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

Sepehr Assadi*   Vishvajeet N†

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.

Contents

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., [KK15, KKS15, KKSV17, BDV18, KK19] on maximum cut, [AKL17, EHL$^+$15, CCE$^+$16, MV16, MV18, CJMM17, KKS14] on maximum matching size, [BKS02, BOV13, CJ17, MVV16, BC17, BFKP16, KMPV19] on subgraph counting, [GVV17, GT19, CGV20] on CSPs, and [HP16, MMPS17, PS18, CFPS20] on property testing, among others (see also [VY11, CFPS20, AKSY20] for a more detailed discussion of this line of work).

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 [HP16] 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 [AKSY20] 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 [CRT05] (see [PS18]).

Our goal in this paper is to make further progress on understanding the limits of multi-pass graph streaming algorithms for 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 [VY11, AKSY20], 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 then follow by easy reductions from this problem.

Our proof techniques are potentially useful for addressing other questions along these lines. We first use a "decorrelation" step to break the strong promise in the input graphs (that the cycles are either all short or all long) when proving the lower bound; this however comes at a cost of having to prove a lower bound for algorithms that succeed with a low probability of $1/2 + 1/\text{poly}(n)$. The main ingredient of the proof is then a hardness amplification step which allows us to obtain such a lower bound from any standard lower bound, i.e., a one with not-so-little probability of success. The key to this argument is a *streaming XOR Lemma*, in spirit of classical Yao's XOR Lemma [Yao82], that we prove in this paper. We elaborate on our results and techniques in details in the following.

1.1   Gap Cycle Counting With a Little Bit of "Noise"

Already a decade ago, Verbin and Yu [VY11] 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 [GKK$^+$07], 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. [EHL+15, BS15, KK15, LW16, HP16, GVV17, BCK+18, KKS15, AKL17, GT19, KKSV17, KK19, KKP18, CGV20]. These lower bounds were all for single-pass algorithms. Very recently, [AKSY20] 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 $\text{polylog}(n)$ space. The work of [AKSY20] 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 [VY11, AKSY20] 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. In particular, in our *noisy gap cycle counting* problem, 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 (see Definition 4.1 and Figure 1).

(a) The original 4-cycle vs 8-cycle problem.   (b) The noisy 4-cycle vs 8-cycle problem.
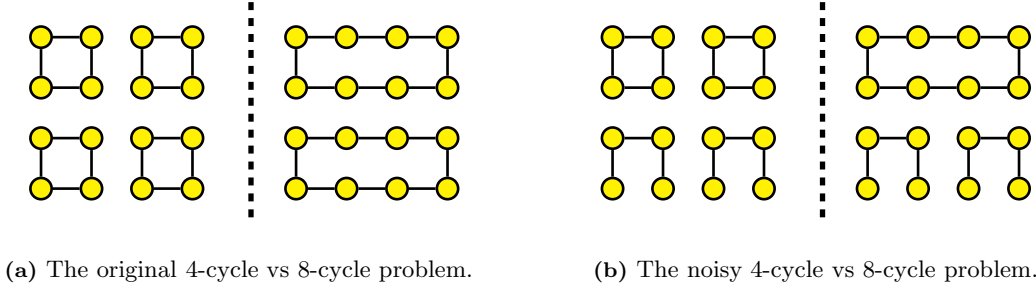
Figure 1: An illustration of the graphs in the original gap cycle counting problem for $k = 4$ versus the graphs in the new noisy gap cycle counting problem. The actual graph consists of $\Theta(n/k)$ copies of these smaller subgraphs.

We prove the following lower bound for this noisy gap cycle counting problem.

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 (see Appendix C). 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 [AKSY20] (see also Appendix D for a brief technical comparison of our work with that of [AKSY20]).

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] [VY11, 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, Result 1 settles a qualitatively similar form of this conjecture which allows for an $n^{1-O(p/k)}$-space p-pass tradeoff.

---

[1]The conjecture of [VY11] 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.

1.2 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; see Corollary 4.13). As a result, we can recover *all* graph streaming lower bounds of [AKSY20] 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:*
>
> – *$(1+\varepsilon)$-approximation of maximum matching size, maximum cut value, maximum acyclic subgraph, and minimum spanning tree weight;*
>
> – *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 [KK15, KKS15] for maximum cut, [EHL+15, BS15] for maximum matching, [GVV17, CGV20] for maximum acyclic subgraph, [FKM+05, HP16] for minimum spanning tree, and [HP16] for the property testing problems. These results were recently extended by [AKSY20] 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 [AKSY20], 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 5.

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. [KKS14, KMNT20, CFPS20, PS18, MMPS17]. In particular, Monemizadeh et. al. [MMPS17] 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 [PS18] 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. [CFPS20] 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.

1.3 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, $f^{\oplus \ell} := \text{XOR}_\ell \circ f = f(x_1) \oplus \cdots \oplus f(x_\ell)$. 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 $\approx \delta^\ell$ (see Appendix A.3). 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 [Yao82, GNW11, IW97, Lev85, Imp95, GRZ20], communi-

cation complexity [VW08, She11, GRZ20], and query complexity [Sha03, She11, BKLS20]. 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. [BJKS02, JRS03, JPY12, BRWY13, RS16, GH09, MWY13, PVZ12] 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 $\leq 1/2 \cdot (1 + (2\delta)^\ell)$.*

In Appendix B, we further discuss the notion of "weak" vs "strong" XOR Lemmas in the context of Result 3 and and in particular show the optimality of this result.

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 [NW91, PRV99, Yeh16, CCM08, GM08, GM09, FKM+08, JRS03, GO13, ACK19, BGGS19, GS20] (see Definition 4.3). 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 [GM09, NW91, Yeh16].

2 Notation and 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 $\mathrm{bias}(\mathsf{X}) := |\Pr(\mathsf{X} = 0) - \Pr(\mathsf{X} = 1)|$; see Appendix A.3 for more details.

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. Appendix A contains the definitions and standard background on these notions that we need in our proofs.

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 (this is the common approach when proving streaming lower bounds; see, e.g. [GM08, LNW14, BGW20]). In particular, we shall define

streaming algorithms as multi-party communication protocols[2] as follows. and then point out the subtle differences with what one typically expect of a streaming algorithm.

Definition 2.1 (**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:*

(i) *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.

(ii) *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.*

(iii) *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.)

Let us point out a couple differences with what one may expect of streaming algorithms. Firstly, we allow our streaming algorithms to do an unbounded amount of work using an unbounded amount of space *between* the arrival of each stream element; we only bound the space in transition between two elements. Secondly, we allow streaming algorithms to maintain a "state" for each stream element across multiple passes (as each player of the streaming algorithm "remembers" the messages it receives in previous passes as well). Finally, a player P_0 is introduced for notational convenience so that every pass of the algorithm involves one message per main players $P_1, \ldots, P_n$.

Clearly, any lower bound proven for streaming algorithms in Definition 2.1 will hold also for more restrictive definitions of streaming algorithm, and that is what we use in this paper. We shall note that however almost all streaming lower bounds we are aware of directly work with this definition and thus we claim no strengthening in proving our lower bounds under this definition; rather, we merely use this formalism to carry out various reductions between our problems.[3]

3   Streaming XOR Lemma

Let Σ_n be any collection of length-n input streams and $f : \Sigma_n \to \{0, 1\}$ be a function which can be interpreted as a streaming decision problem: Given a length-n stream $\sigma \in \Sigma_n$, output $f(\sigma)$. Using f, and for any integer $\ell \geq 1$, we can define another streaming decision problem over length $(n\ell)$-streams: For $\sigma_1, \ldots, \sigma_\ell \in \Sigma_n{}^\ell$, compute $f^{\oplus \ell}(\sigma_1, \ldots, \sigma_\ell)$ over the stream $\sigma_1 \,\|\, \sigma_2 \,\|\, \cdots \,\|\, \sigma_\ell$. We prove the following Streaming XOR Lemma for computing $f^{\oplus \ell}$ in this section.

Theorem 1 (**Streaming XOR Lemma**). *Fix any function $f : \Sigma_n \to \{0, 1\}$, any distribution μ on Σ_n, and any integer $\ell > 1$. Suppose any p-pass s-space streaming algorithm can only compute f over μ with probability at most $\frac{1}{2} + \delta$ for some $\delta > 0$. Then, any p-pass s-space algorithm for $f^{\oplus \ell}$ on the stream $\sigma_1 \,\|\, \cdots \,\|\, \sigma_\ell$ for $(\sigma_1, \ldots, \sigma_\ell) \sim \mu^\ell$ succeeds with probability at most $\frac{1}{2} \cdot (1 + (2\delta)^\ell)$.*

Recall the intuition at the beginning of Section 1.3 behind any form of XOR Lemma: taking XOR of *independent* bits dampens their biases exponentially and thus the algorithm for $f^{\oplus \ell}$ that

---

[2]We refer the reader to [KN97] for the standard definitions from communication complexity used in this paper.

[3]We could have alternatively presented our results in the (NIH) multi-party communication model. However, considering that our proofs work with varying number of players in different steps and that our focus is primarily on proving streaming lower bounds, we found it more natural to work with streaming algorithms directly.

computes each $f(\sigma_i)$ individually satisfies Theorem 1. 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 Theorem 1.

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 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 following:

Proof of Theorem 1

We setup the following game for proving this theorem (see also Figure 2):

(i) There are a total of ℓ players $Q_1, \ldots, Q_\ell$ who receive input streams $\sigma_1, \ldots, \sigma_\ell$, respectively.

(ii) 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).

(iii) 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.

(iv) 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 2.1). 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.

(v) The *cost* of a protocol is the <u>maximum size of the memory</u> 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 3.1. *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 $R_0, R_1, \ldots, R_{n \cdot \ell}$. We can generate a protocol π from A as follows:

(i) 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
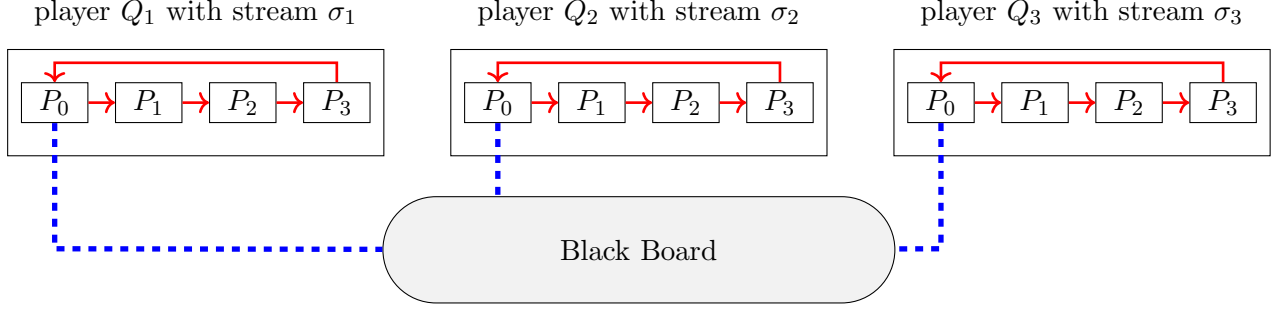
6

Figure 2: An illustration of the communication game in the proof of Theorem 1 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).

first pass of A over their inputs by simulating R_0 to R_n (this incurs a cost of s).

(*ii*) 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.

(*iii*) 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} (again, cost of only s by the space bound of A). Then, like step (*ii*), the message of R_{2n} to R_{2n+1} will be posted on the board via P_0 of $\mathcal{A}_2$.

(*iv*) 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 Theorem 1 by Lemma 3.1.

To continue we need the following definitions:

- For any $i \in [\ell]$ and any choice of the final board content $\mathsf{B}^p = B$, we define:

$$\mathrm{bias}_\pi(i, B) := 2 \cdot \max_{\theta \in \{0,1\}} \Pr_{(\sigma_1,\ldots,\sigma_\ell) \sim \mu^\ell} (f(\sigma_i) = \theta \mid \mathsf{B}^p = B) - 1;$$

  in other words, $\mathrm{bias}_\pi(i, B)$ is equal to $\mathrm{bias}\,(f(\sigma_i))$ for $\sigma_i \sim \mu^\ell \mid \mathsf{B}^p = B$.

- For any choice of the final board content $\mathsf{B}^p = B$, we define:

$$\mathrm{bias}_\pi(B) := 2 \cdot \max_{\theta \in \{0,1\}} \Pr_{(\sigma_1,\ldots,\sigma_\ell) \sim \mu^\ell} \left(f^{\oplus\ell}(\sigma_1, \ldots, \sigma_\ell) = \theta \mid \mathsf{B}^p = B \right) - 1;$$

  in other words, $\mathrm{bias}_\pi(B)$ is equal to $\mathrm{bias}\,(f(\sigma_1) \oplus \cdots \oplus f(\sigma_\ell))$ for $\sigma_1, \ldots, \sigma_\ell \sim \mu^\ell \mid \mathsf{B}^p = B$.

With these definitions, we have,

$$\Pr_{(\sigma_1,\ldots,\sigma_\ell)\sim\mu^\ell} (\pi \text{ is correct}) = \mathbb{E}_B \left[\Pr_{(\sigma_1,\ldots,\sigma_\ell)\sim\mu^\ell} (\pi \text{ is correct} \mid \mathsf{B}^p = B) \right]$$

$$\leq \mathbb{E}_B \left[\max_{\theta\in\{0,1\}} \Pr_{(\sigma_1,\ldots,\sigma_\ell)\sim\mu^\ell} \left(f^{\oplus\ell}(\sigma_1,\ldots,\sigma_\ell) = \theta \mid \mathsf{B}^p = B \right) \right]$$

(conditioned on $\mathsf{B}^p = B$, the answer of π is fixed to some $\theta \in \{0,1\}$)

$$= \mathbb{E}_B \left[\frac{1}{2} \cdot (1 + \text{bias}_\pi(B)) \right] = \frac{1}{2} + \frac{1}{2} \cdot \mathbb{E}_B [\text{bias}_\pi(B)]. \tag{1}$$

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 Theorem 1. 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 3.2. *For any $i \in [\ell]$, $\mathbb{E}_B [\text{bias}_\pi(i, B)] \leq 2\delta$.*

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 Theorem 1 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

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

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

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

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 2 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_i^1. 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_i^1 of π and thus B_{i+1}^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).

8

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 Eq (2). 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 Theorem 1. ∎

To extend the bounds in this lemma to bias(B), we like to use the fact that XOR dampens the bias of *independent* bits (see Appendix A.3). 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.

