving an asymptotically tight lower bound for a more relaxed variant that allows for some "noise" in the input.

2 OUR CONTRIBUTIONS

Our main contribution is an asymptotically optimal multi-pass streaming lower bound for a *noisy* version of the gap cycle counting problem, wherein the graph consists of a disjoint union of either k-cycles or $2k$-cycles on $\Theta(n)$ vertices, plus vertex-disjoint paths of length $k-1$ (the "noise") on the remaining vertices; the goal as before is to distinguish between the two cases. We define the problem formally as follows (see also Figure 1).

Problem 1. ***Noisy Gap Cycle Counting Problem (NGC)***

Let $k, t \in \mathbb{N}^+$ and $n = 6t \cdot k$. In $\mathbf{NGC}_{n,k}$, we have an n-vertex graph G with the promise that G either contains (i) $2t$ vertex-disjoint k-cycles, or (ii) t vertex-disjoint $(2k)$-cycles; in both cases, the remaining vertices of G are partitioned into $4t$ vertex-disjoint paths of length $k-1$ (the "noise" part of the graph). The goal is to distinguish between these two cases.

2.1 Gap Cycle Counting with a Little Bit of Noise

We prove the following lower bound for **NGC**.

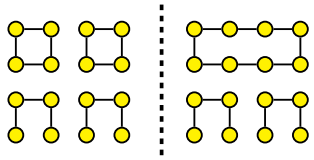

Figure 1: An illustration of the graphs in the noisy gap cycle counting problem. The actual graph consists of $\Theta(n/k)$ copies of these smaller subgraphs.

Result 1. For any constant $k > 0$, any p-pass streaming algorithm for the noisy gap cycle counting problem requires $n^{1-O(p/k)}$ space to succeed with large constant probability.

Result 1 obtains asymptotically optimal bounds for noisy gap cycle counting: on one end of the tradeoff, one can solve this problem in just one pass by sampling $\approx n^{1-1/k}$ random vertices and storing all their edges to find a k-cycle or a $(k+1)$-path. On the other end, we can simply "chase" the neighborhood of $O(1)$ random vertices in $\approx k$ passes to solve the problem. In the middle of these two extremes, there is the algorithm that samples $\approx n^{1-p/k}$ vertices and chase all of them in p passes and "stitch" them together to form k-cycles or $(k+1)$-paths. Result 1 matches all these tradeoffs asymptotically. Moreover, as a corollary, we obtain that any algorithm for this problem requires $n^{\Omega(1)}$ space or $\Omega(k)$ passes, *exponentially* improving the bounds of [3].

We remark that Verbin and Yu conjectured that any p-pass algorithm for this problem requires $n^{1-2/k}$ space as long as $k < p/2-1$[1] [64, Conjecture 5.4]. This conjecture as stated is too strong as the $O(n^{1-p/k})$ space algorithm above refutes it already for $p > 2$. However, 1 settles a qualitatively similar form of this conjecture which allows for an $n^{1-O(p/k)}$-space p-pass tradeoff.

2.2 Graph Streaming Lower Bounds from Noisy Gap Cycle Counting

We use our lower bound in Result 1 in a similar manner as prior work to prove several new graph streaming lower bounds. The difference is that we now have to handle the extra noise in the problem; it turns out however that, as expected, this noise does not have a serious effect on the reductions (it also helps that we prove Result 1 in a stronger form where, informally speaking, one endpoint of every noise path is already known to the algorithm. As a result, we can recover *all* graph streaming lower bounds of [3] with a much stronger guarantee:

Result 2. For any $\varepsilon > 0$, any p-pass algorithm for any of the following problems on n-vertex graphs requires $n^{1-O(\varepsilon \cdot p)}$ space:

[1] The conjecture of [64] is stated more generally for two-party communication protocols and for the no-noise version of the problem; the statement here is an immediate corollary of this conjecture.

(1) $(1+\varepsilon)$-approximation of maximum matching size, maximum cut value, maximum acyclic subgraph, and minimum spanning tree weight;
(2) property testing of connectivity, bipartiteness, and cycle-freeness for parameter ε.

Moreover, these lower bounds continue to hold even on bounded-degree planar graphs.

Prior to our work, $n^{1-O(\varepsilon)}$ space lower bounds for single-pass algorithms have been obtained in [43, 47] for maximum cut, [14, 22] for maximum matching, [18, 34] for maximum acyclic subgraph, [23, 35] for minimum spanning tree, and [35] for the property testing problems. These results were recently extended by [3] to p-pass algorithms with the space of $n^{1-O(\varepsilon^{1/2p})}$ and thus $\Omega(\log(1/\varepsilon))$ passes for $n^{o(1)}$-space algorithms. Our Result 2 exponentially improves the dependence on number of passes in [3], and in particular implies that any $n^{o(1)}$-space streaming algorithm for these problems require $\Omega(1/\varepsilon)$ passes. For multiple of these problems, this bound can be matched by already known upper bounds and is thus optimal. We elaborate on these results further in Section 7.

We conclude this part by remarking that many of the problems we consider in Result 2 have been also studied in *random order streams*; see, e.g. [21, 42, 46, 55, 57]. In particular, Monemizadeh et. al. [55] showed that $(1+\varepsilon)$-approximation of matching size (in bounded-degree graphs) can be done in $O_\varepsilon(\log n)$ space and a single pass if the edges are arriving in a random order; similar bounds were obtained by Peng and Sohler [57] for approximating the weight of minimum spanning tree (in bounded-weight graphs) and property testing of connectedness (see also the work of Czumaj et.al. [21] for a recent generalization of these results). Our Result 2 thus demonstrate just how much harder solving these problems is in adversarial-order streams even with almost $1/\varepsilon$ passes.

2.3 Streaming XOR Lemma

A key part of our proof of Result 1 is a general hardness amplification step: Let f be a Boolean function over a distribution $x \sim \mu$; for any integer $\ell > 1$, consider the ℓ-fold-XOR-composition of f over the distribution of inputs $x_1, \ldots, x_\ell \sim \mu^\ell$, namely, $f^{\oplus\ell} := \mathrm{XOR}_\ell \circ f = f(x_1) \oplus \cdots \oplus f(x_\ell)$. How much harder is to compute $f^{\oplus\ell}$ compared to f? Notice that if solving f (with certain resources) has success probability $\leq 1/2 + \delta$, and that all the algorithm for $f^{\oplus\ell}$ does is to solve each $f(x_i)$ independently and take their XOR, then its success probability would be $\approx 1/2 + \delta^\ell$. This is simply because XOR of ℓ independent random bits with bias δ only has bias $\approx \delta^\ell$. Can a more clever strategy (with the same resources) beat this naive way of computing $f^{\oplus\ell}$?

These questions are generally referred to as *XOR Lemmas* and have been studied extensively in different settings like circuit complexity [26, 27, 36, 37, 48, 66], communication complexity [26, 62, 65], and query complexity [12, 61, 62]. However, despite the extensive attention that similar hardness amplification questions such as direct sum and direct product have received in the streaming model (see, e.g. [5, 9, 29, 38, 39, 54, 58, 60] and references therein), we are not aware of any type of XOR Lemma for streaming algorithms. Thus, an important contribution of our work is to prove exactly such a result; considering its generality, we believe this result to be of independent interest.

Result 3. Suppose any p-pass s-space streaming algorithm for f over a distribution $\sigma \sim \mu$ succeeds with probability $\leq 1/2 + \delta$. Then, any p-pass s-space algorithm for $f^{\oplus\ell}$ over the *concatenation* of streams $\sigma_1, \cdots, \sigma_\ell \sim \mu^\ell$ only succeeds with probability $\leq 1/2 \cdot (1 + (2\delta)^\ell)$.

Let us now mention how Result 3 is used in the proof of Result 1. Consider the following problem: given a graph G in (noisy) gap cycle counting and a single vertex $v \in G$, "chase" the depth-$(k/2)$ neighborhood of v to see if they form a k-cycle or a $(k+1)$-path. This problem is quite similar to the pointer chasing problem studied extensively in communication complexity and streaming, e.g., in [1, 4, 15, 24, 28, 30–32, 39, 56, 59, 67] (see Definition 3.3). The gap cycle counting problem then can be thought of as $\approx n/k$ instances of this problem that are highly *correlated*: they are all in the same graph and they all either form a k-cycle or a $(k+1)$-path. The first step of our lower bound is an argument that "decorrelates" these instances which implies that one of them should be solved with probability of success $1/2 + \Omega(k/n)$. This probability of success is still way below the threshold for any of the standard pointer chasing lower bounds to kick in. This is where we use our streaming XOR Lemma: we give a reduction that embeds XOR of ℓ instances of depth-$(k/2\ell)$ pointer chasing as a single depth-$(k/2)$ instance; applying our Result 3 then reduces our task to proving a lower bound for pointer chasing with probability of success $1/2 + \Omega((k/n)^{1/\ell})$ (in $k/2\ell$ passes), which brings us to the "standard" territory. The last step is then to prove this lower bound over our hard instances which are different from standard ones, e.g., in [31, 56, 67].

3 PRELIMINARIES

Notation. For a Boolean function f and integer $\ell \geq 1$, we use $f^{\oplus\ell}$ to denote the composition of f with the ℓ-fold XOR function, i.e., $f^{\oplus\ell}(x_1, \ldots, x_\ell) = f(x_1) \oplus \cdots \oplus f(x_\ell)$. Throughout the paper, we denote input stream by σ and $|\sigma|$ denote the length of the stream. For any two streams σ_1, σ_2, we use $\sigma_1 \,||\, \sigma_2$ to denote the $|\sigma_1|+|\sigma_2|$ length stream obtained by concatenating σ_2 at the end of σ_1. When it can lead to confusion, we use sans serif font for random variables (e.g. X) and normal font for their realization (e.g. X). We use $\mathrm{supp}(\mathsf{X})$ to denote the support of random variable X. For a 0/1-random variable X, we define the *bias* of X as $\mathrm{bias}(\mathsf{X}) := |\Pr(\mathsf{X}=0) - \Pr(\mathsf{X}=1)|$.

Information theory. For random variables X, Y, $\mathbb{H}(\mathsf{X})$ denotes the Shannon entropy of X, $\mathbb{I}(\mathsf{X};\mathsf{Y})$ denotes the mutual information, $\|\mathsf{X}-\mathsf{Y}\|_{\mathrm{tvd}}$ denotes the total variation distance between the distributions of X, Y, and $\mathbb{D}(\mathsf{X} \,||\, \mathsf{Y})$ is their KL-divergence.

Streaming algorithms. For the purpose of our lower bounds, we shall work with a more powerful model than what is typically considered the streaming model, which we define as follows:

*Definition 3.1 (**Streaming algorithms**).* For any integers $n, p, s \geq 1$, we define a p-pass s-space streaming algorithm working on a length-n stream $\sigma = (x_1, \ldots, x_n)$ as a $(n+1)$-player communication protocol between players $P_0, \ldots, P_n$ wherein:

(1) Each player P_i for $i \geq 1$ receives x_i as the *input* and player P_0 has no input; the players also have access to *private randomness*.
(2) The players communicate in this order: P_0 sends a message to P_1 who sends a message to P_2 and so on up until P_n who sends a message to P_0; this constitutes one *pass* of the algorithm. The players then continue like this for p passes and at the end, P_0 outputs the answer.
(3) Each message of a player in a given round is an *arbitrary* function of its input, *all* the messages received by this player so far, and its private randomness and has size s bits *exactly*.

(We note that this model is non-uniform and is defined for each choice of n individually.)

We also need some structure on the family of graphs we work with, in particular the ones defined as follows:

*Definition 3.2 (**Layered Graph**).* For any integers $w, d \geq 1$, we define a (w, d)-*layered graph*, with *width* w and *depth* d, as any graph $G = (V, E)$ with the following properties:

(1) Vertices V consist of $d+1$ *layers* of vertices $V_1, \ldots, V_{d+1}$, each of size w.
(2) Edges E consist of d *matchings* $M_1, \ldots, M_d$ where M_i is a perfect matching between M_i, M_{i+1}.

For any vertex $v \in V_1$, we use $P(v)$ to denote the *unique* vertex reachable from v in V_{d+1}. Moreover, by a *random* layered graph, we mean a layered graph whose matchings are chosen *uniformly at random* and *independently* but the partitioning of vertices into the layers is fixed.

In our proof, we also work the *pointer chasing* problem (although with several non-standard aspects). We define this problem as follows:

*Definition 3.3 (**Pointer Chasing (PC)**).* Let $m, b \in \mathbb{N}^+$. In $\mathbf{PC}_{m,b}$, we have a (m, b)-layered graph on layers $V_1, \ldots, V_{b+1}$, an arbitrary vertex $s \in V_1$, and an arbitrary equipartition X, Y of V_{b+1}. The goal is to decide whether $P(s) \in V_{b+1}$ belongs to X (a X-instance) or Y (a Y-instance).

4 TECHNICAL OVERVIEW

We state our main streaming lower bound for the noisy gap cycle counting problem that formalizes Result 1 and proceed to give an overview of our techniques in this section.

Theorem 4.1. *For every $k \in \mathbb{N}^+$, any p-pass streaming algorithm for Noisy Gap Cycle Counting $\mathbf{NGC}_{n,k}$ with probability of success at least $2/3$ requires $\Omega\left(\frac{1}{p^5} \cdot (n/k)^{1-O(p/k)}\right)$ space.*

Note that for this lower bound to be non-trivial, both k needs to be at least some large constant, and p should be smaller than k by a similar factor.

We first design a hard input distribution for $\mathbf{NGC}_{n,k}$ and prove its useful properties for our purpose. Our hard input distribution is constructed as follows. We first sample a random (w, d)-layered graph G_0 for parameter $w = 3t$ and $d = \frac{k-2}{2}$ for $\mathbf{NGC}_{n,k}$ *conditioned on the following event*:

- Let $S \subseteq V_1$ be a fixed subset of size t, and X, Y be a fixed equipartition of V_{d+1} (say, both are the lexicographically-first option); then $\{P(v) \mid v \in S\}$ is either a subset of X or Y.

We construct the final graph G from four *identical* copies of G_0 (on disjoint sets of vertices), plus some fixed gadget so that it satisfies the following property: when all vertices $v \in S$ have $P(v) \in X$, the resulting graph G has $2t$ cycles of length k each; otherwise, it has t cycles of length $2k$ instead; in both cases, the graph G also has $4t$ paths of length $k-1$. We now present the formal description of the distribution (see also Figure 2).

Distribution 1. The distribution μ_{NGC} for $\mathbf{NGC}_{n,k}$ for given parameters n, k and $t = n/6k$.

(1) Let $d := \frac{(k-2)}{2}$ and sample a random $(3t, d)$-layered graph G_0 on vertices $V_1, \ldots, V_{d+1}$ and matchings $M_1, \ldots, M_d$ *conditioned* on the following event:
   - Let S be a fixed t-subset of V_1 and X, Y be a fixed equipartition of V_{d+1}. Then, $\{P(v) \mid v \in S\}$ is entirely a subset of X (a X-*instance*) or a subset of Y (a Y-*instance*).
(2) Create the following graph $G = (V, E)$ on groups of vertices V_j^i for $i \in [4]$ and $j \in [d+1]$ using four identical copies of the graph sampled G_0 above:
   (a) For every $j \in [d+1]$, let V_j^i be the copies of V_j in G_0 and define S^i, X^i, Y^i as copies of S, X, Y, respectively (the same for all $i \in [4]$) – for any vertex $v \in G_0$ and $i \in [4]$, we use $copy(v, i)$ to denote the copy of v in V^i.
   (b) Connect V_j^i to V_{j+1}^i for any i, j by a matching M_j^i corresponding to M_j of G_0.
   (c) Connect S^1 to S^3, and S^2 to S^4 using identity perfect matchings, respectively. Similarly, connect X^1 to X^3 and X^2 to X^4, and Y^1 to Y^4 and Y^2 to Y^3 using identity perfect matchings, respectively (note the crucial change between the treatment of X^i and Y^i).
(3) The input stream consists of edges inserted in (2c) in some arbitrary order, followed by M_j^i in *decreasing* order of j and increasing order of i (the order inside each M_j^i is arbitrary), i.e., this part of the stream is $M_d^1 \,||\, \cdots \,||\, M_1^1 \,||\, \cdots \,||\, M_d^4 \,||\, \cdots \,||\, M_1^4$.

We present a streamlined overview for proving Theorem 4.1 in the following three steps.

Step one: Decorrelating the distribution. Recall that by our previous discussion, our task at this point is to prove a lower bound for the following problem: Given a $(3t, d)$-layered graph G_0 and a set S of t vertices in the first layer, decide whether following edges of G_0 takes these vertices to X or Y in the last layer. If we look at a single vertex $v \in S$, this problem is a pointer chasing problem along the edges of the d matchings of G_0. The challenge here is

that we are not solving any one pointer chasing problem though, but rather a collection of t *correlated* ones. This problem is quite simpler (algorithmically) than original pointer chasing as we only need to get "lucky enough" to chase one of them. Concretely, an algorithm that samples $\approx t^{1-1/d}$ edges of the graph has a constant probability of finding one complete path and solves the problem.

To bypass this challenge, we consider a generalized version of the distribution μ_{NGC} wherein every vertex in S has *almost* the same probability of ending up in the set X or Y, independent of the choice of other starting vertices. We prove that even though these distributions do not correspond to valid instances of **NGC**, still, if we run any algorithm for **NGC** over these inputs, it has to do some "non-trivial work": informally speaking, it will be able to solve the pointer chasing instance corresponding to one of the starting vertices with a probability of $1/2+\Omega(1/t)$ – this time however, this instance is independent of the choice of remaining vertices in S (owing to the introduction of noise). This hybrid argument allows us to reduce the problem to a low probability [of success] pointer chasing problem, which we tackle in the next step.

It is worth pointing out that this step matches the intuition that to solve **NGC**, we need to "find" at least one k-cycle or a $(k+1)$-path in the graph.

Step two: Applying the streaming XOR Lemma. The pointer chasing problem we now need to prove a lower bound for requires a really low probability of success, which is way below the threshold for any of the standard lower bounds to kick in. Our next step is then to apply our hardness amplification result in Result 3 to reduce this to a more standard pointer chasing problem with higher probability of success. This requires us to cast our pointer chasing instance as an XOR of several other *independent* pointer chasing instances.

To do this for p-pass algorithms, we "chop" the layered graph into $\approx k/p$ *consecutive* groups of $\approx p$ layers. we show how one can carefully connect these groups together to get $\approx k/p$ *independent* instances of pointer chasing in each group, so that the XOR of their answers determine the answer to the original problem. This step uses similar ideas as definition of Distribution 1 and some further randomization tricks. We can now apply our streaming XOR Lemma (Result 3) and reduce the problem to proving a lower bound for pointer chasing on depth $\approx p$ layered graphs with probability of success $1/2+1/t^{\Theta(p/k)}$, which is the content of the next step.

This step also matches the intuition that to solve **NGC** in p passes, we need to be able to "create" roughly k/p paths of length p inside the *same* k-cycle or $(k+1)$-path.

Step three: Lower bound for the single-copy problem. We are now in the familiar territory in which the goal is to prove a lower bound for a depth $\approx p$ pointer chasing problem with probability of success $1/2+1/t^{\Theta(p/k)}$. The main difference is that our distribution do not match that of standard lower bounds, say [31, 56, 59, 67], which can be made to work with layered graphs but need random degree-one graphs instead of random matchings (so higher entropy inputs). Nevertheless, we show that this can be managed with some further crucial modifications.

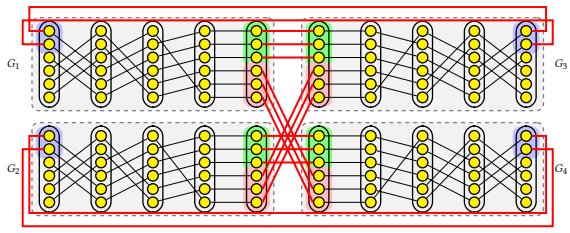

Figure 2: An illustration of Distribution 1 for $t = 2$ and $k = 10$. The thin (black) edges are formed by the random input matchings while thick (red) edges are input-independent. The final graph is obtained via four identical copies of a $(6, 4)$-layered graph and the sets S, X, Y are marked in each one. The input stream consists of the edges these matchings ordered from the inner matchings to the outer ones. The input drawn shows a k-cycle instance.

All in all, this step allows us to prove that solving pointer chasing on depth $(p+1)$ layered graphs in p passes and $n^{1-o(p/k)}$ space does not lead to success probability of $1/2+1/t^{\Theta(p/k)}$. Tracing back these steps and plugging in the parameters, concludes the proof of the theorem.

Finally, this step also matches the standard intuition that "finding" a path of length $> p$ in p passes, with large probability of success, is not possible in much less than near-linear space.

5 A LOWER BOUND FOR NGC

We now describe the details of each steps as outlined previously.

5.1 Step One: Decorrelating Distribution 1

Recall that our goal in this step is to reduce **NGC** over μ_{NGC} to solving **PC** (over a slightly smaller graph) albeit with a much lower probability of success. Consider the following distribution for **PC**:

Distribution 2. The distribution μ_{PC} for $\mathbf{PC}_{m,b}$ for given width and depth parameters m, b.

(1) Sample a random (m, b)-layered graph with an arbitrary vertex $s \in V_1$ and an arbitrary equipartition X, Y of V_{d+1} (say, both are lexicographically-first options).
(2) Let the input stream be $M_b \,||\, \cdots \,||\, M_1$ (with arbitrary orderings in each matching).

We prove the following lemma in this section.

Lemma 5.1. *Suppose there is a p-pass s-space streaming algorithm A for $\mathbf{NGC}_{n,k}$ on μ_{NGC} that succeeds with probability at least $2/3$. Then, there is a p-pass s-space streaming algorithm for $\mathbf{PC}_{m,b}$ on μ_{PC} for some even $m := \Theta(n/k)$ and $b := \frac{k-2}{2}$ with probability of success at least $\frac{1}{2}+\frac{1}{6m}$.*

For the rest of this section, we fix the algorithm A in Lemma 5.1 to use it in a reduction. Our reduction uses a hybrid argument

and thus is going to be *algorithm-dependent*, i.e., use A in a non-black-box way. To do so, we first need to define a family of hybrid distributions.