Lemma 3.3. *For any B, $(\sigma_1, \ldots, \sigma_\ell \sim \mu^\ell \mid \mathsf{B}^p = B) = \times_{i=1}^\ell \left(\sigma_i \sim \mu^\ell \mid \mathsf{B}^p = B \right)$, i.e., the input streams σ_i's are independent _even_ conditioned on $\mathsf{B}^p = B$.*

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_i^j, B_i^j, and B^j, for M_i^j, B_i^j, 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 $i \in [\ell]$,

$$\mathbb{I}(\mathsf{X}_i \,;\mathsf{X}_{-i} \mid \mathsf{B}^p) = 0 \tag{3}$$

where $\mathsf{X}_{-i} = (\mathsf{X}_1, \ldots, \mathsf{X}_{i-1}, \mathsf{X}_{i+1}, \ldots, \mathsf{X}_\ell)$. By Fact A.1-(2), 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 are going to peel off the messages written on the board one by one from the conditioning of Eq (3) without ever increasing the mutual information term. Then, we will 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 originally independent to finalize the proof. Formally,

$$\mathbb{I}(\mathsf{X}_i \,;\mathsf{X}_{-i} \mid \mathsf{B}) = \mathbb{I}(\mathsf{X}_i \,;\mathsf{X}_{-i} \mid \mathsf{B}_{i+1}^p, \mathsf{M}_{i+1}^p, \ldots, \mathsf{M}_\ell^p)$$
(as the content of the board after B_{i+1}^p are only the last messages of players Q_{i+1} to Q_ℓ)
$$\leq \mathbb{I}(\mathsf{X}_i \,;\mathsf{X}_{-i} \mid \mathsf{B}_{i+1}^p),$$

which holds by Proposition A.4 because $\mathsf{X}_i \perp \mathsf{M}_{i+1}^p, \ldots, \mathsf{M}_\ell^p \mid \mathsf{B}_{i+1}^p, \mathsf{X}_{-i}$ 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_{i+1}^p, and thus in the above term, $(\mathsf{M}_{i+1}^p, \ldots, \mathsf{M}_\ell^p)$ is deterministically fixed after conditioning on $\mathsf{B}_{i+1}^p, \mathsf{X}_{-i}$. We can further write,

$$\mathbb{I}(\mathsf{X}_i \,;\mathsf{X}_{-i} \mid \mathsf{B}_{i+1}^p) = \mathbb{I}(\mathsf{X}_i \,;\mathsf{X}_{-i} \mid \mathsf{B}_i^p, \mathsf{M}_i^p)$$
(as the content of the board between B_{i+1}^p and B_i^p changes only by the last message of player Q_i)
$$\leq \mathbb{I}(\mathsf{X}_i \,;\mathsf{X}_{-i} \mid \mathsf{B}_i^p),$$

which again holds by Proposition A.4 because $\mathsf{X}_{-i} \perp \mathsf{M}_i^p \mid \mathsf{B}_i^p, \mathsf{X}_i$ as M_i^p is a deterministic function of B_i^p and X_i. Finally,

$$\mathbb{I}(\mathsf{X}_i \,;\mathsf{X}_{-i} \mid \mathsf{B}_i^p) = \mathbb{I}(\mathsf{X}_i \,;\mathsf{X}_{-i} \mid \mathsf{B}^{p-1}, \mathsf{M}_1^p, \ldots, \mathsf{M}_{i-1}^p)$$
(as the board between B^{p-1} and B_i^p changes only by the last messages of players Q_1 to $i-1$)

9

$$\leq \mathbb{I}(\mathsf{X}_i \, ; \mathsf{X}_{-i} \mid \mathsf{B}^{p-1}),$$

by Proposition A.4, exactly as in the first part above because $\mathsf{X}_i \perp \mathsf{M}_1^p, \ldots, \mathsf{M}_{i-1}^p \mid \mathsf{B}^{p-1}, \mathsf{X}_{-i}$, as conditioning on $\mathsf{B}^{p-1}, \mathsf{X}_{-i}$ 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 Eq (3). Applying this argument for all p rounds, we have that,

$$\mathbb{I}(\mathsf{X}_i \, ; \mathsf{X}_{-i} \mid \mathsf{B}^p) \leq \mathbb{I}(\mathsf{X}_i \, ; \mathsf{X}_{-i} \mid \mathsf{B}^{p-1}) \leq \cdots \leq \mathbb{I}(\mathsf{X}_i \, ; \mathsf{X}_{-i} \mid \mathsf{B}^0) = \mathbb{I}(\mathsf{X}_i \, ; \mathsf{X}_{-i}) = 0,$$

where the last equality is because $\mathsf{X}_i \perp \mathsf{X}_{-i}$ in the distribution μ^ℓ and thus the mutual information is zero between by Fact A.1-(2). This proves Eq (3) and concludes the proof. ∎

Finally, we use Lemmas 3.2 and 3.3 with Eq (1) to bound the success probability of protocol π.

Lemma 3.4. *Protocol π succeeds with probability at most $\frac{1}{2} \cdot \left(1 + (2\delta)^\ell\right)$.*

Proof. We will prove that $\mathbb{E}_B\left[\mathrm{bias}(B)\right] \leq (2\delta)^\ell$ which implies the lemma by Eq (1). Fix any B and consider the random variables $f(\sigma_1), \ldots, f(\sigma_\ell)$ for $(\sigma_1, \ldots, \sigma_\ell) \sim \mu^\ell \mid \mathsf{B}^p = B$. By Lemma 3.3, even in the distribution $\mu^\ell \mid \mathsf{B}^p = B$, 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,

$$\mathrm{bias}_\pi(B) = \mathrm{bias}(f(\sigma_1) \oplus \ldots \oplus f(\sigma_\ell) \mid \mathsf{B}^p = B) = \prod_{i=1}^\ell \mathrm{bias}(f(\sigma_i) \mid \mathsf{B}^p = B) = \prod_{i=1}^\ell \mathrm{bias}_\pi(i, B),$$

where the first and last equalities are by the definitions of $\mathrm{bias}_\pi(B)$ and $\mathrm{bias}_\pi(i, B)$, and the middle equality is by Proposition A.9 and the independence of $f(\sigma_i)$'s conditioned on B, namely, the fact that XOR dampens the biases of independent random bits. Finally,

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

where the last equality is by the independence of $\mathrm{bias}_\pi(i, B)$ and the inequality by Lemma 3.2. ∎

Theorem 1 now follows immediately from Lemmas 3.1 and 3.4.

4 The Lower Bound for the Noisy Gap Cycle Counting Problem

We prove our main streaming lower bound for the noisy gap cycle counting problem in this section, defined formally as follows (see also Figure 1 in the Introduction for an illustration).

Definition 4.1 (**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.*

We prove the following lower bound for this problem that formalizes Result 1.

Theorem 2. *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.*

10

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. The rest of this section is organized as follows. We first design a hard input distribution for $\mathbf{NGC}_{n,k}$ and prove its useful properties for our purpose. We then give a high level plan of the lower bound for this distribution, followed by the details of each step, and the proof of the theorem. We start out with some extra definitions as we need some structure on the family of graphs we work with in order to describe our hard distribution.

Definition 4.2 (**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:*

(i) *Vertices V consist of $d + 1$ layers of vertices $V_1, \ldots, V_{d+1}$, each of size w.*

(ii) *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 4.3 (**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).*

Figure 3 gives an illustration of this problem.

(a) X-instance of $\mathbf{PC}_{6,4}$ (b) Y-instance of $\mathbf{PC}_{6,4}$
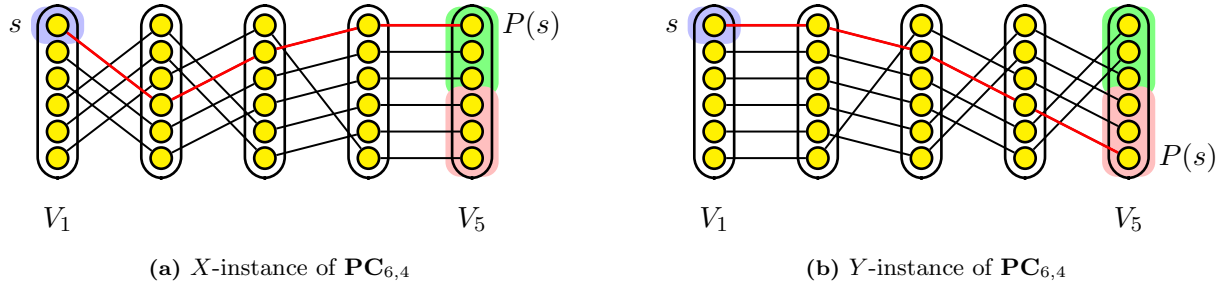
Figure 3: An illustration of $\mathbf{PC}_{m,b}$ for $m = 6$ and $b = 4$. The edges are perfect matchings that go between consecutive vertex layers. The start vertex s is depicted in the vertex layer V_1, and the sets X, Y are marked in the vertex layer V_5. $P(s)$, the unique vertex in V_5 reachable from s is also shown.

4.1 A Hard Distribution for NGC

We can now define our hard input distribution. We 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 Figure 4 for an illustration).
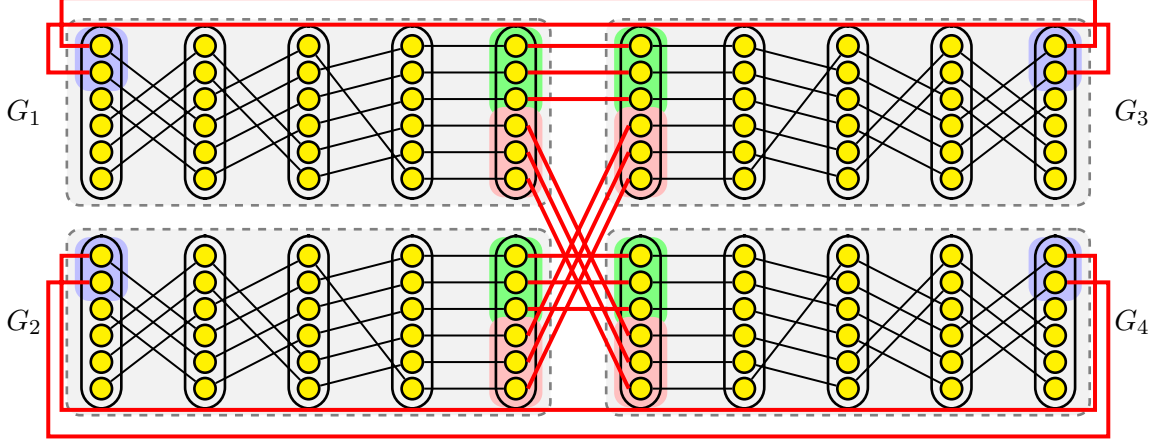
11

Figure 4: 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.

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

(i) 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*).

(ii) 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).

(iii) The input stream consists of edges inserted in $((ii)c)$ 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 $M_d^1 \, || \, \cdots \, || \, M_1^1 \, || \, \cdots \, || \, M_d^4 \, || \, \cdots \, || \, M_1^4$.

A careful reader may have noticed that Distribution 1 already impose some constraints on the choice of k, e.g., k needs to be even and more constraints will also follow (we also need t to be an even number so V_{d+1} admits an equipartition). Yet Theorem 2 is supposed to work for all choices of k, in particular, we need odd values of k for some of our reductions. We will however fix this later by some simple padding argument when concluding the proof of Theorem 2.

We start by proving that this distribution indeed outputs valid instances of $\mathbf{NGC}_{n,k}$.

Lemma 4.4. *Let G be a graph sampled from Distribution 1. Then,*

(i) *if G_0 is a X-instance, then G consists of $2t$ cycles of length k and $4t$ paths of length $k-1$;*

(ii) *if G_0 is a Y-instance, then G consists of t cycles of length $2k$ and $4t$ paths of length $k-1$.*

Proof. Suppose first that G_0 is a X-instance. Fix a vertex $v \in S$ of G_0 and consider $copy(v,1) \in S^1$. We claim that $copy(v,1)$ belongs to the following k-cycle in G:

$$copy(v,1) \in S^1 \rightsquigarrow copy(P(v),1) \in X^1 \rightarrow copy(P(v),3) \in X^3 \rightsquigarrow copy(v,3) \in S^3 \rightarrow copy(v,1) \in S^1;$$

the first path exists by the definition of $P(v) \in X$ in G_0, the next edge by the identity perfect matching between X^1 and X^3, the next path by reversing the identical path $P(v) \in X$ to v in G_0, and the final edge by the identity perfect matching between S^1 and S^3. Note that this cycle in particular also contains $copy(v,3) \in S^3$ and its length is $d + 1 + d + 1 = 2 \cdot \frac{(k-2)}{2} + 2 = k$.

As a result, there are $|S| = t$ disjoint k-cycles, each including a pair of vertices from S^1 and S^3. By symmetry, this is also happening for another t disjoint k-cycles for vertices of S^2 and S^4, which gives us $2t$ disjoint k-cycles as desired. Moreover, as these k-cycles cover, for every $i \in [4]$, all vertices of S^i and all other remaining vertices in V^i have degree one, there is no other cycle; instead, vertices of $V^i \setminus S^i$ belong to a path of length $k-1$, giving us $4t$ disjoint $(k-1)$-paths.

Now suppose that G_0 is a Y-instance. Again, fix a vertex $v \in S$ of G_0 and consider $copy(v,1) \in S^1$. We now claim that $copy(v,1)$ belongs to the following $2k$-cycle in G:

$$copy(v,1) \in S^1 \rightsquigarrow copy(P(v),1) \in Y^1 \rightarrow copy(P(v),4) \in Y^4 \rightsquigarrow copy(v,4) \in S^4 \rightarrow copy(v,2) \in S^2,$$
$$copy(v,2) \in S^2 \rightsquigarrow copy(P(v),2) \in Y^2 \rightarrow copy(P(v),3) \in Y^3 \rightsquigarrow copy(v,3) \in S^3 \rightarrow copy(v,1) \in S^1;$$

this is exactly as in case (i) except that the switch in connecting Y^1 to Y^4 and Y^2 to Y^3 instead, results in $copy(v,1)$ reaching $copy(v,2)$ and thus it takes another "round" for $copy(v,2)$ to reach $copy(v,1)$ also, doubling the length of the cycle. The rest of the argument is exactly as before and we omit the details (but, notice that this doubling effect will not happen for the $(k-1)$-paths as their endpoints have degree one only). This concludes the proof. ∎

In addition to proving the validity of distribution Distribution 1, Lemma 4.4 also has a message for our lower bound: any algorithm that can solve $\mathbf{NGC}_{n,k}$ over the distribution μ_{NGC}, can also decided whether a given graph G_0 sampled in μ_{NGC} is a X-instance or a Y-instance (since the instance is defined deterministically given G_0, and the remaining edges are input-independent, we can just do a straightforward reduction between the two problems). This latter problem is what we prove our lower bound for.

Before moving on, let us also make the following important remark about the source of hardness of $\mathbf{NGC}$ (over μ_{NGC}) that will be used in several of our reductions in the subsequent section.

Remark 4.5. *The following information is known by any streaming algorithm that solves $\mathbf{NGC}$ over the distribution μ_{NGC}:*

(i) *One endpoint of every noise path in the graph G;*

(ii) *A set of t four-tuples of vertices (u_1, u_2, u_3, u_4) such that in the k-cycle case u_1, u_2 and u_3, u_4 belong to two disjoint k-cycles each, while in the $2k$-cycle case, all belong to the same $2k$-cycle.*

Notice that these vertices cover all paths and cycles of the graph in both cases.

The first point is because the set S in G_0 is fixed and we can take $copy(v, i)$ for $v \notin S$ and $i \in \{1, 2\}$ as endpoints of every path. The second point is similarly because we can take the tuples (u_1, u_2, u_3, u_4) to be $u_1 = copy(v, 1), u_2 = copy(v, 3), u_3 = copy(v, 2), u_4 = copy(v, 4)$ and use the proof of Lemma 4.4 to see that these vertices belong to desired cycles.

4.2 The High Level Plan

Our plan for proving Theorem 2 involves the following three steps (the reader is encouraged to check the algorithm for **NGC** in Appendix C as it helps with the intuition of this proof).

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 Theorem 1 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 (Theorem 1) 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: a 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 [GM09, NW91, PRV99, Yeh16], 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.

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.

4.3   Step One: Decorrelating Distribution 1

Our goal in this section 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.

(i) 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).