Recall the parameters $t = (n/6k)$ and $d = \frac{(k-2)}{2}$ in the definition of μ_{NGC}. For any vector $f = (f_1, \ldots, f_t) \in \{0,1\}^t$, we define the distribution $\mu(f)$ as follows:

- **Hybrid distribution $\mu(f)$**: Sample a random $(3t, d)$-layered graph G_0 (with fixed $S \subseteq V_1$ and equipartition X, Y of V_{d+1}) conditioned on the following event: "for any vertex $v_i \in S$, $P(v_i)$ belongs to X if $f_i = 0$ and belongs to Y if $f_i = 1$". Plug this graph G_0 in Distribution 1 instead and return the resulting stream for the created graph G.

With this definition, we have that $\mu_{\text{NGC}} = \frac{1}{2} \cdot \mu(0^t) + \frac{1}{2} \cdot \mu(1^t)$. The problem of working with $\mu(0^t)$ and $\mu(1^t)$ directly is that their **PC** instances are highly correlated (all vertices in S either go to X or to Y). Thus, it is unclear which instance is actually "solved". On the other hand, remaining distributions $\mu(f)$ may generate graphs that are not in the support of μ_{NGC} or even well-defined for **NGC**. Nevertheless, we will show that A still needs to do something non-trivial over these distributions: there is a pair of neighboring vectors g, h that differ in exactly one coordinate such that A is still able to distinguish between them, namely, "solve" the pointer chasing instance on their differing index (although with a much lower probability). We now formalize this.

Let $mem(A)$ denote the final content of the memory of A. Let $\mu(g)$ and $\mu(h)$ be any two distributions in the family above. With a slight abuse of notation, we say that A *distinguishes* between $\mu(g)$ and $\mu(h)$ with probability $p > 0$, if given a sample from either $\mu(g)$ or $\mu(h)$, we can run A over the sample and use maximum likelihood estimation of $mem(A)$ to decide which distribution it was sampled from with probability at least p. Define the following $t+1$ vectors:

$$f^0 = (0, \ldots, 0)$$

$$f^1 = (1, 0, \ldots, 0)$$

$$\vdots$$

$$f^i = (\underbrace{1, \ldots, 1}_{i}\, 0, \ldots, 0)$$

$$\vdots$$

$$f^t = (1, \ldots, 1)$$

We prove that A distinguishes between two consecutive distributions in this sequence.

Claim 5.2. *There is an index $i^* \in [t]$ such that A distinguishes between $\mu(f^{i^*-1})$ and $\mu(f^{i^*})$ with probability at least $\frac{1}{2} + \frac{1}{6t}$.*

Proof. Since A can solve **NGC** on instances drawn from $\mu_{\text{NGC}} = \frac{1}{2} \cdot \mu(0^t) + \frac{1}{2} \cdot \mu(1^t)$ with probability of success at least $2/3$, we have that,

$$\| \Big(mem(A) \mid \mu(f^0)\Big) - \big(mem(A) \mid \mu(f^t)\big) \|_{\text{tvd}} \geq 1/3, \qquad (1)$$

as the algorithm uses only $mem(A)$ at the end to output the answer (here $(mem(A) \mid \mu)$ denotes the distribution of $mem(A)$ conditioned on the input sampled from μ).

Suppose we run algorithm A on each of the distributions $\mu(f^i)$ (even though beside f^0, f^t, neither of them correspond to an **NGC** instance). Then, by triangle inequality,

$$\| \Big(mem(A) \mid \mu(f^0)\Big) - \big(mem(A) \mid \mu(f^t)\big) \|_{\text{tvd}}$$

$$\leq \sum_{i=1}^{t} \| \Big(mem(A) \mid \mu(f^{i-1})\Big) - \Big(mem(A) \mid \mu(f^i)\Big) \|_{\text{tvd}},$$

which, together with Equation (1), implies that there is an index $i^* \in [t]$ such that

$$\| \Big(mem(A) \mid \mu(f^{i^*-1})\Big) - \Big(mem(A) \mid \mu(f^{i^*})\Big) \|_{\text{tvd}} \geq \frac{1}{3t}. \qquad (2)$$

Thus, A distinguishes between $\mu(f^{i^*-1})$ and $\mu(f^{i^*})$ with probability $\geq \frac{1}{2} + \frac{1}{6t}$. ∎

Let us define our final distribution $\mu^* := \frac{1}{2} \cdot \mu(f^{i^*-1}) + \frac{1}{2} \cdot \mu(f^{i^*})$. Claim 5.2 suggests a way of solving instances of **PC** by embedding them in the index i^* of μ^* and running A over the resulting input. We now give a process for sampling from μ^* which is crucial for this embedding.

Claim 5.3. *The following process samples a $(3t, d)$-layered graph G_0 from the distribution of μ^*:*

1. *Sample $(t-1)$ vertex-disjoint paths from vertices in $S \setminus v_{i^*}$ to vertices in V_{d+1} conditioned on the event that "for any vertex $v_i \in S \setminus \{v_{i^*}\}$, $P(v_i)$ is in X if $f_i^{i^*} = 0$ and in Y if $f_i^{i^*} = 1$".*
2. *Let $c := |(i^* - 1) - (t - i^*)|$, the discrepancy in the number of 0's and 1's in both vectors f^{i^*-1}, f^{i^*} when we ignore index i^*. Sample c random vertex-disjoint path starting from $V_1 \setminus S$ to the remaining vertices of X in V_{d+1} if 0's of f^{i^*} are fewer that 1's, and to Y otherwise.*
3. *Sample a random $(2t+1-c, d)$-layered graph on the remaining vertices.*

Proof. For any vertex $v \in V_1$, define $\mathcal{P}(v)$ as the path starting from v and ending in V_{d+1}. Considering G_0 is a $(3t, d)$-layered graph consisting of perfect matchings between the layers, the paths $\mathcal{P}(v)$ are vertex-disjoint and of length $d+1$. As such, we can think of the process of sampling G_0 in μ^* as sampling these vertex-disjoint paths.

In step (1), we are sampling $\mathcal{P}(v)$ for all $v \in S \setminus v_{i^*}$. As $f_j^{i^*-1} = f_j^{i^*}$ for all $j \neq i^*$, this step can just sample these paths uniformly at random conditioned on an appropriate endpoint in X and Y for them. Thus far, the sampling process is the same as μ^*.

Let us now examine what happens to the choice of $\mathcal{P}(v_{i^*})$ at this point. Since μ^* is a uniform mixture of f^{i^*-1}, f^{i^*}, the path $\mathcal{P}(v_{i^*})$ should end up at either X or Y with the same probability. However, considering we already conditioned on $\mathcal{P}(v_j)$ for $j \neq i^*$, the number of remaining X and Y vertices are not equal. This means that $\mathcal{P}(v_{i^*})$ is *not* a uniformly random path in the rest of the graph.

The goal of step (2) is to fix this[2]. We can first sample $\mathcal{P}(v)$ for c vertices in $V_1 \setminus S$ so that they all end up in an X or Y vertex, depending on which of the sets has more remaining vertices. This equalizes the size of the remaining X and Y vertices, while keeping the distribution intact, using the randomness in choices of these c vertices.

Finally, at this point, we need to sample $\mathcal{P}(v)$ for remaining vertices conditioned on $\mathcal{P}(v_{i^*})$ having the same probability of landing in X or Y. Considering sizes of remainder of X and Y are equal, this can be done by sampling a uniform set of vertex-disjoint paths, or alternatively, a random layered graph on the remaining vertices which are $3t-(t-1)-c = 2t+1-c$. This is precisely what is done in step (3), concluding the proof. ∎

A simple corollary of the process in Claim 5.3 is the following conditional independence: let $\mathsf{R}_1, \mathsf{R}_2, \mathsf{R}_3$ denote the random variables for choices in steps (1), (2), (3) of this process; then, conditioned on any choice R_1, R_2 of $\mathsf{R}_1, \mathsf{R}_2$, the variable R_3 is distributed as a random $(2t+1-c, d)$-layered graph on the remaining vertices, with independent choice of edges, now that we conditioned on its vertices. Let $H_0 \subseteq G_0$, denote this subgraph. By Claim 5.2 and an averaging argument, there is a choice of R_1, R_2 such that,

$$\Pr_{H_0 \sim \mathsf{R}_3}\left(A \text{ distinguishes between } \mu(f^{i^*-1}), \mu(f^{i^*}) \mid R_1, R_2\right) \geq \frac{1}{2} + \frac{1}{6t}. \tag{3}$$

Distinguishing between $\mu(f^{i^*-1}), \mu(f^{i^*})$ is to decide whether $v_{i^*} \in V_1$, has $P(v_{i^*})$ in X or Y – this is equivalent to solving **PC** over the graph H_0 for $s = v_{i^*}$. We now use this to finalize our reduction and prove Lemma 5.1.

Proof of Lemma 5.1. Let c be the parameter in Claim 5.3 and note that since t is even (by construction of μ_{NGC}), c should be odd (as $c = |t - 2i^* + 1|$). Let $m := 2t+1-c$ which is an even number as desired and $b := d = \frac{(k-2)}{2}$; moreover note that since $c \leq t-1$, $m \geq t+2 = \Theta(n/k)$. We design a streaming algorithm B from A for $\mathbf{PC}_{m,b}$ for over the distribution μ_{PC}.

Given $G \sim \mu_{\text{PC}}$, algorithm B uses Claim 5.3 to create the graph

$$G_0 \sim \mu^* \mid \mathsf{R}_1 = R_1, \mathsf{R}_2 = R_2, \mathsf{H}_0 = G,$$

where R_1, R_2 are the choices in Equation (3). To be precise, by setting $H_0 = G$, we mean that the players of B pick a canonical mapping between vertices of G and H_0 such that $s = v_{i^*}$, the X-vertices (resp. Y-vertices) of G are mapped to X-vertices (Y-vertices), and each player in A with $e \in H_0$ has a unique player in B that simulates it. The players then run A over G_0 to distinguish $\mu(f^{i^*-1})$ from $\mu(f^{i^*})$ and output $P(s) \in X$ if the answer of A was $\mu(f^{i^*-1})$ and otherwise output $P(s) \in Y$.

The algorithm B is still a p-pass s-space algorithm (recall that in Definition 3.1, the players are computationally unbounded and so can do their part of creating the graph G_0 without any communication). By the independence property argued for Equation (3), the distribution of graphs G_0 above matches that of this equation. As such, B outputs the correct answer with probability $\frac{1}{2} + \frac{1}{6t} \geq \frac{1}{2} + \frac{1}{6m}$, finalizing the proof. ∎

[2] A simple analogy may help here: suppose we have four red balls and two green balls and we want to sample a ball uniformly so that its color is red or green with the same probability. We can first sample two red balls uniformly and throw them out and then sample a ball uniformly from the rest.

5.2 Step Two: Applying the Streaming XOR Lemma

By Lemma 5.1, our task is reduced to proving a low-probability lower bound for $\mathbf{PC}_{m,b}$ over the distribution μ_{PC}. Our goal in this step, is to use our streaming XOR Lemma in Result 3, to reduce this problem to another $\mathbf{PC}_{\hat{m},\hat{b}}$ problem over distribution μ_{PC} (for choices of $\hat{m}, \hat{b}$ as functions of m, b). We prove the following lemma in this section, which realizes our goal.

Lemma 5.4. *For every $m, b, \ell \in \mathbb{N}^+$ such that m is even and 2ℓ divides $b-1$, the following holds. Suppose there is a p-pass s-space algorithm A for $\mathbf{PC}_{m,b}$ that succeeds with probability at least $1/2+\delta$ on μ_{PC}. Then, there is a p-pass s-space algorithm for $\mathbf{PC}_{\hat{m},\hat{b}}$ over μ_{PC}, for some $\hat{m} = \frac{m}{2}$ and $\hat{b} = \frac{b-1}{2\ell} - 1$, that succeeds with probability at least $\frac{1}{2} \cdot (1 + (2\delta)^{1/\ell})$.*

The key to the proof of Lemma 5.4 is the streaming XOR Lemma; however, to be able to apply the XOR Lemma, we first need to cast $\mathbf{PC}_{m,b}$ as an XOR problem, which we do in the following, using a simple graph product.

Let us start by defining a simple graph product, which we call the *XOR product*: Given ℓ layered graphs $G_1, \ldots, G_\ell$ as instances of **PC** problem, this product generates a graph $H := \oplus_{i=1}^{\ell} G_i$ such that the answer to a **PC** problem on H is equal to XOR of answers to **PC** on $G_1, \ldots, G_\ell$. We define this product formally as follows.

XOR product. Suppose we have a set of $V := (V_1, \ldots, V_{d+1})$ of vertices, each of size w, and an equipartition X, Y of V_{d+1}. Consider ℓ different (w,d)-layered graphs $G_1, \ldots, G_\ell$ on these sets of vertices. The XOR product graph $H := \oplus_{i=1}^{\ell} G_i$ is the following graph (see also Figure 3):

- **Vertex-set:** Create vertex-sets $U_j^{r,i}$ for $r \in [\ell]$, $i \in [4]$, and $j \in [d+1]$, such that for every choice of r, i, $U_j^{r,i}$ is a copy of V_j. For any $v \in V_j$, $copy(v, r, i)$ denotes the copy of v in $U_j^{r,i}$. Additionally, we define $X^{r,i}, Y^{r,i}$ as copies of X and Y in $U_{d+1}^{r,i}$.
- **Edge-set:** The first part creates four identical copies of each G_r on $U^{r,i}$-vertices for $i \in [4]$. More formally, for any edge (u,v) in G_r, we connect $copy(u, r, i)$ to $copy(v, r, i)$ for all $i \in [4]$.
  The second part connects these separate graphs. For every $r \in [\ell]$, connect $X^{r,1}$ to $X^{r,3}$ and $X^{r,2}$ to $X^{r,4}$ using identity perfect matchings. Conversely, connect $Y^{r,1}$ to $Y^{r,4}$ and $Y^{r,2}$ to $Y^{r,3}$ using identity perfect matchings. Finally, for every $r \in [\ell-1]$, connect $U_1^{r,3}$ to $U_1^{r+1,1}$ and $U_1^{r,4}$ to $U_1^{r+1,2}$ using identity perfect matchings.

The outcome of this product is another layered graph with width $2w$ and depth $\ell \cdot (2 \cdot d + 1) + \ell - 1$. This concludes the description of the XOR product.

We now state the main property of this product. In the following, for an instance G of the **PC** problem, we write $\mathbf{PC}(G) \in \{0,1\}$ to

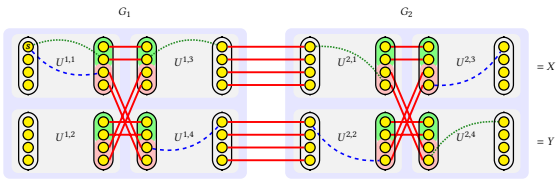

Figure 3: An illustration of the XOR product H of two graphs G_1, G_2. Here, the $X^{r,i}, Y^{r,i}$ sets are specified for each graph – these sets are entirely unrelated to equipartition X, Y of H drawn on the right. Moreover two potential paths out of s are drawn: (i) the dotted (green) one corresponds to $\mathbf{PC}(G_1) = 0, \mathbf{PC}(G_2) = 1$ and so $\mathbf{PC}(G_1 \oplus G_2) = \mathbf{PC}(G_1) \oplus \mathbf{PC}(G_2) = 1$ which is true as s reaches Y in H; (ii) the dashed (blue) one corresponds to $\mathbf{PC}(G_1) = \mathbf{PC}(G_2) = 1$ and so $\mathbf{PC}(G_1 \oplus G_2) = \mathbf{PC}(G_1) \oplus \mathbf{PC}(G_2) = 0$ which is true as s reaches X in H.

denote the answer of **PC** on G which is 0 if G is a X-instance and is 1 if G is a Y-instance. Suppose we have ℓ different instances of $\mathbf{PC}_{\hat{m},\hat{b}}$ as graphs $G_1, \ldots, G_\ell$ on the same set of vertices (and same s_r, X_r, Y_r across all $r \in [\ell]$). Consider the (m, b)-layered graph $H := \oplus_{r=1}^{\ell} G_i$ and define the following $\mathbf{PC}_{m,b}$ instance over H. For

$$s = copy(v, 1, 1), \qquad X = U_1^{\ell,3}, \qquad Y = U_1^{\ell,4},$$

decide whether $P(s)$ (in H) belongs to X or Y (it is worth pointing out that the sets X, Y defined for H are completely unrelated to sets X_r, Y_r for $r \in [\ell]$). We now have the following claim (a "proof by picture" is illustrated in Figure 3).

Claim 5.5. *For any integer ℓ and $H := \oplus_{r=1}^{\ell} G_r$, we have*

$$\mathbf{PC}_{m,b}(H) = \oplus_{r=1}^{\ell} \mathbf{PC}_{\hat{m},\hat{b}}(G_r)$$

Proof. Recall that in any layered graph, each vertex is part of a unique path from the first layer to the last one; for any $v \in H$, we denote this path by $\mathcal{P}(v)$. Consider any graph G_r for $r \in [\ell]$ and its starting vertex s_r. Notice that there are four copies of s_r across subgraphs $U^{r,i}$ for $i \in [4]$. Let us denote these copies by $s^{r,i} = copy(s, r, i)$ for $i \in [4]$.

If $\mathbf{PC}(G_r) = 0$, then $P(s_r) \in X_r$ by definition. We claim that in this case, $\mathcal{P}(s^{r,1})$ contains $\mathcal{P}(s^{r,3})$ because of the following path:

$$\begin{aligned} s^{r,1} =& copy(s_r, r, 1) \rightsquigarrow copy(P(s_r), r, 1) \in X^{r,1} \\ & \to copy(P(s_r), r, 3) \in X^{r,3} \rightsquigarrow copy(s_r, r, 3) = s^{r,3} \end{aligned}$$

the first path exists because $U_1^{r,1}$ to $U_{\hat{b}}^{r,1}$ in H are connected the same as G_r, the edge exists by the definition of H, and again $U_{\hat{b}}^{r,3}$ goes to $U_1^{r,3}$ the same as G_r. By the same exact reason, $\mathcal{P}(s^{r,2})$ contains $\mathcal{P}(s^{r,4})$.

Conversely, if $\mathbf{PC}(G_r) = 1$, then $P(s_r) \in Y_r$ by definition. We claim that in this case, $\mathcal{P}(s^{r,1})$ contains $\mathcal{P}(s^{r,4})$ instead because of the following path in H:

$$\begin{aligned} s^{r,1} =& copy(s_r, r, 1) \rightsquigarrow copy(P(s_r), r, 1) \in Y^{r,1} \\ & \to copy(P(s_r), r, 4) \in Y^{r,4} \rightsquigarrow copy(s_r, r, 4) = s^{r,4} \end{aligned}$$

this is exactly as before except for the fact that $Y^{r,1}$ is instead connected to $Y^{r,4}$ by a perfect matching. Again, we also have that $\mathcal{P}(s^{r,2})$ contains $\mathcal{P}(s^{r,3})$ in this case.

Now, consider the path $\mathcal{P}(s)$ in H. Recall that $s = s^{1,1}$ by definition. By the above argument, the path $\mathcal{P}(s^{1,1})$ then either contains $s^{1,3}$ in $U_1^{1,3}$ or $s^{1,4}$ in $U_1^{1,4}$. In the first case, $\mathcal{P}(s^{1,1})$ will then contain $\mathcal{P}(s^{2,1})$ and in the second case, it will next contain $\mathcal{P}(s^{2,2})$ by the construction of the last set of edges added to H in its definition. Continuing this inductively from $s^{2,1}$ and $s^{2,2}$ and their corresponding paths $\mathcal{P}(s^{2,1})$ and $\mathcal{P}(s^{2,2})$, we get that $\mathcal{P}(s)$ goes through a collection of vertices s^{r,i_r} for *every* $r \in [\ell]$ and *exactly one* $i_r \in [2]$ until it eventually ends up at either $s^{\ell,3} \in X$ or $s^{\ell,4} \in Y$ (which, we can think of as $s^{\ell+1,1}$ and $s^{\ell+1,2}$, respectively, for the ease of notation).

Next, consider any pair s^{r,i_r} and $s^{r+1,i_{r+1}}$ on $\mathcal{P}(s)$. By the argument above, if $\mathbf{PC}(G_r) = 0$, then $i_{r+1} = i_r$ while if $\mathbf{PC}(G_r) = 1$, then $i_{r+1} = 3 - i_r$, i.e., it "flips". Thus, starting from $s = s^{1,i_1} = s^{1,1}$, $\mathcal{P}(s)$ ends up at $s^{\ell+1,1} = s^{\ell,3} \in X$ if the number of flips is even and at $s^{\ell+1,2} = s^{\ell,4} \in Y$ if it is odd. This means that $\mathbf{PC}_{m,b}(H) = \oplus_{r=1}^{\ell} \mathbf{PC}_{\hat{m},\hat{b}}(G_r)$, proving the claim. ∎

Equipped with the XOR product and Claim 5.5, we can now prove Lemma 5.4.

Proof of Lemma 5.4. Consider ℓ *independent* instances $G_1, \ldots, G_\ell$ of $\mathbf{PC}_{\hat{m},\hat{b}}$ sampled from ${\mu_{\mathbf{PC}}}^{\ell}$. Define $H := \oplus_{r=1}^{\ell} G_r$, which is a (m, b)-layered graph for $m = 2\hat{m}$ and $b = \ell \cdot (2 \cdot \hat{b} + 1) + \ell - 1$ (these parameters match those of Lemma 5.4). By Claim 5.5, we have $\mathbf{PC}(H) = \oplus_{r=1}^{\ell} \mathbf{PC}(G_r)$.

We can now apply (the contrapositive of) our streaming XOR Lemma (Result 3) to obtain: if we have a p-pass s-space streaming algorithm for **PC** over the distribution of $H = \oplus_{r=1}^{\ell} G_r$ over ${\mu_{\mathbf{PC}}}^{\ell}$ with probability of success $1/2 + \delta$, then we will also have a p-pass s-space streaming algorithm for $\mathbf{PC}_{\hat{m},\hat{b}}$ over $\mu_{\mathbf{PC}}$ with $1/2 \cdot (1 + (2\delta)^{1/\ell})$ (by re-parameterizing δ to match that of Result 3).