(ii) 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 4.6. *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 <u>even</u> $m := \Theta(n/k)$ 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 4.6 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 $\boldsymbol{\mu(f)}$**: 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 $\mu_{\mathsf{NGC}} = \frac{1}{2} \cdot \mu(0^t) + \frac{1}{2} \cdot \mu(1^t)$. 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:

$$f^0 = (0, \dots, 0), \quad f^1 = (1, 0, \dots, 0), \quad \cdots \quad f^i = (\underbrace{1, \dots, 1}_{i}, 0, \dots, 0), \quad \cdots \quad f^t = (1, \dots, 1).$$

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

Claim 4.7. *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 $\mu_{\mathsf{NGC}} = \frac{1}{2} \cdot \mu(0^t) + \frac{1}{2} \cdot \mu(1^t)$ with probability of success at least $2/3$, we have that (see Fact A.7),

$$\| \left(mem(A) \mid \mu(f^0) \right) - \left(mem(A) \mid \mu(f^t) \right) \|_{\mathrm{tvd}} \geq 1/3, \tag{4}$$

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,

$$\| \left(mem(A) \mid \mu(f^0) \right) - \left(mem(A) \mid \mu(f^t) \right) \|_{\mathrm{tvd}} \leq \sum_{i=1}^{t} \| \left(mem(A) \mid \mu(f^{i-1}) \right) - \left(mem(A) \mid \mu(f^i) \right) \|_{\mathrm{tvd}},$$

which, together with Eq (4), implies that there is an index $i^* \in [t]$ such that

$$\| \left(mem(A) \mid \mu(f^{i^*-1}) \right) - \left(mem(A) \mid \mu(f^{i^*}) \right) \|_{\mathrm{tvd}} \geq \frac{1}{3t}. \tag{5}$$

Thus, A distinguishes between $\mu(f^{i^*-1})$ and $\mu(f^{i^*})$ with probability $\geq \frac{1}{2} + \frac{1}{6t}$ by Fact A.7. ∎

Let us define our final distribution $\mu^* := \frac{1}{2} \cdot \mu(f^{i^*-1}) + \frac{1}{2} \cdot \mu(f^{i^*})$. Claim 4.7 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 4.8. *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[4]. 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 4.8 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 4.7 and an averaging argument, there is a choice of R_1, R_2 such that,

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

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 4.6.

Proof of Lemma 4.6. Let c be the parameter in Claim 4.8 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 $G \sim \mu_{\mathsf{PC}}$, algorithm B uses Claim 4.8 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 Eq (6). 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 2.1, the players are computationally unbounded and so can do their part of creating the graph G_0 without any

---

[4]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.

communication). By the independence property argued for Eq (6), 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. ∎

4.4 Step Two: Applying the Streaming XOR Lemma

By Lemma 4.6, 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 Theorem 1, 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 4.9. *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 4.9 is the streaming XOR Lemma that we already established; 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 $\mathbf{PC}$ problem, this product generates a graph $H := \oplus_{i=1}^{\ell} G_i$ such that the answer to a $\mathbf{PC}$ problem on H is equal to XOR of answers to $\mathbf{PC}$ 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 5):

- **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 $\mathbf{PC}$ problem, we write $\mathbf{PC}(G) \in \{0, 1\}$ to denote the answer of $\mathbf{PC}$ 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

$$s = copy(v, 1, 1), \qquad X = U_1^{\ell,3}, \qquad Y = U_1^{\ell,4},$$
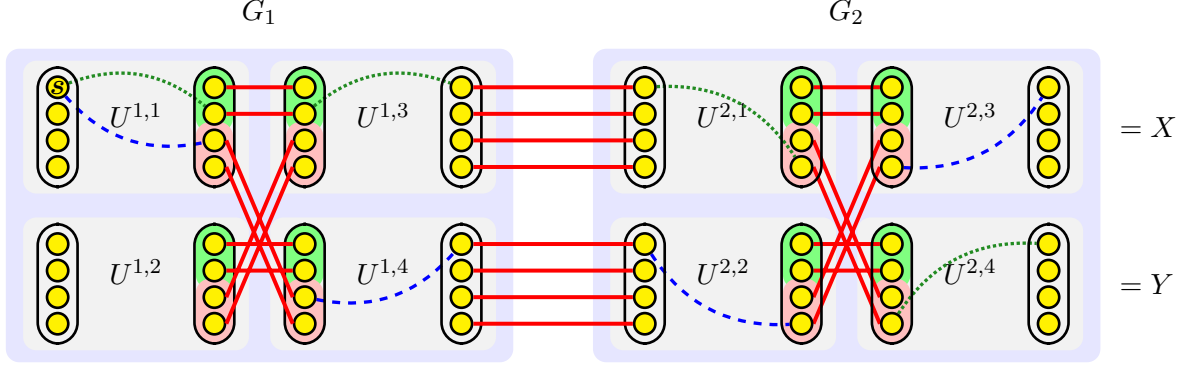
18

Figure 5: 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.

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 5).

Claim 4.10. *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 in H:

$$s^{r,1} = copy(s_r, r, 1) \rightsquigarrow copy(P(s_r), r, 1) \in X^{r,1} \rightarrow copy(P(s_r), r, 3) \in X^{r,3} \rightsquigarrow copy(s_r, r, 3) = s^{r,3};$$

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:

$$s^{r,1} = copy(s_r, r, 1) \rightsquigarrow copy(P(s_r), r, 1) \in Y^{r,1} \rightarrow copy(P(s_r), r, 4) \in Y^{r,4} \rightsquigarrow copy(s_r, r, 4) = s^{r,4};$$

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).

19

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 4.10, we can now prove Lemma 4.9.

Proof of Lemma 4.9. Consider ℓ *independent* instances $G_1, \ldots, G_\ell$ of $\mathbf{PC}_{\hat{m},\hat{b}}$ sampled from $\mu_{\mathsf{PC}}{}^\ell$. 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 4.9). By Claim 4.10, we have $\mathbf{PC}(H) = \oplus_{r=1}^{\ell} \mathbf{PC}(G_r)$.

We can now apply (the contrapositive of) our streaming XOR Lemma (Theorem 1) to obtain: if we have a p-pass s-space streaming algorithm for $\mathbf{PC}$ over the distribution of $H = \oplus_{r=1}^{\ell} G_r$ over $\mu_{\mathsf{PC}}{}^\ell$ 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 μ_{PC} with $1/2 \cdot (1 + (2\delta)^{1/\ell})$ (by re-parameterizing δ to match that of Theorem 1).

We are still however not done because the algorithm A in the lemma works on the distribution μ_{PC} while the distribution H induced by $\mu_{\mathsf{PC}}{}^\ell$ does not match μ_{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. ∎

4.5 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 prove the following lemma in this section.

Lemma 4.11. *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 $s = \Omega(\frac{1}{b^5} \cdot \hat{m}^{1-4/\ell})$.*

The proof of this lemma is similar to the known communication complexity lower bounds for pointer chasing such as [GM09, NW91, PRV99, Yeh16]; 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 [GM09]). 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 [GM09, Yeh16] with some key modifications, in particular to allow for handling random matchings. We prove the following proposition, which implies Lemma 4.11.

Proposition 4.12. *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 $s = \Omega\left(\frac{\gamma^4}{b^5} \cdot m\right)$.*

20

This is a good place to point out concretely why we need step two of our approach in Lemma 4.9, instead of simply applying Proposition 4.12 directly to our original **PC** problem in the reduction of Lemma 4.6. 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 [GM09, NW91, PRV99, Yeh16] 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 4.6. As such, the hardness amplification step of Lemma 4.9 is the crucial step in our approach.

We postpone the proof of Proposition 4.12 to Section 6 to keep the flow of the current argument and instead show how Lemma 4.11 follows immediately from this.

Proof of Lemma 4.11. 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 4.12. 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 4.12, 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. ∎

4.6 Putting Everything Together: Proof of Theorem 2

We are now ready to prove Theorem 2.

Proof of Theorem 2. 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 4.6, 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 4.6, 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 4.6 in this step.

- **Step three:** By Lemma 4.11, 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 2 for infinitely many values of $k \in \mathbb{N}^+$, i.e., the ones where $\frac{k-4}{4p+8}$ is an integer.

We now extend this lower bound to all values of k. Given any integer k, find the largest integer $\tilde{k} \le k$ so that $\frac{\tilde{k}-4}{4p+8}$ is an integer. Clearly, $\tilde{k} = \Theta(k)$. Sample a graph G from μ_{NGC} of Distribution 1 for parameters n and $\tilde{k}$. Recall that in step $((ii)\mathrm{c})$ of Distribution 1, we connect the sets S^1 to S^3, and S^2 to S^4 using identity perfect matchings. We now replace each of these edges with a path of length $k - \tilde{k} + 1$ (or equivalently, put $k - \tilde{k}$ new vertices in the middle of each path). This can only increase the number of vertices by a constant factor.

In this new graph, the length of each original $\tilde{k}$-cycle becomes $(\tilde{k} - 1) + k - \tilde{k} + 1 = k$, and each $2\tilde{k}$-cycle becomes $2(\tilde{k} - 1) + 2(k - \tilde{k} + 1) = 2k$. As such, we can apply the lower bound for parameters n and $\tilde{k}$ to this graph as well and since the number of vertices and k are asymptotically the same, we obtain the desired lower bound. This finalizes the proof of Theorem 2. ∎

We conclude this section by stating a corollary (of the proof) of Theorem 2 and Remark 4.5 that we will use in some of our reductions (and can also be useful for future reductions from **NGC**).

Corollary 4.13. *For every $k \in \mathbb{N}^+$, the lower bound of Theorem 2 continues to hold even if we additionally provide the following information to the algorithm beforehand:*

(i) One endpoint of every noise path in the graph G;

(ii) A set of t four-tuples of vertices (u_1, u_2, u_3, u_4) such that in the k-cycle case u_1, u_2 and u_3, u_4 belong to two disjoint k-cycles each, while in the $2k$-cycle case, all belong to the same $2k$-cycle.

Moreover, this lower bound also hold when the graph is directed *with directed k-cycles and $(k-1)$-paths or directed $2k$-cycles and $(k-1)$-paths.*

Proof. The first two parts follow immediately from Remark 4.5 as when proving the lower bound for Distribution 1, we anyway assume this information was known by the streaming algorithm. The last part follows because we can alter Distribution 1 to direct the edges from the first layer to the last one and back (left-to-right for "inside" edges in Figure 4 and right-to-left for "outside" ones), which makes the cycles and paths directed. Again, the lower bound holds verbatim for this distribution as well because the partitioning of vertices in the layers are fixed in Distribution 1 and so the direction of the edges does not reveal any new information to the algorithm. ∎

5 Other Graph Streaming Lower Bounds via Reductions

We now present several reductions from **NGC** for establishing graph streaming lower bounds. These results collectively formalize Result 2. Our reductions are similar in spirit to the ones in prior work and particularly [AKSY20] (based on their *One-or-Many-Cycles problem* which is another variant of gap cycle counting problems); the main novelty here is how we can handle the "noise" part of **NGC** in the reductions but this is not particularly challenging (and become mostly relevant to problems such as minimum weight spanning tree or property testing of connectivity). In the following, we will define each problem and present the most relevant prior work, our new result, and a short discussion on whether our result is/seems to be optimal or not. We refer the interested reader to [AKSY20, Section 7 and Appendix B] for a comprehensive summary of the prior results on the problems studied in this section.

Before moving on, let us note that the work of [AKSY20] also considers some lower bounds beyond graph streams for problems such as Schatten norms of matrices or Sorting-by-Block-Interchange; our new lower bounds also apply to these problems with some additional work but we

opted to focus primarily on graph streaming lower bounds in this paper (with the exception of the Matrix Rank problem which follows immediately from our results).

5.1 Minimum Spanning Tree

Given an undirected graph $G = (V, E)$, with edge-weights $w : E \rightarrow \{1, 2, \ldots, W\}$, 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. This is one the earliest problems studied in the streaming setting [FKM+08, AGM12, HP16] and it is known that $O(n \log(nW))$ space and a single-pass suffices for finding an *exact* MST [FKM+08] and $n^{1-\Theta(\varepsilon/W)}$ and a single-pass for finding a $(1+\varepsilon)$-approximation [HP16]. On the lower bound front, $\Omega(n)$ and $n^{1-O(\varepsilon/W)}$ lower bounds for exact answer and approximate answer via single-pass algorithms where proved in [FKM+08] and [HP16], respectively. The only known multi-pass lower bound for $(1+\varepsilon)$-approximation is that of [AKSY20] that proves that $n^{o(1)}$-space needs $\Omega(\log(1/\varepsilon))$ passes.

We present the following lower bound for this problem:

Theorem 3. *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 $W = O(1)$.*

Proof. Consider a **NGC** instance G on $n = 6t \cdot k$ vertices from μ_{NGC} for largest integer $k \leq \frac{W-1}{12\varepsilon}$. We crucially use Corollary 4.13 to perform this reduction.

Using Corollary 4.13 allows us to assume that the algorithm for G has the extra knowledge of the following: a collection of t tuples $(u_1^i, u_2^i, u_3^i, u_4^i)$ for $i \in [t]$ (covering the cycles) and $4t$ vertices $v_1, \ldots, v_t$ (covering the paths), with the properties specified by Corollary 4.13.

Pick any arbitrary bijection $\phi : [t] \rightarrow [t]$ such that $\phi(i) \neq i$ for $i \in [t]$, and any arbitrary injective one-to-three mapping $\psi : [t] \rightarrow [4t]$. Create the following graph from G (see Figure 6):

(i) For any $i \in [t]$, connect u_1^i to $u_3^{\phi(i)}$ with a new edge of weight 1.

(ii) For any $i \in [t]$, connect u_2^i to u_4^i with a new edge of weight W.

(iii) For any $i \in [t]$, let $\psi(i) = (a_1^i, a_2^i, a_3^i, a_4^i)$ and connect u_1^i to $v_{a_1^i}, v_{a_2^i}, v_{a_3^i}, v_{a_4^i}$.

This addition to the graph can be created by any streaming algorithm of Definition 2.1 without spending any extra space. In the following, we use H to refer to this new graph obtained from G and H_1 to be the subgraph of H consists of only the edges of weight 1 in H.

Firstly, suppose that G belongs to the k-cycle case. We claim that in this case, H_1 consists of t connected components. These components are, for all $i \in [t]$, u_1^i and its cycle in G (including u_2^i), plus $u_3^{\phi(i)}$ and its cycle in G (including $u_3^{\phi(i)}$), plus $v_{a_1^i}, v_{a_2^i}, v_{a_3^i}, v_{a_4^i}$ and their paths in G for $(a_1^i, a_2^i, a_3^i, a_4^i) = \psi(i)$; all these vertices belong to the same connected and all edges going out of this component only has weight W and thus does not belong to H_1.