We are still however not done because the algorithm A in the lemma works on the distribution $\mu_{\mathbf{PC}}$ while the distribution H induced by ${\mu_{\mathbf{PC}}}^{\ell}$ does not match $\mu_{\mathbf{PC}}$ due to the reduction. However, this is easy to fix. Pick random permutations $\pi_1, \ldots, \pi_{b+1}$ and use π_i to relabel vertices of layer V_i of the layered graph H to obtain a graph G. The distribution of G is now a random (m, b)-layered graph and thus we can run A over this graph for checking if $\pi_1(s)$ is in $\pi_{b+1}(X)$ or $\pi_{b+1}(Y)$ instead and obtain the answer to H as well. An averaging argument for fixing a choice of $\pi_1, \ldots, \pi_{b+1}$ finalizes the proof. ∎

5.3 Step Three: A Lower Bound for the Single-Copy Problem

The previous step allows us to instead of proving a lower bound for XOR of ℓ copies of the problem, prove a weaker lower bound for a single copy, which translates to a "standard" lower bound for pointer chasing. Our goal in this step is to prove this weaker lower bound. We state the following lemma in this section; the proof is included in the full version of the paper.

Lemma 5.6. *Let A be a p-pass s-space streaming algorithm for $\mathbf{PC}_{\hat{m},\hat{b}}$ over μ_{PC} with probability of success at least $\frac{1}{2}+\frac{1}{10\hat{m}^{1/\ell}}$. Then, either $p > \hat{b}-1$ or $s = \Omega(\frac{1}{\hat{b}^5}\cdot\hat{m}^{1-4/\ell})$.*

The proof of this lemma is similar to the known communication complexity lower bounds for pointer chasing such as [31, 56, 59, 67]; the catch however is that these lower bounds are for the distribution wherein each vertex in a layer V_i *independently* samples a neighbor in V_{i+1} (vertices of V_{i+1} can receive more than one edge) as opposed to a random matching. This independence between the choice of vertices is crucial in these lower bounds but at the same time working with such a distribution breaks multiple of our reductions (there are other minor differences as well, for instance, we will consider the lower bound directly for streaming algorithms to obtain (slightly) improved bounds but this is similar to [31]). As such, this final step of our proof is to show a new lower bound for the pointer chasing over the desired distribution, following the proofs of [31, 67] with some key modifications, in particular to allow for handling random matchings. We prove the following proposition, which implies Lemma 5.6.

Proposition 5.7. *Consider a p-pass s-space streaming algorithm for $\mathbf{PC}_{m,b}$ over random (m,b)-layered graphs with matchings $M_1,\ldots,M_b$ given in the stream $M_b \,||\, \cdots \,||\, M_1$. For $\gamma \in (0,1/2)$, if the algorithm succeeds with probability at least $1/2+\gamma$ then either $p > b-1$ or $s = \Omega\left(\frac{\gamma^4}{b^5}\cdot m\right)$.*

This is a good place to point out concretely why we need step two of our approach in Lemma 5.4, instead of simply applying Proposition 5.7 directly to our original **PC** problem in the reduction of Lemma 5.1. This is because the best advantage over random guessing i.e., parameter γ, this lower bound can provide is $\ll (\frac{1}{m})^{1/4}$ to give any meaningful space bound. Indeed, none of the other pointer chasing lower bounds such as [31, 56, 59, 67] can provide any meaningful guarantees when $\gamma \approx \frac{1}{m}$ while we need an almost-linear lower bound for $\gamma < \frac{1}{m}$ to apply our reduction in Lemma 5.1. As such, the hardness amplification step of Lemma 5.4 is the crucial step in our approach.

We postpone the proof of Proposition 5.7 to the full version and instead show how Lemma 5.6 follows immediately from this.

Proof of Lemma 5.6. Let $m = \hat{m}$, $b = \hat{b}$, and $\gamma = \frac{1}{10\hat{m}^{1/\ell}}$. The distribution μ_{PC} is the same as the hard distribution of Proposition 5.7. As such, if A solves $\mathbf{PC}_{\hat{m},\hat{b}}$ with probability of success at least $\frac{1}{2}+\frac{1}{10\hat{m}^{1/\ell}} = \frac{1}{2}+\gamma$, then, by Proposition 5.7, either $p > \hat{b}-1$ or

$$s = \Omega(\frac{\gamma^4}{\hat{b}^5}\cdot\hat{m}) = \Omega(\frac{1}{\hat{b}^5}\cdot\hat{m}^{1-4/\ell}),$$

concluding the proof. ∎

5.4 Putting Everything Together: Proof of Theorem 4.1

We are now ready to prove Theorem 4.1.

Proof of Theorem 4.1. Let A be a p-pass s-space streaming algorithm for $\mathbf{NGC}_{n,k}$ with probability of success at least $2/3$ over the distribution μ_{PC}. Let us go over each of the three steps in our approach below.

- **Step one:** By Lemma 5.1, existence of A implies a p-pass s-space streaming algorithm B for $\mathbf{PC}_{m,b}$ on μ_{PC} for *even* $m := \Theta(n/k)$ and $b := \frac{k-2}{2}$ with probability of success at least $\frac{1}{2}+\frac{1}{6m}$.
- **Step two:** Pick $\ell := \frac{b-1}{2p+4}$. By Lemma 5.1, existence of B implies a p-pass s-space streaming algorithm C for $\mathbf{PC}_{\hat{m},\hat{b}}$ on μ_{PC} for $\hat{m} = \frac{m}{2}$ and $\hat{b} = \frac{b-1}{2\ell} - 1 = p+1$ with probability of success at least $\frac{1}{2}\cdot\left(1+(\frac{2}{6m})^{1/\ell}\right) \geq \frac{1}{2}+\frac{1}{10\hat{m}^{1/\ell}}$.

  Note that m is even by the guarantee of previous part and $b-1$ divides 2ℓ by the choice of ℓ so we can indeed apply Lemma 5.1 in this step.
- **Step three:** By Lemma 5.6, considering C is a p-pass s-space streaming algorithm for $\mathbf{PC}_{\hat{m},\hat{b}}$ with probability of success at least $\frac{1}{2}+\frac{1}{10\hat{m}^{1/\ell}}$ and $p = \hat{b}-1$, we have that $s = \Omega(\frac{1}{\hat{b}^5}\cdot\hat{m}^{1-4/\ell})$.

We can now retrace these parameters to the original parameters n,k of $\mathbf{NGC}_{n,k}$. Firstly, $\hat{m} = \Theta(m) = \Theta(n/k)$ and $\hat{b} = p+1$. Secondly,

$$\ell = \frac{b-1}{2p+4} = \frac{(k-2)/2-1}{2p+4} = \frac{k-4}{4p+8}.$$

As such, the lower bound on the space complexity of all algorithms A, B and C above translates to

$$s = \Omega\left(\frac{1}{p^5}\cdot(n/k)^{1-\frac{4p+8}{k-4}}\right) = \Omega\left(\frac{1}{p^5}\cdot(n/k)^{1-O(p/k)}\right).$$

This proves Theorem 4.1 for infinitely many values of $k \in \mathbb{N}^+$, i.e., the ones where $\frac{k-4}{4p+8}$ is an integer.

We can extend this lower bound to all values of k by replacing the identity perfect matching between the sets S^1 to S^3, and S^2 to S^4 by a longer path of appropriate length.

This finalizes the proof of Theorem 4.1. ∎

6 STREAMING XOR LEMMA

We now give a proof of our streaming XOR lemma, which stands as an independent hardness amplification result in its own right.

Recall the intuition at the beginning of Section 2.3 behind any form of XOR Lemma: taking XOR of *independent* bits dampens their biases exponentially and thus the algorithm for $f^{\oplus\ell}$ that computes each $f(\sigma_i)$ individually satisfies Result 3. In general however, we cannot expect the algorithm to approach these subproblems independently as it may instead try to *correlate* its success probabilities across different subproblems (say, with probability $1/2+\delta$ all subproblems are correct and with the remaining probability, all are wrong). This is the main barrier in proving any form of XOR Lemma and what need to overcome in proving Result 3.

The main ideas of the proof consist of the following steps: (a) set up a ℓ-player "communication game", with one player per σ_i, whose lower bounds also imply lower bounds for streaming algorithms of $f^{\oplus\ell}$, (b) give enough extra power to this game so that no player is responsible for compressing the input of another player, and show that the players success in computing each $f(\sigma_i)$ becomes

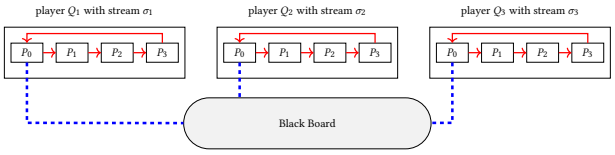

Figure 4: An illustration of the communication game in the proof of Result 3 for $n = 4$ and $\ell = 3$. The solid (red) lines draw the messages of players of each inner streaming algorithm, while dashed (blue) lines draw the communicated messages between the players of the game and the blackboard (from players P_0 of each inner streaming algorithm).

uncorrelated, (c) limit the power of the game so that streaming lower bounds for f also imply lower bounds for computing each $f(\sigma_i)$ in this game. We now formalize this in the rest of this section.

We setup the following game for proving this theorem (see also Figure 4):

(1) There are a total of ℓ players $Q_1, \ldots, Q_\ell$ who receive input streams $\sigma_1, \ldots, \sigma_\ell$, respectively.
(2) The players communicate with each other in *rounds* via a *blackboard*. In each round, the players go in turn with Q_1 writing a message on the board, followed by Q_2, all the way to Q_ℓ; these messages are visible to everyone (and are not altered or erased after written).
(3) For any player Q_i and round j, we use M_i^j to denote the message written on the board by Q_i in j-th round. We additionally use B_i^j to denote the content of the board *before* the message M_i^j is written and B^j to denote the content of the board *after* round j.
(4) Messages of each Q_i is generated by a *deterministic multi-pass streaming* algorithm $\mathcal{A}_i$ that runs on σ_i (with one inner player per element of the stream as in Definition 3.1). In each round j, the player P_0 of $\mathcal{A}_i$ is additionally given the content of the board B_i^j, then $\mathcal{A}_i$ makes its j-th pass over σ_i, and then P_0 of $\mathcal{A}_i$ outputs M_i^j on the board.
(5) The *cost* of a protocol is the maximum size of the memory of any algorithm $\mathcal{A}_i$.

Let us emphasize that this game is not at all a standard communication complexity problem: in our game, the communication between the players is *unbounded* and the cost of the algorithm is instead governed by the memory of streaming algorithms run by each player as opposed to having computationally unbounded players.

We first show that if we can lower bound the cost of protocols in this game, we immediately get a lower bound for streaming algorithms of $f^{\oplus \ell}$.

Lemma 6.1. *Any p-pass s-space algorithm A (deterministic or randomized) for computing $f^{\oplus \ell}$ implies a (deterministic) p-round protocol π with cost at most s and the same probability of success.*

Proof. Without loss of generality, we can assume A is deterministic as by an averaging argument, there is a fixing of the randomness of the algorithm that gives the same success probability over μ^ℓ.