On the other hand, we claim that when G belongs to the $2k$-cycle case, H_1 becomes connected. For all $i \in [t]$, the cycle of u_1^i also contains u_3^i; for the sake of argument suppose we add a new edge of weight 1 between u_1^i and u_3^i (this cannot change the connectivity of the graph H_1). Now we have two edge-disjoint perfect matchings between $\{u_1^i \mid i \in [t]\}$ on one side and $\{u_3^i \mid i \in [t]\}$ on the other side, one from the new (artificial) edges we just added and another from the edges
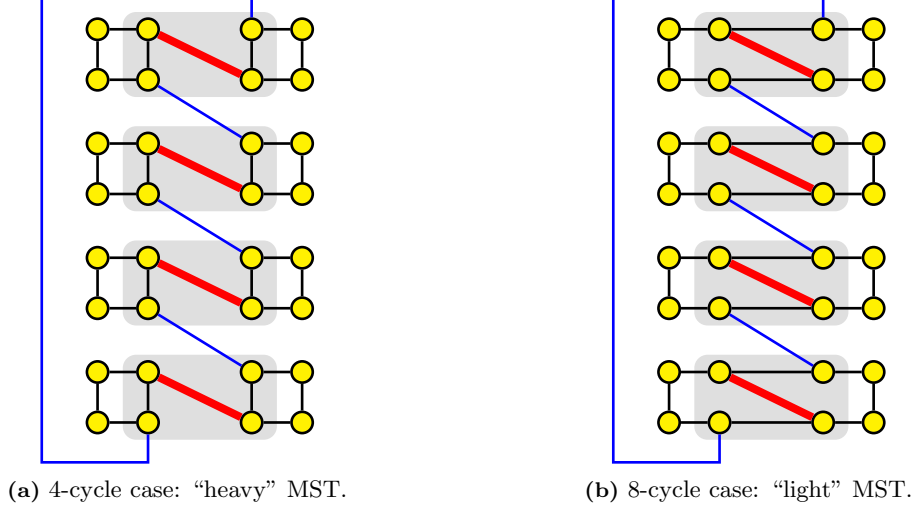
(a) 4-cycle case: "heavy" MST. **(b)** 8-cycle case: "light" MST.

Figure 6: An illustration of the MST reduction. The vertices inside each block (gray) are one of the four tuples $(u_1^i, u_2^i, u_3^i, u_4^i)$. The middle thick (red) edges have weight W while middle thin (blue) edges have weight 1; both these groups of edges are added as part of the reduction. The outer thin (black) edges are the original edges of the 4-cycle vs 8-cycle problem. To avoid clutter, we have not drawn the noise paths, however, they can be thought of as being partitioned into t groups of size four and connecting group i to vertex u_1^i of each tuple using an edge of weight 1; clearly, this does not break planarity. The graph on the left has 4 connected component without the heavy edges while the right one is connected even without those edges.

in H_1 between u_1^i and $u_3^{\phi(i)}$. These two matchings form a Hamiltonian cycle over these sets thus connecting them all together. As all other vertices of the graph are connected to some u_1^i, the entire graph becomes connected.

Now in the first case, any MST of H needs to pick at least $t - 1$ edges from $H \setminus H_1$ which are all of weight W, making its weight at least $n - t + (t - 1) \cdot W$ (the fact that H itself is connected is exactly the same H_1 being connected in the second case). In the second case, every MST of H has weight $n - 1$ as H_1 is already connected. The choice of k ensures that the weight of MST of H in the first case is $(1 + \varepsilon)$ times larger that the second case.

As such a p-pass streaming algorithm for $(1 + \varepsilon)$-approximation MST can be used as a distinguisher for the two cases of the graph G. By Corollary 4.13, we have that that space complexity of the algorithm should be

$$\Omega\left(\frac{1}{p^5} \cdot (n/k)^{1-O(p/k)}\right) = \Omega\left(\frac{1}{p^5} \cdot (\varepsilon \cdot n/W)^{1-O(\varepsilon \cdot p/W)}\right).$$

The fact that the lower bound holds on bounded-degree planar graphs is simply because the distribution μ_{NGC} in Distribution 1 we use here is supported on graphs which are disjoint-union of cycles and paths and the set of edges we added increase the degree by at most a constant factor and does not break the planarity (see Figure 6 for an illustration). ∎

Optimality of Theorem 3? The bounds obtained by our algorithm are actually *asymptotically optimal* for any constant $W \in \mathbb{N}^+$. To obtain the upper bound, we can run a streaming implementation of the query algorithm of [CRT05] in $O(1/\varepsilon)$ passes and $O_\varepsilon(\text{polylog } n)$ space (see [HP16] and [MMPS17] for details on simulating these query algorithms in the streaming model – note that in general, by allowing p passes over the input, we can simulate p rounds of adaptive querying in a straightforward way).

5.2 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. Maximum matching is among the most studied problems in the streaming setting and listing the prior results on this problem is beyond the scope of our work. We only note that there are various algorithms for approximating matching size in polylog(n) space in arbitrary graphs in random-order streams [MMPS17,KKS14,KMNT20] or planar graphs in adversarial streams [EHL$^+$15, CCE$^+$16, CJMM17, MV18]. The best single-pass lower bounds for this problem rule out $< (3/2)$-approximation in $o(\sqrt{n})$ space [EHL$^+$15] and $(1 + \varepsilon)$-approximation in $n^{1-O(\varepsilon)}$ space [BS15] (and $n^{2-O(\varepsilon)}$ space on dense graphs [AKL17]); the only known multi-pass lower bound is that $(1 + \varepsilon)$-approximation in $n^{o(1)}$-space needs $\Omega(\log(1/\varepsilon))$ passes [AKSY20].

We present the following lower bound for this problem:

Theorem 4. *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.*

Proof. Consider a **NGC** instance G on $n = 6t \cdot k$ vertices from μ_{NGC}, for largest *odd* integer $k \leq \frac{1}{3\varepsilon}$.

In one case, G contains $2t$ vertex-disjoint *odd* cycles of length k each, and $4t$ vertex disjoint paths of length $k - 1$. Any maximum matching of G can match $\frac{k-1}{2}$ edges from each odd cycle and $\frac{k-1}{2}$ from each path of length $k - 1$. Thus the value of maximum matching in this case is $t \cdot (k - 1) + 2t \cdot (k - 1)$, which equals $(n/2) - (n/2k)$.

In the other case, G contains t vertex-disjoint *even* cycles of length $2k$ each, and $4t$ vertex-disjoint paths of length $k - 1$. Any maximum matching of G can match k edges from each odd cycle and $\frac{k-1}{2}$ from each path of length $k - 1$. Thus the value of maximum matching in this case is $t \cdot k + 2t \cdot (k - 1)$, which equals $(n/2) - (n/2k) + (n/6k) > (n/2) - (n/2k) + \varepsilon(n/2)$.

As such a p-pass streaming algorithm for $(1 + \varepsilon)$-approximation of maximum matching can be used as a distinguisher for the two cases of the graph G. By Theorem 2, we have that that space complexity of the algorithm should be

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

The fact that the lower bound holds on bounded-degree planar is simply because the distribution μ_{NGC} in Distribution 1 we use here is supported on graphs which are disjoint-union of cycles and paths, and thus clearly are both bounded-degree and planar. ∎

As a consequence of the standard equivalence between estimating matching size and computing the rank of the Tutte matrix [Tut47] with entries chosen randomly established in [Lov79] ([BS15] performs this reduction in the streaming model), we get the following result as well.

Corollary 5.1. *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.*

This result considerably strengthen prior bounds in [BS15,LW16,AKSY20] for this fundamental problem.

Optimality of Theorem 4? Theorem 4 provides the currently best multi-pass lower bound for $(1 + \varepsilon)$-approximation of maximum matching in *any* family of graphs. We do not know whether there is a matching upper bound as well, namely, an algorithm with $n^{o(1)}$-space and $O(1/\varepsilon)$ passes on general graphs or the lower bound can be improved further.

There are however already known algorithms for this problem on bounded degree graphs. In particular, [MMPS17] give a streaming implementation of the query algorithm of [NO08] that achieves a $\pm \varepsilon n$ *additive* approximation to maximum matching in $O_\varepsilon(\log n)$ bits of space in a single-pass in *random-streams*; the same exact algorithm can also be implemented in this much space and $O_\varepsilon(1)$ passes in *arbitrary* streams although the dependence is much worse than $\Omega(1/\varepsilon)$ in our lower bound. Closing this gap remains a fascinating open question.

5.3 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. This problem has been studied extensively in the graph streaming model in [KKS15, KK15, BDV18, KK19, KKSV17, AKSY20], with best lower bound of < 2-approximation in $\Omega(n)$ space in single-pass graphs [KK19] and $(1+\varepsilon)$-approximation in $n^{o(1)}$-space in $\Omega(\log(1/\varepsilon))$ passes [AKSY20].

We present the following lower bound for this problem.

Theorem 5. *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.*

Proof. Consider a **NGC** instance G on $n = 6t \cdot k$ vertices from μ_{NGC} for largest *odd* integer $k \le \frac{1}{3\varepsilon}$.

In one case, G contains $2t$ vertex-disjoint *odd* cycles of length k each, and $4t$ vertex disjoint paths of length $k - 1$. Any cut of G must leave out one edge from each cycle, and thus, the value of maximum cut in this case is $2t \cdot (k - 1) + 4t \cdot (k - 1)$, which equals $n - n/k$.

In the other case, G contains t vertex-disjoint *even* cycles of length $2k$ each, and $4t$ vertex-disjoint paths of length $k - 1$. Thus, G *is bipartite* and there exists a cut such that all edges in the graph cross that cut. The value of maximum cut in this case is $t \cdot 2k + 4t \cdot (k - 1)$, which equals $n - 2n/3k = (n - n/k) + n/3k \ge (n - n/k) + \varepsilon n$.

As such a p-pass streaming algorithm for $(1+\varepsilon)$-approximation of maximum cut can be used as a distinguisher for the two cases of the graph G. By Theorem 2, we have that that space complexity of the algorithm should be

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

The fact that the lower bound holds on bounded-degree planar is simply because the distribution μ_{NGC} in Distribution 1 we use here is supported on graphs which are disjoint-union of cycles and paths, and thus clearly are both bounded-degree and planar. ∎

Optimality of Theorem 5? Theorem 5 provides the currently best multi-pass lower bound for $(1 + \varepsilon)$-approximation of maximum cut in *any* family of graphs. We are not sure if it is possible to obtain a $n^{o(1)}$-space algorithm with $O(1/\varepsilon)$-passes (or for that matter, even independent of n)

for this problem on general graphs and thus the lower bound can perhaps be improved further. We shall remark however that a $(1 + \varepsilon)$-approximation of maximum cut is possible in a single pass and $\widetilde{O}(n/\varepsilon^2)$ space or even $o(n)$ space for sufficiently dense graphs [BDV18] (by computing a cut sparsifier in streaming; see [AG09]), when we allow exponential time to the algorithms.

On the other hand, we suspect that such an algorithm should be possible for planar graphs (or at least for the bounded-degree ones – note that finding maximum cut in planar graphs is closely tied to the maximum weight matching problem and is also solvable in polynomial time [Had75]); we leave this question as an interesting open problem for future work.

5.4 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. This is a canonical CSP problem (alongside maximum cut) and has been studied in the streaming model in [GT19, GVV17, CGMV20, AKSY20, CGV20], with the best lower bound of $\Omega(\sqrt{n})$ for single-pass algorithm with $< (7/8)$-approximation, and $\Omega(\log(1/\varepsilon))$ pass lower bound for $(1+\varepsilon)$-approximation algorithms with $n^{o(1)}$-space [AKSY20].

We present the following lower bound for this problem:

Theorem 6. *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.*

Proof. The proof is along the previous lines, except that we are going to apply it to the *directed* version of the problem in Corollary 4.13. Consider a **NGC** instance G on $n = 6t \cdot k$ vertices drawn from μ_{NGC}, for largest integer $k \leq \frac{1}{6\varepsilon}$. In line with Corollary 4.13, we assume that G is directed.

Since cycles of G are disjoint, the size of the largest acyclic subgraph of G is exactly equal to the number of edges of G minus its number of cycles. The number of edges in our graphs is $2t \cdot k + 4t \cdot (k - 1)$ in both cases. The number of cycles is $2t$ in one case and t in another. Thus, the size of largest acyclic subgraph in one case is $6t \cdot k - 6t = n - (n/k)$ and in the other case it is $6t \cdot k - 5t = n - (n/k) + (n/6k) = n - (n/k) + \varepsilon \cdot n$.

As such a p-pass streaming algorithm for size of a largest acyclic subgraph can be used as a distinguisher for the two cases of the directed graph G. By Corollary 4.13, we have that that space complexity of the algorithm should be

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

The fact that the lower bound holds on bounded-degree planar is simply because the distribution μ_{NGC} in Distribution 1 we use here is supported on graphs which are disjoint-union of cycles and paths, and thus clearly are both bounded-degree and planar. ∎

Optimality of Theorem 6? Theorem 6 provides the currently best multi-pass lower bound for $(1 + \varepsilon)$-approximation of maximum acyclic graph in *any* family of graphs. However, we are not aware of any algorithmic work on this problem in the streaming setting except for [CGMV20] who considered the closely related problem of feedback arc set: minimum number of edges that needs to be deleted from the graph before making it acyclic (this number is equal to the number of edges

27

minus the answer to our original problem). As such, at this point, we do not know much about the complexity of this problem and consequently optimality of Theorem 6 (note however that a 2-approximation in $O(\log n)$ space is trivial by returning half the number of edges).

5.5 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 define a graph G as being ε-far from the properties we consider as follows:

- **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.

Traditionally, these problems have been studied in the query complexity model, but more recently, they also received an extensive attention in the streaming model [HP16,CFPS20,PS18,MMPS17]. In particular, [HP16] gave an upper bound of $n^{1-\Theta(\varepsilon)}$-space and single-pass for the first two-problems and $n^{1-\Theta(\varepsilon^2)}$ and single-pass for the latter problem on *planar* graphs. From the lower bound perspective, [HP16] proved $n^{1-O(\varepsilon)}$ space lower bounds for these problems in single-pass streams and [AKSY20] proved that $n^{o(1)}$-space algorithms require $\Omega(\log(1/\varepsilon))$ passes.

We prove the following lower bound for these problems:

Theorem 7. *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.*

Proof. The proofs of each of these parts are very similar to the previous results of this section and thus we only briefly mention each one.

Connectivity: The proof of this part is identical to that of Theorem 3 with the difference that we no longer add edges of weight W. Thus, in one case the graph has εn connected components and in the other case it is connected. The rest follows verbatim as Theorem 3.

Bipartiteness: The proof of this part is identical to that of Theorem 5. The graphs in Theorem 5 in one case miss εn edge from any cut, thus need εn of their edges removed to become bipartite, while in the other case they are bipartite. The rest follows verbatim as Theorem 5.

Cycle-freeness: Consider a **NGC** instance G on $n = 6t \cdot k$ vertices drawn from μ_{NGC} for largest integer $k \leq \frac{1}{6\varepsilon}$. We crucially use Corollary 4.13 to perform this reduction.

Using Corollary 4.13 allows us to assume that the algorithm for G has the extra knowledge of the following: a collection of t tuples $(u_1^i, u_2^i, u_3^i, u_4^i)$ for $i \in [t]$ (covering the cycles) with the properties specified by Corollary 4.13 (we do not need the extra knowledge of the paths here).

Given a graph G sampled from μ_{NGC}, simply remove one arbitrary edge incident on u_1^i for all $i \in [t]$. This way, u_1^i cannot be part of any cycle in G. Now, in one case, G still has t edge-disjoint

k-cycles and thus we need to remove $t = (n/6k) = \varepsilon n$ edges from G to make it cycle-free. In the other case, every cycle of G has lost an edge and thus it is cycle-free. The lower bound now follows from Corollary 4.13 as all our other lower bounds in this section. ∎

Optimality of Theorem 7? The bounds obtained by our algorithm for connectivity and cycle-freeness are asymptotically optimal for *any graph* by the reductions in [HP16] and our discussion for Theorem 3. In particular, both problems can be solved in $O(1/\varepsilon)$ passes and $O_\varepsilon(\mathrm{polylog}(n))$ space by simulating the algorithm of [CRT05] in the streaming model. For bipartiteness, we are not aware of a non-trivial algorithm in arbitrary graphs. However, for *planar* graphs, the approach of [HP16] based on the query algorithm of [CMOS11] also immediately implies a tester in $O(1/\varepsilon^2)$ passes and $O_\varepsilon(\mathrm{polylog}(n))$ space for this problem. Bridging this gap remains an open question.

This concludes the list of all lower bounds in Result 2. We remark that for all these problems, no lower bound better than $\Omega(\log(1/\varepsilon))$ passes for $n^{o(1)}$-space algorithms were known even for arbitrary graphs. Our results thus exponentially improves over prior work and in multiple cases lead to asymptotically optimal bounds as discussed above.

6 A Streaming Lower Bound for Pointer Chasing

We present the proof of Proposition 4.12 in this section.

Proposition (Restatement of Proposition 4.12). *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* $s = \Omega\left(\frac{\gamma^4}{b^5} \cdot m\right)$.

Recall that a (m,b)-layered graph consists of $b+1$ layers $V_1, \ldots, V_{b+1}$ of size m each and b matchings $M_1, \ldots, M_b$ where M_i is between V_i and V_{i+1}. We shall consider each matching as a bijection $M_i : V_i \to V_{i+1}$, where for any vertex $v \in V_i$, $M_i(v)$ denotes the matched pair of v in V_{i+1}.

An instance of $\mathbf{PC}_{m,b}$ consists a (m,b)-layered graph, a fixed starting vertex $s \in V_1$, and a fixed equipartition X, Y of V_{b+1}. We define the pointers $P_1(s), \ldots, P_{d+1}(s)$, where $P_i(s)$ is the unique vertex reachable in V_i from s. This way, we have $P_{i+1}(s) = M_i(P_i(s))$. Initially, the pointer $P_1(s)$ is known and the goal is to decide whether $P_{b+1}(s)$ belongs to X or Y.

We prove the lower bound more generally for b-party communication protocols in the following:

(*i*) There are b players $Q_b, \ldots, Q_1$ who receive input matchings $M_b, \ldots, M_1$, respectively.

(*ii*) The players communicate with each other in *rounds* via a *blackboard*. In each round, the players go in turn with Q_b writing a message on the board, followed by Q_{b-1}, all the way to Q_1; these messages are visible to everyone (and are not altered or erased after written).

(*iii*) For any player Q_i and round j, we use Π_i^j to denote the message written on the board by Q_i in the j-th round.

(*iv*) At the end of p rounds, player Q_1 outputs the answer on the board, i.e. the last message Π_1^p is the answer to the problem.

(*v*) The *communication cost* of a protocol is the <u>maximum number of bits</u> communicated by any player in any round.
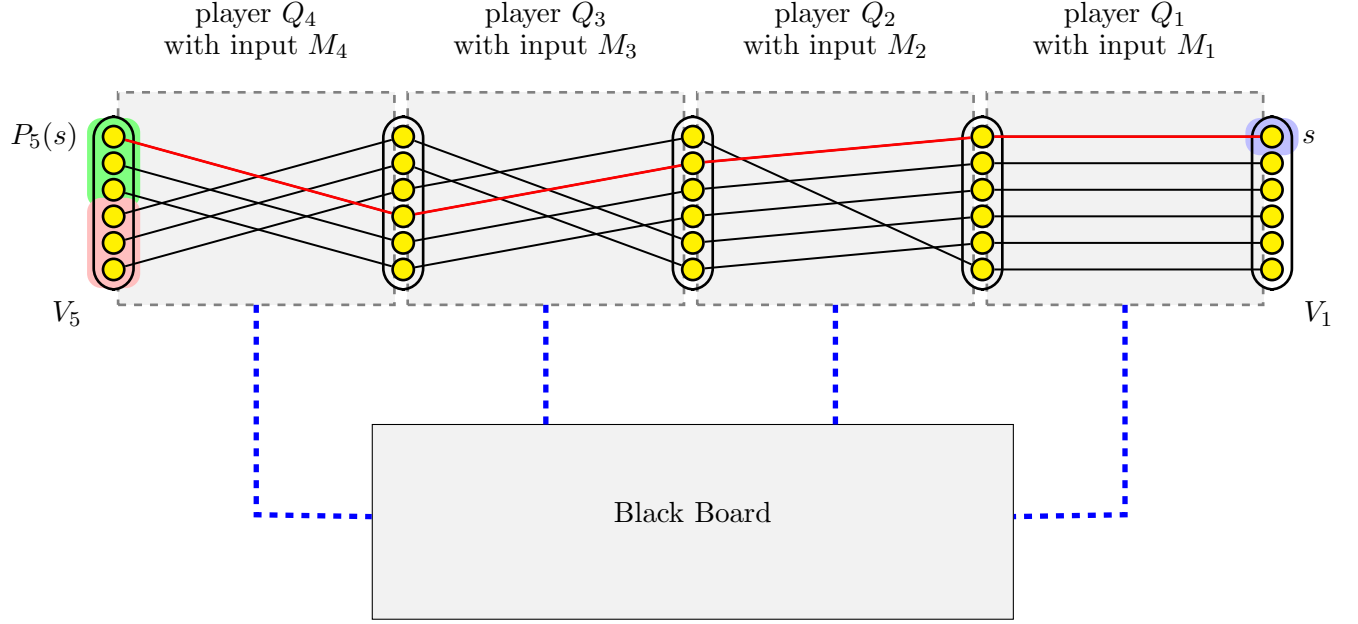

Figure 7: An illustration of the communication game in the proof of Proposition 4.12 for $b = 4$. The $m + 1$ internal players corresponding to the stream M_4 (resp. M_3, M_2, M_1) are merged into a single player Q_4 (resp. Q_3, Q_2, Q_1), who gets the matching M_4 (resp. M_3, M_2, M_1) as input. The dashed (blue) lines draw the communicated messages between the players of the game and the blackboard.

Figure 7 gives an illustration of this b player game. Recall that our Definition 2.1 corresponds to a $(m \cdot b)$-party communication protocol. Considering the input stream in Proposition 4.12 is $M_b \| \cdots \| M_1$, a p-pass s-space streaming algorithm for this stream induces a p-round communication protocol with cost s in the above game via a simulation argument, if we "merge" the m players of each matching M_i in the streaming algorithm into a single player Q_i. In other words, we impose the space limitation only on consecutive players between two different matchings. We prove a lower bound for any p-round communication protocol with cost s that solves the above game over the input distribution[5].

For the rest of the proof, let Π denote a $(b - 1)$-round protocol with probability of success at least $1/2 + \gamma$ over the distribution of inputs. We are going to lower bound the communication cost of this protocol. By the easy direction of Yao's minimax principle (an averaging argument), we can assume Π is deterministic. We use Π^r to denote all the messages communicated in round r, and $\Pi^r_{<i}$, $\Pi^r_{>i}$, and Π^r_{-i} to denote, respectively, the messages communicated by players $Q_1, \ldots, Q_{i-1}$, players $Q_{i+1}, \ldots, Q_b$, and players $Q_1, \ldots, Q_{i-1}, Q_{i+1}, \ldots, Q_b$ in round r. Finally, for any round $r \in [p]$, define:

$$Z_r = (P_1(s), \ldots, P_{r+1}(s), \Pi^1, \ldots, \Pi^r),$$

namely, the information "revealed" to *all* players *after* round r

Throughout the proof, we use sans serif fonts to denote the random variables for the definitions above, e.g., Π^r_i is a random variable for Π^r_i. Also, with a slight abuse of notation, we may use random variables and their distributions interchangeably. We can now start the proof.

---

[5]Note that proving a lower bound in the message passing model is enough to prove a lower bound for a streaming algorithm, yet here we are proving it in the blackboard model for simplicity; this can only strengthen our result.

The main part of the proof is the following inductive lemma. Roughly speaking, it states that the distribution of $(r + 2)$-th pointer after r rounds of communication is close to its original distribution (the protocol is "lagging behind" the pointer it needs to chase – note that the first pointer to chase is $P_2(s)$ as $P_1(s)$ is known globally), even conditioned on the information available to all the players.

Lemma 6.1. *Suppose communication cost of π is $s < \left(\frac{1}{100r} \cdot (\frac{\gamma}{b})^4 \cdot m\right)$, then, for any $r \in [p]$,*

$$\mathop{\mathbb{E}}_{Z_r} \|\mathsf{P}_{r+2}(s) - (\mathsf{P}_{r+2}(s) \mid Z_r)\|_{\mathrm{tvd}} < \gamma \cdot \frac{r}{b}. \tag{7}$$

The proof of this lemma is by induction. The base case for $r = 0$ holds because $Z_0 = P_1(s)$ which is fixed deterministically and thus conditioning on it is irrelevant, and Π^0 is defined to be empty (as there is no communication yet); so both LHS and RHS are zero. Suppose Eq (7) is true up to round r and we prove it for round r itself.

We first show that in round r, the only player that we need to focus on is Q_{r+1}[6]. In other words, we can remove the contribution of all other players in the conditioning of Eq (7).

Claim 6.2. *For any choice of $Z_r = (Z_{r-1}, \Pi^r_{-(r+1)}, \Pi^r_{r+1}, P_{r+1}(s))$,*

$$\mathsf{P}_{r+2}(s) \perp \Pi^r_{-(r+1)} \mid Z_{r-1}, \Pi^r_{r+1}, P_{r+1}(s).$$

Proof. Define $M^-_{r+1} := (M_1, \ldots, M_r, M_{r+2}, \ldots, M_b)$. We have,

$$\mathbb{I}(\mathsf{P}_{r+2}(s); \Pi^r_{-(r+1)} \mid Z_{r-1}, \Pi^r_{r+1}, \mathsf{P}_{r+1}(s)) \leq \mathbb{I}(\mathsf{M}_{r+1}; \mathsf{M}^-_{r+1} \mid Z_{r-1}, \Pi^r_{r+1}, \mathsf{P}_{r+1}(s))$$
(by applying (Fact A.1-(7)) twice as new variables determine old ones (conditionally))
$$\leq \mathbb{I}(\mathsf{M}_{r+1}; \mathsf{M}^-_{r+1}),$$

where the second inequality is a repeated application of Proposition A.4 as each conditioned variable is a function of exactly one side of the mutual information term (conditioned on the remaining ones); this is exactly the same argument as in Lemma 3.3 and we omit the tedious calculations.

We now have $\mathbb{I}(\mathsf{M}_{r+1}; \mathsf{M}^-_{r+1}) = 0$ as in the input, the matchings are chosen independently. This means the first term is also zero, proving the desired independence (by Fact A.1-(2)). $\blacksquare$

By Claim 6.2, we can simplify the LHS of Eq (7) to the following, i.e., get rid of the messages of all players other than Q_{r+1}.

$$\mathop{\mathbb{E}}_{Z_r} \|\mathsf{P}_{r+2}(s) - (\mathsf{P}_{r+2}(s) \mid Z_r)\|_{\mathrm{tvd}} = \mathop{\mathbb{E}}_{\substack{Z_{r-1}, \Pi^r_{r+1}, \\ P_{r+1}(s)}} \|\mathsf{P}_{r+2}(s) - (\mathsf{P}_{r+2}(s) \mid Z_{r-1}, \Pi^r_{r+1}, P_{r+1}(s))\|_{\mathrm{tvd}}. \tag{8}$$

We further simplify the RHS of Eq (8) as follows,

$$\mathop{\mathbb{E}}_{\substack{Z_{r-1}, \Pi^r_{r+1}, \\ P_{r+1}(s)}} \|\mathsf{P}_{r+2}(s) - (\mathsf{P}_{r+2}(s) \mid Z_{r-1}, \Pi^r_{r+1}, P_{r+1}(s))\|_{\mathrm{tvd}}$$

$$= \mathop{\mathbb{E}}_{Z_{r-1}, \Pi^r_{r+1}} \mathop{\mathbb{E}}_{v \sim P_{r+1}(s)|Z_{r-1}, \Pi^r_{r+1}} \|\mathsf{P}_{r+2}(s) - (\mathsf{M}_{r+1}(v) \mid Z_{r-1}, \Pi^r_{r+1}, \mathsf{P}_{r+1}(s) = v)\|_{\mathrm{tvd}}$$

$$(\text{as } P_{r+2}(s) = M_{r+1}(P_{r+1}(s)))$$

---

[6]Of course, player Q_r would reveal the next pointer; the interesting question we need to understand is that whether player Q_{r+1} can also reveal the subsequent pointer in this round.

$$= \mathop{\mathbb{E}}_{Z_{r-1}, \Pi^r_{r+1}} \mathop{\mathbb{E}}_{v \sim \mathsf{P}_{r+1}(s) | Z_{r-1}, \Pi^r_{r+1}} \| \mathsf{P}_{r+2}(s) - (\mathsf{M}_{r+1}(v) \mid Z_{r-1}, \Pi^r_{r+1}) \|_{\mathrm{tvd}} \tag{9}$$

$$= \mathop{\mathbb{E}}_{Z_{r-1}, \Pi^r_{r+1}} \mathop{\mathbb{E}}_{v \sim \mathsf{P}_{r+1}(s) | Z_{r-1}} \| \mathsf{P}_{r+2}(s) - (\mathsf{M}_{r+1}(v) \mid Z_{r-1}, \Pi^r_{r+1}) \|_{\mathrm{tvd}}, \tag{10}$$

where Eq (9) and Eq (10) hold by the following two claims, respectively.