To avoid confusion, let us denote the players of A as $R_0, R_1, \ldots, \ldots, R_{n\cdot\ell}$. We can generate a protocol π from A as follows:

(1) Q_1 runs A as $\mathcal{A}_1$ on σ_1: P_0 (of $\mathcal{A}_1$) sends a message to P_1 and so on until P_n by running the first pass of A over their inputs by simulating R_0 to R_n (this incurs a cost of s).
(2) At this point, P_n has the same input and message as player R_n of A. Thus, P_n can send the message of R_n to R_{n+1} instead to P_0 which finishes the first pass of $\mathcal{A}_1$ (again by a cost of only s as the message of R_n to R_{n+1} has size s). P_0 of $\mathcal{A}_1$ then can post this received message on the blackboard as message M_1^1 of player Q_1.
(3) Now it is Q_2's turn to run A as $\mathcal{A}_2$ on σ_2: P_0 (of $\mathcal{A}_2$) reads the content of the board and pass it along to P_1; this way, P_1 to P_n can continue the first pass of A over their inputs by simulating R_{n+1} to R_{2n}. Then, like step (ii), the message of R_{2n} to R_{2n+1} will be posted on the board via P_0 of $\mathcal{A}_2$.
(4) The players continue like this until they run every p passes of A in p rounds over their inputs and output the same answer.

As the cost of this protocol is s and it fully simulates A, we obtain the result. ∎

Fix a p-round communication protocol π with cost at most s in this game and suppose the inputs of players are sampled from the product distribution μ^ℓ. For the rest of the proof, we bound the probability of success of π which will imply Result 3 by Lemma 6.1.

To continue we need the following definitions:

- For any $i \in [\ell]$ and any choice of the final board content $\mathrm{B}^p = B$, we define $\mathrm{bias}_\pi(i, B)$ as equal to
$$2 \cdot \max_{\theta \in \{0,1\}} \Pr_{(\sigma_1,\ldots,\sigma_\ell)\sim\mu^\ell} \left(f(\sigma_i) = \theta \mid \mathrm{B}^p = B\right) - 1;$$
i.e., $\mathrm{bias}_\pi(i, B)$ equals bias $(f(\sigma_i))$ for $\sigma_i \sim \mu^\ell \mid \mathrm{B}^p = B$.
- For any choice of the final board content $\mathrm{B}^p = B$, we define $\mathrm{bias}_\pi(B)$ as equal to
$$2 \cdot \max_{\theta \in \{0,1\}} \Pr_{(\sigma_1,\ldots,\sigma_\ell)\sim\mu^\ell} \left(f^{\oplus\ell}(\sigma_1, \ldots, \sigma_\ell) = \theta \mid \mathrm{B}^p = B\right) - 1;$$
i.e., $\mathrm{bias}_\pi(B)$ equals bias $(f(\sigma_1) \oplus \cdots \oplus f(\sigma_\ell))$ for $\sigma_1, \ldots, \sigma_\ell \sim \mu^\ell \mid \mathrm{B}^p = B$.

With these definitions, we have,

$$\Pr_{(\sigma_1,\ldots,\sigma_\ell)\sim\mu^\ell} (\pi \text{ is correct}) \tag{4}$$
$$= \mathop{\mathbb{E}}_{B} \left[\Pr_{(\sigma_1,\ldots,\sigma_\ell)\sim\mu^\ell} \left(\pi \text{ is correct} \mid \mathrm{B}^p = B\right) \right]$$
$$\leq \mathop{\mathbb{E}}_{B} \left[\max_{\theta \in \{0,1\}} \Pr_{(\sigma_1,\ldots,\sigma_\ell)\sim\mu^\ell} \left(f^{\oplus\ell}(\sigma_1, \ldots, \sigma_\ell) = \theta \mid \mathrm{B}^p = B\right) \right]$$
(conditioned on $\mathrm{B}^p = B$, the answer of π is fixed to some $\theta \in \{0, 1\}$)
$$= \mathop{\mathbb{E}}_{B} \left[\frac{1}{2} \cdot (1 + \mathrm{bias}_\pi(B)) \right] = \frac{1}{2} + \frac{1}{2} \cdot \mathop{\mathbb{E}}_{B} \left[\mathrm{bias}_\pi(B) \right]. \tag{5}$$

As such, $\mathbb{E}_B[\text{bias}_\pi(B)]$ measures the advantage of π in outputting the answer over random guessing. Our goal in the remainder of this section is to bound this expectation. In order to do so, we first bound each $\mathbb{E}_B[\text{bias}_\pi(i,B)]$ for $i \in [\ell]$, and then prove a crucial independence property between these variables that allows us to extend these bounds appropriately to $\mathbb{E}_B[\text{bias}_\pi(B)]$ as well.

In the following, we prove that the protocol π is not able to change the bias of any single $f(\sigma_i)$ by more than 2δ, or alternatively, it cannot "solve" $f(\sigma_i)$ correctly with probability $> 1/2 + \delta$. Intuitively, this should be true as π is effectively running a p-pass s-space streaming algorithm $\mathcal{A}_i$ on σ_i and so we can apply the assumption of Result 3. The catch is that π in general is stronger than a streaming algorithm (which is necessary to establish the other parts of the lower bound) and some additional care is needed to simulate π "projected" on σ_i via a streaming algorithm.

Lemma 6.2. *For any $i \in [\ell]$, $\mathbb{E}_B[\text{bias}_\pi(i,B)] \le 2\delta$.*

Proof. To prove this lemma, we only need to turn π into a p-pass s-space streaming algorithm A for computing $f(\sigma_i)$ on the stream $\sigma_i \sim \mu$ (and not the entire input); the rest follows directly from the assumption of Result 3 on the success probability of streaming algorithms on μ.

Suppose by way of contradiction that $\mathbb{E}_B[\text{bias}_\pi(i,B)] > 2\delta$. Consider the estimator

$$g(B) := \arg\max_{\theta \in \{0,1\}} \Pr_{\mu^\ell}\left(f(\sigma_i) = \theta \mid B^p = B\right).$$

Then, by the definition of $\text{bias}_\pi(i,B)$, we have $\mathbb{E}_B \Pr_{\sigma_i \sim \mu^\ell}(f(\sigma_i) = g(B) \mid B^p = B) > \frac{1}{2} + \delta$. Define $\sigma_{-i} = (\sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_\ell)$.

By an averaging argument, and since $\sigma_1, \ldots, \sigma_\ell$ are independent, there is a fixing of σ_{-i} to some σ^*_{-i} which results in

$$\Pr_{\sigma_i \sim \mu}\left(f(\sigma_i) = g(B^*)\right) > \frac{1}{2} + \delta, \tag{6}$$

where $B^* = B^*(\sigma^*_{-i}, \sigma_i)$ is a random variable for the final content of the board given $(\sigma^*_{-i}, \sigma_i)$ over the randomness of σ_i only. We now use this to design the streaming algorithm A (with σ^*_{-i} "hard coded" in the algorithm); it might be helpful to consult Figure 4 when reading this part.

Given the stream $\sigma \sim \mu$, A works as follows: P_0 of A will simulate running π on $\sigma^*_1, \ldots, \sigma^*_{i-1}$ to obtain B^1_i. This allows P_0 to start running $\mathcal{A}_i$ on $\sigma_i = \sigma$ and $P_0, \ldots, P_n$ can collectively run the first pass of $\mathcal{A}_i$ on σ_i; at the end, P_0 knows the message M^1_i of π and thus B^1_{i+1}; this allows P_0 to simulate π on $\sigma^*_{i+1}, \ldots, \sigma^*_\ell$ on its own and obtain B^1. This finishes one round of the protocol π over $(\sigma^*_1, \ldots, \sigma^*_{-1}, \sigma, \sigma^*_{i+1}, \ldots, \sigma^*_\ell)$, while the players of A only made one pass over σ and communicated s bits each (for running $\mathcal{A}_i$ in space s – note that here P_0 is solely responsible for simulating the blackboard and thus require no further communication).

The players then continue this to simulate all p rounds of π in p passes over the input σ and space of s bits. At the end, P_0 knows the entire content of the entire board B and can output $g(B)$ as the answer to $f(\sigma)$. Over the randomness of $\sigma \sim \mu$, the distribution of $(\sigma^*_1, \ldots, \sigma^*_{-1}, \sigma, \sigma^*_{i+1}, \ldots, \sigma^*_\ell)$ and B is the same as $(\sigma^*_{-i}, \sigma_i)$ and B^* in Equation (6). This means that A, which is a p-pass s-space streaming algorithm, outputs the correct answer to $f(\sigma)$ with probability $> 1/2 + \delta$ contradicting the assumption of Result 3. ∎

To extend the bounds in this lemma to $\text{bias}(B)$, we like to use the fact that XOR dampens the bias of *independent* bits.Thus, we need to establish that these $f(\sigma_i)$ bits are not correlated after conditioning on B, which is done in the following lemma. This can be seen as an analogue of the rectangle property of standard communication protocols on product distributions. We need the following standard facts from information theory for our proofs:

Fact 6.3. *$\mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}) \ge 0$. The equality holds iff A and B are* independent *conditioned on C.*

Fact 6.4. *For random variables $\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}$, if $\mathsf{A} \perp \mathsf{D} \mid \mathsf{C}$, then,*

$$\mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}) \le \mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}, \mathsf{D}).$$

Fact 6.5. *For random variables $\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}$, if $\mathsf{A} \perp \mathsf{D} \mid \mathsf{B}, \mathsf{C}$, then,*

$$\mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}) \ge \mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}, \mathsf{D}).$$

Lemma 6.6. *The input streams σ_i's are independent <u>even</u> conditioned on $\mathsf{B}^p = B$ i.e., for any B, $\left(\sigma_1, \ldots, \sigma_\ell \sim \mu^\ell \mid \mathsf{B}^p = B\right) = \bigtimes_{i=1}^{\ell}\left(\sigma_i \sim \mu^\ell \mid \mathsf{B}^p = B\right)$,.*

Proof. The input streams are originally independent, so we need to show that the protocol π in this game cannot correlate them after we condition on B.

Define the following random variables: X_i for the input σ_i of player i, and M^j_i, B^j_i, and B^j, for M^j_i, B^j_i, and B^j respectively. Our goal is to prove that $\mathsf{X}_1, \ldots, \mathsf{X}_\ell$ are independent conditioned on any choice of $\mathsf{B}^p = B$. To do this, we show that for any $i \in [\ell]$,

$$\mathbb{I}(\mathsf{X}_i\,;\mathsf{X}_{-i} \mid \mathsf{B}^p) = 0 \tag{7}$$

where $\mathsf{X}_{-i} = (\mathsf{X}_1, \ldots, \mathsf{X}_{i-1}, \mathsf{X}_{i+1}, \ldots, \mathsf{X}_\ell)$. By Fact 6.3, this implies that $\mathsf{X}_i \perp \mathsf{X}_{-i} \mid \mathsf{B}^p = B$ for any choice of B and $i \in [\ell]$, which in turn proves the lemma.