Claim 6.3. *For any choice of Z_{r-1}, Π^r_{r+1} and $v \in V_{r+1}$,*

$$\mathsf{M}_{r+1}(v) \perp (\mathsf{P}_{r+1}(s) = v) \mid Z_{r-1}, \Pi^r_{r+1}.$$

Proof. We have,

$$\mathbb{I}(\mathsf{M}_{r+1}(v) \,;\, \mathsf{P}_{r+1}(s) \mid Z_{r-1}, \Pi^r_{r+1}) \leq \mathbb{I}(\mathsf{M}_{r+1} \,;\, \mathsf{M}_r \mid Z_{r-1}, \Pi^r_{r+1})$$
(by data processing inequality (Fact A.1-(7)) as M_r and Z_{r-1} determine $\mathsf{P}_{r+1}(s)$ (conditionally))
$$\leq \mathbb{I}(\mathsf{M}_{r+1} \,;\, \mathsf{M}_r),$$

where the inequality, and the rest of the proof is exactly as in Claim 6.2. ∎

Claim 6.4. *For any choice of Z_{r-1},*

$$\mathsf{P}_{r+1}(s) \perp \Pi^r_{r+1} \mid Z_{r-1}.$$

Proof. We have,

$$\mathbb{I}(\mathsf{P}_{r+1}(s) \,;\, \Pi^r_{r+1} \mid Z_{r-1}) \leq \mathbb{I}(\mathsf{M}_r \,;\, \mathsf{M}_{r+1} \mid Z_{r-1})$$
(by applying (Fact A.1-(7)) as M_{r+1} determines $\Pi^r_{r+1}(s)$ conditioned on Z_{r-1})
$$\leq \mathbb{I}(\mathsf{M}_{r+1} \,;\, \mathsf{M}_r),$$

where the inequality and the rest of the proof is exactly as in Claim 6.2. ∎

So far, we have proved that,

$$\mathop{\mathbb{E}}_{Z_r} \| \mathsf{P}_{r+2}(s) - (\mathsf{P}_{r+2}(s) \mid Z_r) \|_{\mathrm{tvd}} \leq \mathop{\mathbb{E}}_{Z_{r-1}, \Pi^r_{r+1}} \mathop{\mathbb{E}}_{v \sim \mathsf{P}_{r+1}(s) | Z_{r-1}} \| \mathsf{P}_{r+2}(s) - (\mathsf{M}_{r+1}(v) \mid Z_{r-1}, \Pi^r_{r+1}) \|_{\mathrm{tvd}}. \tag{11}$$

We now apply the induction hypothesis to the expected term of $v \sim \mathsf{P}_{r+1}(s) \mid Z_{r-1}$. In particular, let $\mathcal{U}_{r+1}$ denote the uniform distribution over V_{r+1} which is also $\mathsf{P}_{r+1}(s)$ *without* any conditioning. By Fact A.6, we have that,

$$\mathop{\mathbb{E}}_{Z_{r-1}, \Pi^r_{r+1}} \mathop{\mathbb{E}}_{v \sim \mathsf{P}_{r+1}(s) | Z_{r-1}} \| \mathsf{P}_{r+2}(s) - (\mathsf{M}_{r+1}(v) \mid Z_{r-1}, \Pi^r_{r+1}) \|_{\mathrm{tvd}}$$

$$\leq \mathop{\mathbb{E}}_{Z_{r-1}, \Pi^r_{r+1}} \left[\mathop{\mathbb{E}}_{v \sim \mathcal{U}_{r+1}} \| \mathsf{P}_{r+2}(s) - (\mathsf{M}_{r+1}(v) \mid Z_{r-1}, \Pi^r_{r+1}) \|_{\mathrm{tvd}} + \| \mathcal{U}_{r+1} - (\mathsf{P}_{r+1}(s) \mid Z_{r-1}) \|_{\mathrm{tvd}} \right]$$

$$= \mathop{\mathbb{E}}_{Z_{r-1}, \Pi^r_{r+1}} \mathop{\mathbb{E}}_{v \sim \mathcal{U}_{r+1}} \| \mathsf{P}_{r+2}(s) - (\mathsf{M}_{r+1}(v) \mid Z_{r-1}, \Pi^r_{r+1}) \|_{\mathrm{tvd}} + \gamma \cdot \frac{r-1}{b}, \tag{12}$$

by the induction hypothesis for round $r - 1$ as $\mathcal{U}_{r+1} = \mathsf{P}_{r+1}(s)$.

This conclude the first half of the proof of Lemma 6.1. In the remaining half, we bound the first term in the RHS of Eq (12). It is worth mentioning that in the first part, we did not deal with the fact that input of each player is a random matching as opposed to a random function. So, this

part is more or less an extension of two-party approaches for pointer chasing, say [NW91, Yeh16], to the multi-party setting. From here however, we depart from the prior approaches to take into account the distribution of each M_i being a random matching as opposed to a random function.

The following claim shows that we can further narrow down the task to player Q_{r+1} by removing everything on the RHS that is not function of this player even from prior rounds. Formally,

Claim 6.5. *For any choice of* $(\Pi^1_{r+1}, \ldots, \Pi^r_{r+1})$ *(messages of Q_{r+1} so far), and any $v \in V_{r+1}$,*

$$\mathsf{M}_{r+1}(v) \perp \mathsf{Z}_{r-1} \mid (\Pi^1_{r+1}, \ldots, \Pi^r_{r+1}).$$

Proof. We have,

$$\mathbb{I}(\mathsf{M}_{r+1}(v) ; \mathsf{Z}_{r-1} \mid \Pi^1_{r+1}, \ldots, \Pi^r_{r+1}) = \mathbb{I}(\mathsf{M}_{r+1}(v) ; \Pi^1_{-(r+1)}, \ldots, \Pi^r_{-(r+1)}, \mathsf{P}_1(s), \ldots, \mathsf{P}_r(s) \mid \Pi^1_{r+1}, \ldots, \Pi^r_{r+1})$$

$$\text{(these are the remaining variables in } \mathsf{Z}_{r-1} \text{ after conditioning)}$$

$$\leq \mathbb{I}(\mathsf{M}_{r+1} ; \mathsf{M}_{-(r+1)} \mid \Pi^1_{r+1}, \ldots, \Pi^r_{r+1})$$

(by data processing inequality (Fact A.1-(7)) as new variables determine old ones)

$$\leq \mathbb{I}(\mathsf{M}_{r+1} ; \mathsf{M}_{-(r+1)} \mid \Pi^1_{>r+1}, \Pi^1_{r+1}, \ldots, \Pi^r_{r+1})$$

(by Proposition A.3 as $\mathsf{M}_{r+1} \perp \Pi^1_{>r+1}$ by the independence of players' inputs)

$$\leq \mathbb{I}(\mathsf{M}_{r+1} ; \mathsf{M}_{-(r+1)} \mid \Pi^1_{<r+1}, \Pi^1_{>r+1}, \Pi^1_{r+1}, \ldots, \Pi^r_{r+1})$$

(by Proposition A.3 as $\mathsf{M}_{r+1} \perp \Pi^1_{<r+1} \mid \Pi^1_{>r+1}, \Pi^1_{r+1}$ by the independence of players' inputs)

$$= \mathbb{I}(\mathsf{M}_{r+1} ; \mathsf{M}_{-(r+1)} \mid \Pi^1, \Pi^2_{r+1}, \ldots, \Pi^r_{r+1})$$

$$\left(\text{as } \Pi^1_{<r+1}, \Pi^1_{>r+1}, \Pi^1_{r+1} = \Pi^1\right)$$

$$\leq \mathbb{I}(\mathsf{M}_{r+1} ; \mathsf{M}_{-(r+1)} \mid \Pi^1, \ldots, \Pi^r)$$

(using exactly the same argument as above)

$$\leq \mathbb{I}(\mathsf{M}_{r+1} ; \mathsf{M}_{-(r+1)}),$$

where the inequality and the rest of the proof is exactly as in Claim 6.2. ∎

By Claim 6.5, we can simplify the first term of RHS of Eq (12) as follows:

$$\underset{\mathsf{Z}_{r-1}, \Pi^r_{r+1}}{\mathbb{E}} \underset{v \sim \mathcal{U}_{r+1}}{\mathbb{E}} \|\mathsf{P}_{r+2}(s) - (\mathsf{M}_{r+1}(v) \mid \mathsf{Z}_{r-1}, \Pi^r_{r+1})\|_{\text{tvd}}$$

$$= \underset{\mathsf{Z}_{r-1}, \Pi^r_{r+1}}{\mathbb{E}} \underset{v \sim \mathcal{U}_{r+1}}{\mathbb{E}} \|\mathsf{P}_{r+2}(s) - (\mathsf{M}_{r+1}(v) \mid \Pi^1_{r+1}, \ldots, \Pi^r_{r+1})\|_{\text{tvd}}$$

$$= \underset{\Pi^1_{r+1}, \ldots, \Pi^r_{r+1}}{\mathbb{E}} \underset{v \sim \mathcal{U}_{r+1}}{\mathbb{E}} \|\mathsf{P}_{r+2}(s) - (\mathsf{M}_{r+1}(v) \mid \Pi^1_{r+1}, \ldots, \Pi^r_{r+1})\|_{\text{tvd}}.$$

Notice that now the RHS of the above term is purely a function of player Q_{r+1}. Abstractly, the question is to understand how much Q_{r+1} can change the distribution of a *random* edge in their input by their messages in the first r rounds. Intuitively, this should not be much if the size of messages are small. We now formalize this in the following lemma.

Lemma 6.6. *Assuming communication cost of π is $s < \left(\frac{1}{100r} \cdot (\frac{\gamma}{b})^4 \cdot m\right)$,*

$$\underset{\Pi^1_{r+1}, \ldots, \Pi^r_{r+1}}{\mathbb{E}} \underset{v \sim \mathcal{U}_{r+1}}{\mathbb{E}} \|\mathsf{P}_{r+2}(s) - (\mathsf{M}_{r+1}(v) \mid \Pi^1_{r+1}, \ldots, \Pi^r_{r+1})\|_{\text{tvd}} < \gamma \cdot \frac{1}{b}.$$

33

Proof. To avoid the clutter, throughout this proof (only), we are going to drop all the subscripts and superscripts on r and in particular denote:

$$Q := Q_{r+1} \qquad \text{(the player } Q_{r+1} \text{ we are focusing on)}$$
$$\Pi := \Pi_{r+1}^1, \ldots, \Pi_{r+1}^r \qquad \text{(the messages of player } Q_{r+1}\text{)}$$
$$M := M_{r+1} \qquad \text{(the input of player } Q_{r+1} \text{ which is a random matching)}$$
$$D := V_{r+1} \qquad \text{(domain of } M_{r+1} \text{ interpreted as a function)}$$
$$R := V_{r+2} \qquad \text{(range of } M_{r+1} \text{ interpreted as a function)}$$
$$\mathcal{U}_D = \mathcal{U}_{r+1} \qquad \text{(uniform distribution on the domain } D\text{)}$$
$$\mathcal{U}_R = \mathsf{P}_{r+2}(s) \qquad \text{(uniform distribution on the range } R\text{)}.$$

Thus, with this notation, our goal is to prove that

$$\mathop{\mathbb{E}}_{\Pi} \mathop{\mathbb{E}}_{v \sim \mathcal{U}_D} \| \mathcal{U}_R - (\mathsf{M}(v) \mid \Pi) \|_{\mathrm{tvd}} \leq \gamma \cdot \frac{1}{b}.$$

Firstly, by Pinsker's inequality (Fact A.8),

$$\mathop{\mathbb{E}}_{\Pi} \mathop{\mathbb{E}}_{v \sim \mathcal{U}_D} \| \mathcal{U}_R - (\mathsf{M}(v) \mid \Pi) \|_{\mathrm{tvd}} = \mathop{\mathbb{E}}_{\Pi} \mathop{\mathbb{E}}_{v \sim \mathcal{U}_D} \| (\mathsf{M}(v) \mid \Pi) - \mathsf{M}(v) \|_{\mathrm{tvd}}$$

$$\qquad \text{(symmetry of } \| - \|_{\mathrm{tvd}} \text{ and since unconditional distribution of } \mathsf{M}(v) = \mathcal{U}_R \text{ for all } v)$$

$$\leq \mathop{\mathbb{E}}_{\Pi} \mathop{\mathbb{E}}_{v \sim \mathcal{U}_D} \sqrt{\frac{1}{2} \cdot \mathbb{D}(\mathsf{M}(v) \mid \Pi \ || \ \mathsf{M}(v))} \qquad \text{(by Fact A.8)}$$

$$\leq \sqrt{\frac{1}{2} \mathop{\mathbb{E}}_{\Pi} \mathop{\mathbb{E}}_{v \sim \mathcal{U}_D} \mathbb{D}(\mathsf{M}(v) \mid \Pi \ || \ \mathsf{M}(v))} \qquad \text{(by concavity of } \sqrt{\cdot} \text{ and Jensen's inequality)}$$

$$= \sqrt{\frac{1}{2} \cdot \frac{1}{|D|} \cdot \sum_{v \in D} \mathop{\mathbb{E}}_{\Pi} \mathbb{D}(\mathsf{M}(v) \mid \Pi \ || \ \mathsf{M}(v))} \qquad \text{(as } \mathcal{U}_D \text{ is uniform distribution over } D)$$

$$= \sqrt{\frac{1}{2} \cdot \frac{1}{|D|} \cdot \sum_{v \in D} \mathbb{I}(\mathsf{M}(v) \, ; \Pi)}, \tag{13}$$

where the last inequality is by Fact A.5. We thus need to bound the mutual information in the RHS above. For any $v \in D$,

$$\mathbb{I}(\mathsf{M}(v) \, ; \Pi) = \mathbb{H}(\mathsf{M}(v)) - \mathbb{H}(\mathsf{M}(v) \mid \Pi) = \log |D| - \mathbb{H}(\mathsf{M}(v) \mid \Pi), \tag{14}$$

as $\mathsf{M}(v)$ is uniform over D and thus we can apply Fact A.1-(1). Let us take a quick detour that would help put the rest of the argument in some context.

Remark. *It is tempting now to apply the sub-additivity of entropy (Fact A.1-(4)) to have,*

$$\sum_v \mathbb{H}(\mathsf{M}(v) \mid \Pi) \geq \mathbb{H}(\mathsf{M} \mid \Pi) \geq \mathbb{H}(\mathsf{M}) - \mathbb{H}(\Pi);$$

the problem with this approach is that M is a random matching and not a random function and thus $\mathbb{H}(\mathsf{M}) = \log(|D|!) = |D| \log |D| - \Theta(|D|)$ by Stirling's approximation. This means that ignoring even subtraction of $\mathbb{H}(\Pi)$, which means even if Q does not communicate(!), the bound we get on the average entropy is $\log |D| - \Theta(1)$; plugging this in Eq (14) will bound the information term by $\Theta(1)$ which is too much (and quite loose as without Π, this term should be zero).

34

We will use Entropy Subset Inequality[7] of Fact A.2 to bound Eq (14). Let $\beta \leq |D|/2$ be an integer to be determined later. For any $S \subseteq D$, define $\mathsf{M}_S := \{\mathsf{M}(v) \mid v \in S\}$. By Entropy Subset Inequality (Fact A.2),

$$\frac{1}{|D|} \sum_{v \in D} \mathbb{H}(\mathsf{M}(v) \mid \Pi_r) \geq \frac{1}{\binom{|D|}{\beta}} \sum_{S \subseteq D : |S| = \beta} \frac{\mathbb{H}(\mathsf{M}_S \mid \Pi)}{\beta}$$

$$\geq \frac{1}{\binom{|D|}{\beta}} \sum_{S \subseteq D : |S| = \beta} \frac{\mathbb{H}(\mathsf{M}_S) - \mathbb{H}(\Pi)}{\beta}$$

$$\text{(by the chain rule of entropy (Fact A.1-(5)))}$$

$$= \frac{\log\left(\frac{|D|!}{(|D|-\beta)!}\right) - \mathbb{H}(\Pi)}{\beta}$$

$$\text{(because } \mathsf{M}_S \text{ is a uniform partial matching with one endpoint in } S\text{)}$$

$$\geq \log(|D| - \beta) - \frac{\mathbb{H}(\Pi)}{\beta} \qquad \left(\text{as } \frac{a!}{(a-b)!} \geq (a-b)^b\right)$$

$$\geq \log(|D| - \beta) - \frac{r \cdot s}{\beta}.$$

$$\text{(as } \Pi \text{ consists of } r \text{ messages of size } s \text{ bits each and by using Fact A.1-(1))}$$

The rest of the proof is simply calculations. By plugging this bound in Eq (14), we have,

$$\frac{1}{|D|} \cdot \sum_{v \in D} \mathbb{I}(\mathsf{M}_r(v) ; \Pi) \leq \log|D| - \log(|D| - \beta) + \frac{r \cdot s}{\beta}$$

$$= \log\left(1 + \frac{\beta}{|D| - \beta}\right) + \frac{r \cdot s}{\beta}$$

$$\leq \frac{\beta}{|D| - \beta} \cdot \log(e) + \frac{r \cdot s}{\beta} \qquad (\text{as } 1 + x \leq e^x)$$

$$\leq \frac{4\beta}{|D|} + \frac{r \cdot s}{\beta}. \qquad (\text{as } \beta \leq |D|/2 \text{ and } \log(e) \leq 2)$$

We can now set $\beta = \sqrt{r \cdot s \cdot |D|} < |D|/2$ and obtain that

$$\frac{1}{|D|} \cdot \sum_{v \in D} \mathbb{I}(\mathsf{M}_r(v) ; \Pi) \leq 5\sqrt{\frac{r \cdot s}{|D|}}.$$

Plugging in further in Eq (13) proves that,

$$\mathop{\mathbb{E}}_{\Pi} \mathop{\mathbb{E}}_{v \sim \mathcal{U}_D} \|\mathcal{U}_R - (\mathsf{M}(v) \mid \Pi)\|_{\text{tvd}} \leq \sqrt{\frac{1}{2} \cdot 5\sqrt{\frac{r \cdot s}{|D|}}} < \gamma \cdot \frac{1}{b},$$

as $s < \left(\frac{1}{100r} \cdot \left(\frac{\gamma}{b}\right)^4 \cdot m\right)$ and $|D| = m$. This concludes the proof. ∎ Lemma 6.6

By using Lemma 6.6 and Claim 6.5 in Eq (12), we obtain that,

$$\mathop{\mathbb{E}}_{Z_{r-1}, \Pi_{r+1}^r} \mathop{\mathbb{E}}_{v \sim \mathsf{P}_{r+1}(s) \mid Z_{r-1}} \|\mathsf{P}_{r+2}(s) - (\mathsf{M}_{r+1}(v) \mid Z_{r-1}, \Pi_{r+1}^r)\|_{\text{tvd}} < \gamma \cdot \frac{1}{b} + \gamma \cdot \frac{r-1}{b} = \gamma \cdot \frac{r}{b}.$$

---

[7]We note that this part of the proof can also be done using a similar argument in [AKSY20] on high-entropy random permutations, although we found the current approach simpler and more direct.