To this end, we peel off messages written on the board one by one from the conditioning of Equation (7) without ever increasing the mutual information term. Then, we end up with a case when there is no conditioning on any part of B and we can use the fact that X_i and X_{-i} are independent to finalize the proof. Formally,

$$\mathbb{I}(\mathsf{X}_i\,;\mathsf{X}_{-i} \mid \mathsf{B}^p_{i+1}, \mathsf{M}^p_{i+1}, \ldots, \mathsf{M}^p_\ell) \quad \le \mathbb{I}(\mathsf{X}_i\,;\mathsf{X}_{-i} \mid \mathsf{B}^p_{i+1}),$$

which holds by Fact 6.5 because $\mathsf{X}_i \perp \mathsf{M}^p_{i+1}, \ldots, \mathsf{M}^p_\ell \mid \mathsf{B}^p_{i+1}, \mathsf{X}_{-i}$ so dropping the conditioning can only increase the information. This independence itself is because the messages sent by players after i in the last round are deterministic functions of their inputs and the content of the board after player i speaks, namely, B^p_{i+1}, and thus in the above term, $(\mathsf{M}^p_{i+1}, \ldots, \mathsf{M}^p_\ell)$ is deterministically fixed after conditioning on $\mathsf{B}^p_{i+1}, \mathsf{X}_{-i}$. We can further write,

$$\mathbb{I}(\mathsf{X}_i\,;\mathsf{X}_{-i} \mid \mathsf{B}^p_{i+1}) = \mathbb{I}(\mathsf{X}_i\,;\mathsf{X}_{-i} \mid \mathsf{B}^p_i, \mathsf{M}^p_i) \quad \le \mathbb{I}(\mathsf{X}_i\,;\mathsf{X}_{-i} \mid \mathsf{B}^p_i),$$

which again holds by Fact 6.5 because $\mathsf{X}_{-i} \perp \mathsf{M}^p_i \mid \mathsf{B}^p_i, \mathsf{X}_i$ as M^p_i is a deterministic function of B^p_i and X_i. Finally,

$$\mathbb{I}(\mathsf{X}_i\,;\mathsf{X}_{-i} \mid \mathsf{B}^{p-1}, \mathsf{M}^p_1, \ldots, \mathsf{M}^p_{i-1}) \quad \le \mathbb{I}(\mathsf{X}_i\,;\mathsf{X}_{-i} \mid \mathsf{B}^{p-1}),$$

by Fact 6.5, exactly as in the first part above because $X_i \perp M_1^p, \ldots, M_{i-1}^p \mid B^{p-1}, X_{-i}$, as conditioning on B^{p-1}, X_{-i} fixes the last messages sent by players 1 to $i-1$.

This way, we can shave off one entire round of communication from the conditioning in the LHS of Equation (7). Applying this argument for all p rounds, we have that,

$$\mathbb{I}(X_i ; X_{-i} \mid B^p) \leq \mathbb{I}(X_i ; X_{-i} \mid B^{p-1}) \leq \cdots \leq \mathbb{I}(X_i ; X_{-i} \mid B^0)$$

where the last term is zero because $\mathbb{I}(X_i ; X_{-i} \mid B^0) = \mathbb{I}(X_i ; X_{-i})$ and $X_i \perp X_{-i}$ in the distribution μ^ℓ thus we can apply Fact 6.3. This proves Equation (7) and concludes the proof. ∎

Finally, we use Lemmas 6.2 and 6.6 with Equation (5) to bound the success probability of protocol π.

Lemma 6.7. *Protocol π succeeds with probability $\leq \frac{1}{2} \cdot \left(1 + (2\delta)^\ell\right)$.*

Proof. We will prove that $\mathbb{E}_B [\text{bias}(B)] \leq (2\delta)^\ell$ which implies the lemma by Equation (5). Fix any B and consider the random variables $f(\sigma_1), \ldots, f(\sigma_\ell)$ for $(\sigma_1, \ldots, \sigma_\ell) \sim \mu^\ell \mid B^p = B$. By Lemma 6.6, even in the distribution $\mu^\ell \mid B^p = B$, σ_i's are independent which implies that $f(\sigma_1), \ldots, f(\sigma_\ell)$ are also independent random variables conditioned on B. As such, for any B,

$$\text{bias}_\pi(B) = \prod_{i=1}^{\ell} \text{bias}(f(\sigma_i) \mid B^p = B) = \prod_{i=1}^{\ell} \text{bias}_\pi(i, B),$$

where the first and last equalities are by the definitions of $\text{bias}_\pi(B)$ and $\text{bias}_\pi(i, B)$, and the middle equality is by the independence of $f(\sigma_i)$'s conditioned on B, and the fact that XOR dampens the biases of independent random bits. Finally,

$$\mathop{\mathbb{E}}_B [\text{bias}_\pi(B)] = \mathop{\mathbb{E}}_B \left[\prod_{i=1}^{\ell} \text{bias}_\pi(i, B) \right] = \prod_{i=1}^{\ell} \mathop{\mathbb{E}}_b [\text{bias}_\pi(i, B)] \leq (2\delta)^\ell,$$

where the last equality is by the independence of $\text{bias}_\pi(i, B)$ and the inequality by Lemma 6.2. ∎

Result 3 now follows immediately from Lemmas 6.1 and 6.7.

7 OTHER STREAMING LOWER BOUNDS

We state the lower bounds we obtain via simple reductions from **NGC** using the lower bound in Theorem 4.1 in this section. Proofs in this section appear in the full version of the paper.

Minimum Spanning Tree. Given an undirected graph $G = (V, E)$, with edge-weights $w : E \to \{1, 2, \ldots, W\}$, the minimum spanning tree problem asks for an estimate to the weight of a spanning tree in G with the least weight, denoted by MST of G. We prove the following lower bound for this problem:

Theorem 7.1. *For $\varepsilon \in (0, 1)$ and $W \in \mathbb{N}^+$, any p-pass streaming algorithm for $(1+\varepsilon)$-approximation of weight of MST on n-vertex graphs of maximum weight W with probability at least $2/3$ requires $\Omega\left(\frac{1}{p^5} \cdot (\varepsilon \cdot n/W)^{1-O(\varepsilon \cdot p/W)}\right)$ space. This lower bound continues to hold even on bounded-degree planar graphs and also implies that $\Omega(1/\varepsilon)$ passes are needed for $n^{o(1)}$-space even for $W = O(1)$.*

Maximum Matching Size and Matrix Rank. In the maximum matching size problem, our goal is to output an estimate to the *size* of the maximum matching of the input undirected graph $G(V, E)$, i.e. the largest set of vertex-disjoint edges in G. We prove the following lower bound for this problem:

Theorem 7.2. *For any $\varepsilon \in (0, 1)$, any p-pass streaming algorithm for $(1+\varepsilon)$-approximation of size of maximum matching on n-vertex graphs with probability at least $2/3$ requires $\Omega\left(\frac{1}{p^5} \cdot (\varepsilon \cdot n)^{1-O(\varepsilon \cdot p)}\right)$ space. Moreover, this lower bound continues to hold even on bounded-degree planar graphs and also implies that $\Omega(1/\varepsilon)$ passes are needed for any $n^{o(1)}$-space algorithm.*

As a consequence of the standard equivalence between estimating matching size and computing the rank of the Tutte matrix [63] with entries chosen randomly established in [50].

Corollary 7.3. *For any $\varepsilon \in (0, 1)$, any p-pass streaming algorithm for $(1+\varepsilon)$-approximation of rank n-by-n matrices with probability at least $2/3$ requires $\Omega\left(\frac{1}{p^5} \cdot (\varepsilon \cdot n)^{1-O(\varepsilon \cdot p)}\right)$ space. Moreover, this lower bound continues to hold even on matrices with $O(1)$ entries per row and column and also implies that $\Omega(1/\varepsilon)$ passes are needed for any $n^{o(1)}$-space algorithm.*

Maximum Cut. In the maximum cut problem, our goal is to estimate the largest *value* of a cut in an input graph $G(V, E)$ i.e. output an estimate of the size of a bi-partition of vertices maximizing the number of crossing edges. We prove the following lower bound for this problem.

Theorem 7.4. *For any $\varepsilon \in (0, 1)$, any p-pass streaming algorithm for $(1+\varepsilon)$-approximation of value of maximum cut on n-vertex graphs with probability at least $2/3$ requires $\Omega\left(\frac{1}{p^5} \cdot (\varepsilon \cdot n)^{1-O(\varepsilon \cdot p)}\right)$ space. Moreover, this lower bound continues to hold even on bounded-degree planar graphs and also implies that $\Omega(1/\varepsilon)$ passes are needed for any $n^{o(1)}$-space algorithm.*

Maximum Acyclic Subgraph. Given a directed graph $G(V, E)$, the maximum acyclic subgraph problem asks for an estimate to the *size* of the largest acyclic subgraph in G measured in the number of edges. We prove the following lower bound for this problem:

Theorem 7.5. *For any $\varepsilon \in (0, 1)$, any p-pass streaming algorithm for $(1+\varepsilon)$-approximation of size of a largest acyclic subgraph on n-vertex directed graphs with probability at least $2/3$ requires $\Omega\left(\frac{1}{p^5} \cdot (\varepsilon \cdot n)^{1-O(\varepsilon \cdot p)}\right)$ space. Moreover, this lower bound continues to hold even on bounded-degree planar graphs and also implies that $\Omega(1/\varepsilon)$ passes are needed for any $n^{o(1)}$-space algorithm.*

Property Testing: Connectivity, Bipartiteness, and Cycle-freeness. Given a graph property P and an $\varepsilon \in (0, 1)$, an ε-property tester for P is an algorithm that decides whether an input G has the property P or is ε-*far* from having P. We consider the following properties: **Connectivity:** If at least $\varepsilon \cdot n$ edges need to be inserted to G to make it connected, then G is said to be ε-far from being connected; **Bipartiteness:** If at least $\varepsilon \cdot n$ edges need to be deleted from G to make it bipartite, then G is said to be ε-far from being bipartite;

Cycle-freeness: If at least $\varepsilon \cdot n$ edges need to be deleted from G to remove all its cycles, then G is said to be ε-far from being cycle-free.

We prove the following lower bound for these problems:

Theorem 7.6. *For any $\varepsilon \in (0,1)$, any p-pass streaming algorithm which is a ε-property tester for connectivity, bipartiteness, and cycle-freeness on n-vertex graphs with probability at least $2/3$ requires $\Omega\left(\frac{1}{p^5} \cdot (\varepsilon \cdot n)^{1-O(\varepsilon \cdot p)}\right)$ space. Moreover, this lower bound continues to hold even on bounded-degree planar graphs and also implies that $\Omega(1/\varepsilon)$ passes are needed for any $n^{o(1)}$-space algorithm.*

ACKNOWLEDGMENTS

Sepehr Assadi was supported in part by the NSF CAREER Grant CCF-2047061 and a gift from Google Research, and Vishvajeet N was supported in part by the NSF grant CCF-1814409.

REFERENCES

[1] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. 2019. Polynomial pass lower bounds for graph streaming algorithms. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*. 265–276.

[2] Sepehr Assadi, Sanjeev Khanna, and Yang Li. 2017. On Estimating Maximum Matching Size in Graph Streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*. 1723–1742.

[3] Sepehr Assadi, Gillat Kol, Raghuvansh R. Saxena, and Huacheng Yu. 2020. Multi-Pass Graph Streaming Lower Bounds for Cycle Counting, MAX-CUT, Matching Size, and Other Problems. *CoRR* abs/2009.03038. To appear in FOCS 2020 (2020).