Plugging this bound into Eq (11), we get that,

$$\underset{Z_r}{\mathbb{E}} \, \|\mathsf{P}_{r+2}(s) - (\mathsf{P}_{r+2}(s) \mid Z_r)\|_{\mathrm{tvd}} < \gamma \cdot \frac{r}{b},$$

finalizing the proof of the induction step and thus Lemma 6.1.

We now conclude the proof of Proposition 4.12 as follows.

Proof of Proposition 4.12. If $p > b - 1$ we are already done so let us assume that the total number of rounds of the protocol (or passes of the streaming algorithm) is $p = b - 1$. By Lemma 6.1 for the last round $r = p = b - 1$,

$$\underset{Z_p}{\mathbb{E}} \, \|\mathsf{P}_{b+1}(s) - (\mathsf{P}_{b+1}(s) \mid Z_p)\|_{\mathrm{tvd}} < \gamma.$$

On the other hand, the last message of the protocol π, included in Z_p, specifies the answer, which depends on P_{b+1}. Let $O(Z_p) \in \{X, Y\}$ denote the answer of the protocol. As such,

$$
\begin{aligned}
\Pr\left(\pi \text{ is correct}\right) &= \underset{Z_p}{\mathbb{E}} \, \underset{P_{b+1}|Z_p}{\Pr} \, (P_{d+1}(s) \in O(Z_p)) \\
&= \underset{Z_p}{\mathbb{E}} \, \underset{P_{b+1}|Z_p}{\Pr} \, (P_{b+1}(s) \in \text{ a fixed choice of } X \text{ or } Y) \\
&\qquad\qquad\qquad\qquad\qquad \text{(because } O(Z_p) \text{ is deterministically fixed)} \\
&\leq \underset{Z_p}{\mathbb{E}} \left[\underset{P_{b+1}}{\Pr} \, (P_{b+1}(s) \in \text{ a fixed choice of } X \text{ or } Y) + \|\mathsf{P}_{b+1}(s) - (\mathsf{P}_{b+1}(s) \mid Z_p)\|_{\mathrm{tvd}} \right] \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(by Fact A.6)} \\
&= \frac{1}{2} + \underset{Z_p}{\mathbb{E}} \, \|\mathsf{P}_{b+1}(s) - (\mathsf{P}_{b+1}(s) \mid Z_p)\|_{\mathrm{tvd}} \\
&\qquad\qquad \text{(as } X, Y \text{ is an equipartition of } V_{b+1} \text{ and } P_{b+1}(s) \text{ is uniform over } V_{b+1}) \\
&< \frac{1}{2} + \gamma,
\end{aligned}
$$

by the first equation above. This concludes the proof. ∎

Acknowledgement

References

[ACK19]   Sepehr Assadi, Yu Chen, and Sanjeev Khanna. 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.*, pages 265–276, 2019. 4

[AG09]   Kook Jin Ahn and Sudipto Guha. Graph sparsification in the semi-streaming model. In *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II*, pages 328–338, 2009. 27

[AGM12]   Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 459–467. SIAM, 2012. 23

[AKL17]   Sepehr Assadi, Sanjeev Khanna, and Yang Li. 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*, pages 1723–1742, 2017. 1, 2, 25

[AKSY20]   Sepehr Assadi, Gillat Kol, Raghuvansh R. Saxena, and Huacheng Yu. 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. ii, 1, 2, 3, 22, 23, 25, 26, 27, 28, 35, 36, 49, 50

[BC17]   Suman K. Bera and Amit Chakrabarti. 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*, pages 11:1–11:14, 2017. 1

[BCK+18]   Vladimir Braverman, Stephen R. Chestnut, Robert Krauthgamer, Yi Li, David P. Woodruff, and Lin F. Yang. 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*, pages 648–657, 2018. 2

[BDV18]   Aditya Bhaskara, Samira Daruki, and Suresh Venkatasubramanian. 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*, pages 16:1–16:14, 2018. 1, 26, 27

[BFKP16]   Laurent Bulteau, Vincent Froese, Konstantin Kutzkov, and Rasmus Pagh. Triangle counting in dynamic graph streams. *Algorithmica*, 76(1):259–278, 2016. 1

[BGGS19]   Mitali Bafna, Badih Ghazi, Noah Golowich, and Madhu Sudan. 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*, pages 1861–1871, 2019. 4

[BGW20]   Mark Braverman, Sumegha Garg, and David P. Woodruff. The coin problem with applications to data streams. *Electron. Colloquium Comput. Complex.*, 27. To appear in FOCS 2020, 2020. 4

[BJKS02]   Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. 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*, pages 209–218, 2002. 4

[BKLS20]   Joshua Brody, Jae Tak Kim, Peem Lerdputtipongporn, and Hariharan Srinivasulu. A strong XOR lemma for randomized query complexity. *CoRR*, abs/2007.05580, 2020. 4, 47

[BKS02]   Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. 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.*, pages 623–632, 2002. 1

[BOV13]   Vladimir Braverman, Rafail Ostrovsky, and Dan Vilenchik. 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*, pages 244–254, 2013. 1

[BRWY13]   Mark Braverman, Anup Rao, Omri Weinstein, and Amir Yehudayoff. Direct products in communication complexity. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013*, pages 746–755, 2013. 4

[BS15]   Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, September 14-16, 2015, Proceedings*, pages 263–274, 2015. 2, 3, 25

[CCE+16]   Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, January 10-12, 2016*, pages 1326–1344, 2016. 1, 25

[CCM08]   Amit Chakrabarti, Graham Cormode, and Andrew McGregor. Robust lower bounds for communication and stream computation. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, May 17-20, 2008*, pages 641–650, 2008. 4

[CFPS20]   Artur Czumaj, Hendrik Fichtenberger, Pan Peng, and Christian Sohler. 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*, pages 16:1–16:20, 2020. 1, 3, 28

[CGMV20]   Amit Chakrabarti, Prantar Ghosh, Andrew McGregor, and Sofya Vorotnikova. Vertex ordering problems in directed graph streams. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1786–1802, 2020. 27

[CGV20]   Chi-Ning Chou, Alexander Golovnev, and Santhoshini Velusamy. Optimal streaming approximations for all boolean max-2csps. *CoRR*, abs/2004.11796. To appear in FOCS 2020, 2020. 1, 2, 3, 27

[CJ17]   Graham Cormode and Hossein Jowhari. A second look at counting triangles in graph streams (corrected). *Theor. Comput. Sci.*, 683:22–30, 2017. 1

[CJMM17]   Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and S. Muthukrishnan. 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*, pages 29:1–29:15, 2017. 1, 25

[CMOS11] Artur Czumaj, Morteza Monemizadeh, Krzysztof Onak, and Christian Sohler. Planar graphs: Random walks and bipartiteness testing. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 423–432, 2011. 29

[CRT05] Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, 34(6):1370–1379, 2005. 1, 24, 29

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006. 44

[EHL+15] Hossein Esfandiari, Mohammad Taghi Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, January 4-6, 2015*, pages 1217–1233, 2015. 1, 2, 3, 25

[FKM+05] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. 3

[FKM+08] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the data-stream model. *SIAM J. Comput.*, 38(5):1709–1727, 2008. 4, 23

[GH09] Sudipto Guha and Zhiyi Huang. 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*, pages 513–524, 2009. 4

[GKK+07] Dmitry Gavinsky, Julia Kempe, Iordanis Kerenidis, Ran Raz, and Ronald de Wolf. Exponential separations for one-way quantum communication complexity, with applications to cryptography. *STOC*, pages 516–525, 2007. 1

[GM08] Sudipto Guha and Andrew McGregor. 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*, pages 760–772, 2008. 4, 47

[GM09] Sudipto Guha and Andrew McGregor. Stream order and order statistics: Quantile estimation in random-order streams. *SIAM J. Comput.*, 38(5):2044–2059, 2009. 4, 15, 20, 21

[GNW11] Oded Goldreich, Noam Nisan, and Avi Wigderson. 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*, pages 273–301. 2011. 3

[GO13]   Venkatesan Guruswami and Krzysztof Onak. 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*, pages 287–298, 2013. 4

[GRZ20]   Uma Girish, Ran Raz, and Wei Zhan. Lower bounds for XOR of forrelations. *CoRR*, abs/2007.03631, 2020. 3, 4

[GS20]   Noah Golowich and Madhu Sudan. 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*, pages 1076–1095, 2020. 4

[GT19]   Venkatesan Guruswami and Runzhou Tao. 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*, pages 5:1–5:12, 2019. 1, 2, 27

[GVV17]   Venkatesan Guruswami, Ameya Velingker, and Santhoshini Velusamy. 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*, pages 8:1–8:19, 2017. 1, 2, 3, 27

[Had75]   F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM J. Comput.*, pages 221–225, 1975. 27

[HP16]   Zengfeng Huang and Pan Peng. 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*, pages 18:1–18:16, 2016. 1, 2, 3, 23, 24, 28, 29

[Imp95]   Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 538–545, 1995. 3

[IW97]   Russell Impagliazzo and Avi Wigderson. $P = BPP$ 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*, pages 220–229, 1997. 3

[JPY12]   Rahul Jain, Attila Pereszlényi, and Penghui Yao. 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*, pages 167–176, 2012. 4

[JRS03]   Rahul Jain, Jaikumar Radhakrishnan, and Pranab Sen. 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*, pages 300–315, 2003. 4

[KK15]   Dmitry Kogan and Robert Krauthgamer. 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*, pages 367–376, 2015. 1, 2, 3, 26

[KK19]   Michael Kapralov and Dmitry Krachun. An optimal space lower bound for approx-
            imating MAX-CUT. In *Proceedings of the 51st Annual ACM SIGACT Symposium
            on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages
            277–288, 2019. 1, 2, 26

[KKP18]   John Kallaugher, Michael Kapralov, and Eric Price. The sketching complexity of graph
            and hypergraph counting. In *59th IEEE Annual Symposium on Foundations of Com-
            puter Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 556–567, 2018.
            2

[KKS14]   Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size
            from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Sympo-
            sium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*,
            pages 734–751, 2014. 1, 3, 25

[KKS15]   Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. 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*, pages 1263–1282, 2015. 1, 2, 3, 26

[KKSV17]   Michael Kapralov, Sanjeev Khanna, Madhu Sudan, and Ameya Velingker. $(1 + \Omega(1))$-
            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*, pages 1703–1722, 2017. 1, 2, 26

[KMNT20]   Michael Kapralov, Slobodan Mitrovic, Ashkan Norouzi-Fard, and Jakab Tardos. 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*, pages 1753–1772, 2020. 3, 25

[KMPV19]   John Kallaugher, Andrew McGregor, Eric Price, and Sofya Vorotnikova. The com-
            plexity 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*, pages 119–133,
            2019. 1

[KN97]   Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University
            Press, 1997. 5, 47

[Lev85]   Leonid A. Levin. 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*, pages 363–365, 1985. 3

[LNW14]   Yi Li, Huy L. Nguyen, and David P. Woodruff. Turnstile streaming algorithms might
            as well be linear sketches. In *Symposium on Theory of Computing, STOC 2014, New
            York, NY, USA, May 31 - June 03, 2014*, pages 174–183, 2014. 4

[Lov79]   László Lovász. On determinants, matchings, and random algorithms. In *Fundamen-
            tals of Computation Theory, FCT 1979, Proceedings of the Conference on Algebraic,
            Arthmetic, and Categorial Methods in Computation Theory, Berlin/Wendisch-Rietz,
            Germany, September 17-21, 1979*, pages 565–574, 1979. 25

[LW16]   Yi Li and David P. Woodruff. 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*, pages 726–739, 2016. 2, 25

[MMPS17]   Morteza Monemizadeh, S. Muthukrishnan, Pan Peng, and Christian Sohler. 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*, pages 131:1–131:14, 2017. 1, 3, 24, 25, 26, 28

[MV16]   Andrew McGregor and Sofya Vorotnikova. Planar matching in streams revisited. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016*, pages 17:1–17:12, 2016. 1

[MV18]   Andrew McGregor and Sofya Vorotnikova. 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*, pages 14:1–14:4, 2018. 1, 25

[MVV16]   Andrew McGregor, Sofya Vorotnikova, and Hoa T. Vu. 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*, pages 401–411, 2016. 1

[MWY13]   Marco Molinaro, David P. Woodruff, and Grigory Yaroslavtsev. 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*, pages 1738–1756, 2013. 4

[NO08]   Huy N. Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 327–336, 2008. 26

[NW91]   Noam Nisan and Avi Wigderson. 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*, pages 419–429, 1991. 4, 15, 20, 21, 33

[PRV99]   Stephen Ponzio, Jaikumar Radhakrishnan, and Srinivasan Venkatesh. 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*, pages 602–611, 1999. 4, 15, 20, 21

[PS18]   Pan Peng and Christian Sohler. 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*, pages 2449–2466, 2018. 1, 3, 28

[PVZ12]   Jeff M. Phillips, Elad Verbin, and Qin Zhang. 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*, pages 486–501, 2012. 4

[RS16]   Anup Rao and Makrand Sinha. 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*, pages 44:1–44:15, 2016. 4

[Sha03]   Ronen Shaltiel. Towards proving strong direct product theorems. *Comput. Complex.*, 12(1-2):1–22, 2003. 4, 48

[She11]   Alexander A. Sherstov. 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*, pages 41–50, 2011. 4

[TS78]   H Te Sun. Nonnegative entropy measures of multivariate symmetric correlations. 1978. 45

[Tut47]   William T Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 1(2):107–111, 1947. 25

[VW08]   Emanuele Viola and Avi Wigderson. Norms, XOR lemmas, and lower bounds for polynomials and protocols. *Theory Comput.*, 4(1):137–168, 2008. 4

[VY11]   Elad Verbin and Wei Yu. 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*, pages 11–25, 2011. 1, 2

[Wei15]   Omri Weinstein. Information complexity and the quest for interactive compression. *SIGACT News*, 46(2):41–64, 2015. 50

[Yao82]   Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91, 1982. 1, 3

[Yeh16]   Amir Yehudayoff. Pointer chasing via triangular discrimination. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:151, 2016. 4, 15, 20, 21, 33, 47

Appendix

A   Basic Tools From Information Theory

We now briefly introduce some definitions and facts from information theory that are needed in this paper. We refer the interested reader to the text by Cover and Thomas [CT06] for an excellent introduction to this field.