[4] Mitali Bafna, Badih Ghazi, Noah Golowich, and Madhu Sudan. 2019. Communication-Rounds Tradeoffs for Common Randomness and Secret Key Generation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*. 1861–1871.

[5] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. 2002. An Information Statistics Approach to Data Stream and Communication Complexity. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Proceedings*. 209–218.

[6] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. 2002. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA*. 623–632.

[7] Suman K. Bera and Amit Chakrabarti. 2017. Towards Tighter Space Bounds for Counting Triangles and Other Substructures in Graph Streams. In *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*. 11:1–11:14.

[8] Aditya Bhaskara, Samira Daruki, and Suresh Venkatasubramanian. 2018. Sublinear Algorithms for MAXCUT and Correlation Clustering. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*. 16:1–16:14.

[9] Mark Braverman, Anup Rao, Omri Weinstein, and Amir Yehudayoff. 2013. Direct Products in Communication Complexity. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013*. 746–755.

[10] Vladimir Braverman, Stephen R. Chestnut, Robert Krauthgamer, Yi Li, David P. Woodruff, and Lin F. Yang. 2018. Matrix Norms in Data Streams: Faster, Multi-Pass and Row-Order. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. 648–657.

[11] Vladimir Braverman, Rafail Ostrovsky, and Dan Vilenchik. 2013. How Hard Is Counting Triangles in the Streaming Model?. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*. 244–254.

[12] Joshua Brody, Jae Tak Kim, Peem Lerdputtipongporn, and Hariharan Srinivasulu. 2020. A Strong XOR Lemma for Randomized Query Complexity. *CoRR* abs/2007.05580 (2020).

[13] Laurent Bulteau, Vincent Froese, Konstantin Kutzkov, and Rasmus Pagh. 2016. Triangle Counting in Dynamic Graph Streams. *Algorithmica* 76, 1 (2016), 259–278.

[14] Marc Bury and Chris Schwiegelshohn. 2015. Sublinear Estimation of Weighted Matchings in Dynamic Data Streams. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, September 14-16, 2015, Proceedings*. 263–274.

[15] Amit Chakrabarti, Graham Cormode, and Andrew McGregor. 2008. Robust lower bounds for communication and stream computation. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, May 17-20, 2008*. 641–650.

[16] Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. 2005. Approximating the Minimum Spanning Tree Weight in Sublinear Time. *SIAM J. Comput.* 34, 6 (2005), 1370–1379.

[17] Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. 2016. Kernelization via Sampling with Applications to Finding Matchings and Related Problems in Dynamic Graph Streams. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, January 10-12, 2016*. 1326–1344.

[18] Chi-Ning Chou, Alexander Golovnev, and Santhoshini Velusamy. 2020. Optimal Streaming Approximations for all Boolean Max-2CSPs. *CoRR* abs/2004.11796. To appear in FOCS 2020 (2020).

[19] Graham Cormode and Hossein Jowhari. 2017. A second look at counting triangles in graph streams (corrected). *Theor. Comput. Sci.* 683 (2017), 22–30.

[20] Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and S. Muthukrishnan. 2017. The Sparse Awakens: Streaming Algorithms for Matching Size Estimation in Sparse Graphs. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017*. 29:1–29:15.

[21] Artur Czumaj, Hendrik Fichtenberger, Pan Peng, and Christian Sohler. 2020. Testable Properties in General Graphs and Random Order Streaming. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*. 16:1–16:20.

[22] Hossein Esfandiari, Mohammad Taghi Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. 2015. Streaming Algorithms for Estimating the Matching Size in Planar Graphs and Beyond. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, January 4-6, 2015*. 1217–1233.

[23] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. 2005. On graph problems in a semi-streaming model. *Theor. Comput. Sci.* 348, 2-3 (2005), 207–216. https://doi.org/10.1016/j.tcs.2005.09.013

[24] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. 2008. Graph Distances in the Data-Stream Model. *SIAM J. Comput.* 38, 5 (2008), 1709–1727.

[25] Dmitry Gavinsky, Julia Kempe, Iordanis Kerenidis, Ran Raz, and Ronald de Wolf. 2007. Exponential separations for one-way quantum communication complexity, with applications to cryptography. *STOC* (2007), 516–525.

[26] Uma Girish, Ran Raz, and Wei Zhan. 2020. Lower Bounds for XOR of Forrelations. *CoRR* abs/2007.03631 (2020).

[27] Oded Goldreich, Noam Nisan, and Avi Wigderson. 2011. On Yao's XOR-Lemma. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*. 273–301.

[28] Noah Golowich and Madhu Sudan. 2020. Round Complexity of Common Randomness Generation: The Amortized Setting. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*. 1076–1095.

[29] Sudipto Guha and Zhiyi Huang. 2009. Revisiting the Direct Sum Theorem and Space Lower Bounds in Random Order Streams. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*. 513–524.

[30] Sudipto Guha and Andrew McGregor. 2008. Tight Lower Bounds for Multi-pass Stream Computation Via Pass Elimination. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*. 760–772.

[31] Sudipto Guha and Andrew McGregor. 2009. Stream Order and Order Statistics: Quantile Estimation in Random-Order Streams. *SIAM J. Comput.* 38, 5 (2009), 2044–2059.

[32] Venkatesan Guruswami and Krzysztof Onak. 2013. Superlinear Lower Bounds for Multipass Graph Processing. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*. 287–298.

[33] Venkatesan Guruswami and Runzhou Tao. 2019. Streaming Hardness of Unique Games. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*. 5:1–5:12.

[34] Venkatesan Guruswami, Ameya Velingker, and Santhoshini Velusamy. 2017. Streaming Complexity of Approximating Max 2CSP and Max Acyclic Subgraph. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*. 8:1–8:19.

[35] Zengfeng Huang and Pan Peng. 2016. Dynamic Graph Stream Algorithms in o(n) Space. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. 18:1–18:16.

[36] Russell Impagliazzo. 1995. Hard-Core Distributions for Somewhat Hard Problems. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*. 538–545.

[37] Russell Impagliazzo and Avi Wigderson. 1997. *P = BPP* if *E* Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*. 220–229.

[38] Rahul Jain, Attila Pereszlényi, and Penghui Yao. 2012. A Direct Product Theorem for the Two-Party Bounded-Round Public-Coin Communication Complexity. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, October 20-23, 2012*. 167–176.

[39] Rahul Jain, Jaikumar Radhakrishnan, and Pranab Sen. 2003. A Direct Sum Theorem in Communication Complexity via Message Compression. In *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, June 30 - July 4, 2003. Proceedings*. 300–315.

[40] John Kallaugher, Michael Kapralov, and Eric Price. 2018. The Sketching Complexity of Graph and Hypergraph Counting. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*. 556–567.

[41] John Kallaugher, Andrew McGregor, Eric Price, and Sofya Vorotnikova. 2019. The Complexity of Counting Cycles in the Adjacency List Streaming Model. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*. 119–133.

[42] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. 2014. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*. 734–751.

[43] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. 2015. Streaming Lower Bounds for Approximating MAX-CUT. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*. 1263–1282.

[44] Michael Kapralov, Sanjeev Khanna, Madhu Sudan, and Ameya Velingker. 2017. $(1 + \Omega(1))$-Approximation to MAX-CUT Requires Linear Space. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*. 1703–1722.

[45] Michael Kapralov and Dmitry Krachun. 2019. An optimal space lower bound for approximating MAX-CUT. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*. 277–288.

[46] Michael Kapralov, Slobodan Mitrovic, Ashkan Norouzi-Fard, and Jakab Tardos. 2020. Space Efficient Approximation to Maximum Matching Size from Uniform Edge Samples. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*. 1753–1772.

[47] Dmitry Kogan and Robert Krauthgamer. 2015. Sketching Cuts in Graphs and Hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*. 367–376.

[48] Leonid A. Levin. 1985. One-Way Functions and Pseudorandom Generators. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*. 363–365.

[49] Yi Li and David P. Woodruff. 2016. On approximating functions of the singular values in a stream. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. 726–739.

[50] László Lovász. 1979. On determinants, matchings, and random algorithms. In *Fundamentals of Computation Theory, FCT 1979, Proceedings of the Conference on Algebraic, Arithmetic, and Categorial Methods in Computation Theory, Berlin/Wendisch-Rietz, Germany, September 17-21, 1979*. 565–574.

[51] Andrew McGregor and Sofya Vorotnikova. 2016. Planar Matching in Streams Revisited. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016*. 17:1–17:12.

[52] Andrew McGregor and Sofya Vorotnikova. 2018. A Simple, Space-Efficient, Streaming Algorithm for Matchings in Low Arboricity Graphs. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018*. 14:1–14:4.

[53] Andrew McGregor, Sofya Vorotnikova, and Hoa T. Vu. 2016. Better Algorithms for Counting Triangles in Data Streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*. 401–411.

[54] Marco Molinaro, David P. Woodruff, and Grigory Yaroslavtsev. 2013. Beating the Direct Sum Theorem in Communication Complexity with Implications for Sketching. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*. 1738–1756.

[55] Morteza Monemizadeh, S. Muthukrishnan, Pan Peng, and Christian Sohler. 2017. Testable Bounded Degree Graph Properties Are Random Order Streamable. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*. 131:1–131:14.

[56] Noam Nisan and Avi Wigderson. 1991. Rounds in Communication Complexity Revisited. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*. 419–429.

[57] Pan Peng and Christian Sohler. 2018. Estimating Graph Parameters from Random Order Streams. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. 2449–2466.

[58] Jeff M. Phillips, Elad Verbin, and Qin Zhang. 2012. Lower bounds for number-in-hand multiparty communication complexity, made easy. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*. 486–501.

[59] Stephen Ponzio, Jaikumar Radhakrishnan, and Srinivasan Venkatesh. 1999. The Communication Complexity of Pointer Chasing: Applications of Entropy and Sampling. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*. 602–611.

[60] Anup Rao and Makrand Sinha. 2016. A Direct-Sum Theorem for Read-Once Branching Programs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*. 44:1–44:15.

[61] Ronen Shaltiel. 2003. Towards proving strong direct product theorems. *Comput. Complex.* 12, 1-2 (2003), 1–22.

[62] Alexander A. Sherstov. 2011. Strong direct product theorems for quantum communication and query complexity. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*. 41–50.

[63] William T Tutte. 1947. The factorization of linear graphs. *Journal of the London Mathematical Society* 1, 2 (1947), 107–111.

[64] Elad Verbin and Wei Yu. 2011. The Streaming Complexity of Cycle Counting, Sorting by Reversals, and Other Problems. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, January 23-25, 2011*. 11–25.

[65] Emanuele Viola and Avi Wigderson. 2008. Norms, XOR Lemmas, and Lower Bounds for Polynomials and Protocols. *Theory Comput.* 4, 1 (2008), 137–168.

[66] Andrew Chi-Chih Yao. 1982. Theory and Applications of Trapdoor Functions (Extended Abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*. 80–91.

[67] Amir Yehudayoff. 2016. Pointer chasing via triangular discrimination. *Electronic Colloquium on Computational Complexity (ECCC)* 23 (2016), 151.