For a random variable A, we use $\mathrm{supp}(\mathsf{A})$ to denote the support of A and $\mathrm{dist}(\mathsf{A})$ to denote its distribution. When it is clear from the context, we may abuse the notation and use A directly instead of $\mathrm{dist}(\mathsf{A})$, for example, write $A \sim \mathsf{A}$ to mean $A \sim \mathrm{dist}(\mathsf{A})$, i.e., A is sampled from the distribution of random variable A.

We denote the *Shannon Entropy* of a random variable A by $\mathbb{H}(\mathsf{A})$, which is defined as:

$$\mathbb{H}(\mathsf{A}) := \sum_{A \in \mathrm{supp}(\mathsf{A})} \Pr\left(\mathsf{A} = A\right) \cdot \log\left(1/\Pr\left(\mathsf{A} = A\right)\right) \tag{15}$$

The *conditional entropy* of A conditioned on B is denoted by $\mathbb{H}(\mathsf{A} \mid \mathsf{B})$ and defined as:

$$\mathbb{H}(\mathsf{A} \mid \mathsf{B}) := \underset{B \sim \mathsf{B}}{\mathbb{E}} \left[\mathbb{H}(\mathsf{A} \mid \mathsf{B} = B)\right], \tag{16}$$

where $\mathbb{H}(\mathsf{A} \mid \mathsf{B} = B)$ is defined in a standard way by using the distribution of A conditioned on the event $\mathsf{B} = B$ in Eq (15).

The *mutual information* of two random variables A and B is denoted by $\mathbb{I}(\mathsf{A} \,;\mathsf{B})$ and defined as:

$$\mathbb{I}(\mathsf{A} \,;\mathsf{B}) := \mathbb{H}(A) - \mathbb{H}(A \mid B) = \mathbb{H}(B) - \mathbb{H}(B \mid A). \tag{17}$$

The *conditional mutual information* $\mathbb{I}(\mathsf{A} \,;\mathsf{B} \mid \mathsf{C})$ is $\mathbb{H}(\mathsf{A} \mid \mathsf{C}) - \mathbb{H}(\mathsf{A} \mid \mathsf{B},\mathsf{C})$ and hence by linearity of expectation:

$$\mathbb{I}(\mathsf{A} \,;\mathsf{B} \mid \mathsf{C}) = \underset{C \sim \mathsf{C}}{\mathbb{E}} \left[\mathbb{I}(\mathsf{A} \,;\mathsf{B} \mid \mathsf{C} = C)\right]. \tag{18}$$

A.1   Useful Properties of Entropy and Mutual Information

We use the following basic properties of entropy and mutual information throughout.

Fact A.1 (cf. [CT06])**.** *Let* A*,* B*,* C*, and* D *be four (possibly correlated) random variables.*

1. $0 \le \mathbb{H}(\mathsf{A}) \le \log |\mathrm{supp}(\mathsf{A})|$*. The right equality holds iff* $\mathrm{dist}(\mathsf{A})$ *is uniform.*

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

3. Conditioning on a random variable reduces entropy*:* $\mathbb{H}(\mathsf{A} \mid \mathsf{B},\mathsf{C}) \le \mathbb{H}(\mathsf{A} \mid \mathsf{B})$*. The equality holds iff* $\mathsf{A} \perp \mathsf{C} \mid \mathsf{B}$*.*

4. Subadditivity of entropy*:* $\mathbb{H}(\mathsf{A},\mathsf{B} \mid \mathsf{C}) \le \mathbb{H}(\mathsf{A} \mid C) + \mathbb{H}(\mathsf{B} \mid \mathsf{C})$*.*

5. Chain rule for entropy*:* $\mathbb{H}(\mathsf{A},\mathsf{B} \mid \mathsf{C}) = \mathbb{H}(\mathsf{A} \mid \mathsf{C}) + \mathbb{H}(\mathsf{B} \mid \mathsf{C},\mathsf{A})$*.*

6. Chain rule for mutual information*:* $\mathbb{I}(\mathsf{A},\mathsf{B} \,;\mathsf{C} \mid \mathsf{D}) = \mathbb{I}(\mathsf{A} \,;\mathsf{C} \mid \mathsf{D}) + \mathbb{I}(\mathsf{B} \,;\mathsf{C} \mid \mathsf{A},\mathsf{D})$*.*

7. Data processing inequality*: for a deterministic function* $f(\mathsf{A})$*,* $\mathbb{I}(f(\mathsf{A}) \,;\mathsf{B} \mid \mathsf{C}) \le \mathbb{I}(\mathsf{A} \,;\mathsf{B} \mid \mathsf{C})$*.*

We also use the following generalization of sub-additivity of entropy.

Fact A.2 (Entropy Subset Inequality [TS78])**.** *For any set of n random variables* $\mathsf{X}_1, \ldots, \mathsf{X}_n$ *and set* $S \subseteq [n]$, *we define* $\mathsf{X}_S := \{\mathsf{X}_i \mid i \in S\}$. *For every* $k \in [n]$, *define:*

$$\mathbb{H}^{(k)}(\mathsf{X}) := \frac{1}{\binom{n}{k}} \sum_{S \subseteq [n]:|S|=k} \frac{\mathbb{H}(\mathsf{X}_S)}{k}.$$

Then, $\mathbb{H}^{(1)}(\mathsf{X}) \geq \cdots \geq \mathbb{H}^{(n)}(\mathsf{X})$. *This equation also holds for conditional entropy.*

We also use the following two standard propositions, regarding the effect of conditioning on mutual information.

Proposition A.3. *For random variables* $\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}$, *if* $\mathsf{A} \perp \mathsf{D} \mid \mathsf{C}$, *then,*

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

Proof. Since A and D are independent conditioned on C, by Fact A.1-(3), $\mathbb{H}(\mathsf{A} \mid \mathsf{C}) = \mathbb{H}(\mathsf{A} \mid \mathsf{C}, \mathsf{D})$ and $\mathbb{H}(\mathsf{A} \mid \mathsf{C}, \mathsf{B}) \geq \mathbb{H}(\mathsf{A} \mid \mathsf{C}, \mathsf{B}, \mathsf{D})$. We have,

$$\mathbb{I}(\mathsf{A} \,;\mathsf{B} \mid \mathsf{C}) = \mathbb{H}(\mathsf{A} \mid \mathsf{C}) - \mathbb{H}(\mathsf{A} \mid \mathsf{C}, \mathsf{B}) = \mathbb{H}(\mathsf{A} \mid \mathsf{C}, \mathsf{D}) - \mathbb{H}(\mathsf{A} \mid \mathsf{C}, \mathsf{B})$$
$$\leq \mathbb{H}(\mathsf{A} \mid \mathsf{C}, \mathsf{D}) - \mathbb{H}(\mathsf{A} \mid \mathsf{C}, \mathsf{B}, \mathsf{D}) = \mathbb{I}(\mathsf{A} \,;\mathsf{B} \mid \mathsf{C}, \mathsf{D}). \quad \blacksquare$$

Proposition A.4. *For random variables* $\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}$, *if* $\mathsf{A} \perp \mathsf{D} \mid \mathsf{B}, \mathsf{C}$, *then,*

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

Proof. Since $\mathsf{A} \perp \mathsf{D} \mid \mathsf{B}, \mathsf{C}$, by Fact A.1-(3), $\mathbb{H}(\mathsf{A} \mid \mathsf{B}, \mathsf{C}) = \mathbb{H}(\mathsf{A} \mid \mathsf{B}, \mathsf{C}, \mathsf{D})$. Moreover, since conditioning can only reduce the entropy (again by Fact A.1-(3)),

$$\mathbb{I}(\mathsf{A} \,;\mathsf{B} \mid \mathsf{C}) = \mathbb{H}(\mathsf{A} \mid \mathsf{C}) - \mathbb{H}(\mathsf{A} \mid \mathsf{B}, \mathsf{C}) \geq \mathbb{H}(\mathsf{A} \mid \mathsf{D}, \mathsf{C}) - \mathbb{H}(\mathsf{A} \mid \mathsf{B}, \mathsf{C})$$
$$= \mathbb{H}(\mathsf{A} \mid \mathsf{D}, \mathsf{C}) - \mathbb{H}(\mathsf{A} \mid \mathsf{B}, \mathsf{C}, \mathsf{D}) = \mathbb{I}(\mathsf{A} \,;\mathsf{B} \mid \mathsf{C}, \mathsf{D}). \quad \blacksquare$$

A.2   Measures of Distance Between Distributions

We will use the following two standard measures of distance (or divergence) between distributions.

KL-divergence.   For two distributions μ and ν, the *Kullback-Leibler divergence* between μ and ν is denoted by $\mathbb{D}(\mu \mid\mid \nu)$ and defined as:

$$\mathbb{D}(\mu \mid\mid \nu) := \underset{a \sim \mu}{\mathbb{E}} \left[\log \frac{\mathrm{Pr}_\mu(a)}{\mathrm{Pr}_\nu(a)} \right]. \tag{19}$$

The following states the relation between mutual information and KL-divergence.

Fact A.5. *For random variables* $\mathsf{A}, \mathsf{B}, \mathsf{C}$,

$$\mathbb{I}(\mathsf{A} \,;\mathsf{B} \mid \mathsf{C}) = \underset{(b,c) \sim (\mathsf{B},\mathsf{C})}{\mathbb{E}} \left[\mathbb{D}(\mathrm{dist}(\mathsf{A} \mid \mathsf{B} = b, \mathsf{C} = c) \mid\mid \mathrm{dist}(\mathsf{A} \mid \mathsf{C} = c)) \right].$$

Total variation distance. We denote the total variation distance between two distributions μ and ν on the same support Ω by $\|\mu - \nu\|_{\text{tvd}}$, defined as:

$$\|\mu - \nu\|_{\text{tvd}} := \max_{\Omega' \subseteq \Omega} \left(\mu(\Omega') - \nu(\Omega')\right) = \frac{1}{2} \cdot \sum_{x \in \Omega} |\mu(x) - \nu(x)| \,. \tag{20}$$

We use the following basic properties of total variation distance.

Fact A.6. *Suppose μ and ν are two distributions for $\mathcal{E}$, then,* $\Pr_\mu(\mathcal{E}) \leq \Pr_\nu(\mathcal{E}) + \|\mu - \nu\|_{\text{tvd}}$.

Fact A.7. *Suppose μ and ν are two distributions over the same support Ω; then, given one sample s from either μ or ν, the probability we can decide whether s came from μ or ν is $\frac{1}{2} + \frac{1}{2} \cdot \|\mu - \nu\|_{\text{tvd}}$.*

Finally, the following Pinsker's inequality bounds the total variation distance between two distributions based on their KL-divergence,

Fact A.8 (Pinsker's inequality). *For any distributions μ and ν,* $\|\mu - \nu\|_{\text{tvd}} \leq \sqrt{\frac{1}{2} \cdot \mathbb{D}(\mu \parallel \nu)}$.

A.3 XOR and Biases

For a $0/1$ random variable X, we define the *bias* of X as

$$\text{bias}(\mathsf{X}) := |\Pr(\mathsf{X} = 0) - \Pr(\mathsf{X} = 1)| = 2 \arg\max_{x \in \{0,1\}} \Pr(\mathsf{X} = x) - 1,$$

which measures how much X is biased a particular value. For $b := \arg\max_{x \in \{0,1\}} \Pr(\mathsf{X} = x)$:

$$\Pr(\mathsf{X} = x) = \begin{cases} \frac{1}{2} + \frac{\text{bias}(\mathsf{X})}{2} & \text{if } x = b \\ \frac{1}{2} - \frac{\text{bias}(\mathsf{X})}{2} & \text{if } x = 1 - b \end{cases} \,.$$

We use the following standard equality that shows taking XOR of independent random variables significantly dampen the resulting bias[8].

Proposition A.9. *For* independent *random variables* $\mathsf{X}_1, \ldots, \mathsf{X}_t$,

$$\text{bias}(\mathsf{X}_1 \oplus \cdots \oplus \mathsf{X}_t) = \prod_{i=1}^{t} \text{bias}(\mathsf{X}_i).$$

Proof. Let $\beta_i := \text{bias}(\mathsf{X}_i)$ and $b_i = \arg\max_{x \in \{0,1\}} \Pr[\mathsf{X}_i = x]$, and so $\Pr[\mathsf{X}_i = b_i] = \frac{1}{2}(1 + \beta_i)$. We prove this proposition by induction. Consider the base case of $t = 2$. We have,

$$\begin{aligned}
\Pr[\mathsf{X}_1 \oplus \mathsf{X}_2 = b_1 \oplus b_2] &= \Pr[\mathsf{X}_1 = b_1 \wedge \mathsf{X}_2 = b_2] + \Pr[\mathsf{X}_1 = 1 - b_1 \wedge \mathsf{X}_2 = 1 - b_2] \\
&= \frac{1}{2}(1 + \beta_1)\frac{1}{2}(1 + \beta_2) + \frac{1}{2}(1 - \beta_1)\frac{1}{2}(1 - \beta_2) \\
&= \frac{1}{4}\left((1 + \beta_1)(1 + \beta_2) + (1 - \beta_1)(1 - \beta_2)\right) \\
&= \frac{1}{4}\left(1 + \beta_1 + \beta_2 + \beta_1\beta_2 + 1 - \beta_1 - \beta_2 + \beta_1\beta_2\right) \\
&= \frac{1}{2}(1 + \beta_1\beta_2) = \frac{1}{2}(1 + \text{bias}(\mathsf{X}_1)\text{bias}(\mathsf{X}_2)).
\end{aligned}$$

Hence $\mathsf{X}_1 \oplus \mathsf{X}_2$ has bias $\text{bias}(\mathsf{X}_1)\text{bias}(\mathsf{X}_2)$. For the induction step k, we can set $\mathsf{Y} = \mathsf{X}_1 \oplus \ldots \oplus \mathsf{X}_{k-1}$ and by the induction hypothesis, we have $\text{bias}(\mathsf{Y}) = \Pi_{i=1}^{k-1}\text{bias}(\mathsf{X}_i)$. We can then apply the above argument again for random bits Y and X_k, and conclude the proof. ∎

---

[8]This is basically the intuition why one would expect any form of XOR Lemma to hold in the first place.

B Strong vs Weak XOR Lemmas and Optimality of Theorem 1

The literature on XOR Lemmas distinguishes between the notion of "weak" XOR Lemmas vs "Strong" XOR Lemmas. Roughly speaking, in a weak XOR Lemma, the goal is to show that the advantage over random guessing for $f^{\oplus \ell}$ is exponentially smaller than that of f, given *the same resources* (or sometimes even less resources). In a strong XOR Lemma however, the goal is to show that this exponential drop in the advantage happens even if we are given almost ℓ *times more resources* than what is needed for solving f itself. The reason to expect strong XOR Lemmas to hold, and they do hold in certain setting such as query complexity [BKLS20], is that we expect the ℓ-copy algorithm to have to "spread" its resources across each copies of f and thus get to spend ℓ times less resources per each individual copy. Which of these two categories our Theorem 1 belongs to?

Syntactically speaking, our Streaming XOR Lemma is clearly a weak XOR lemma as it uses the same exact resource of p-pass and s-space for both f and $f^{\oplus \ell}$. On the other hand, it is easy to see that one of the resources of interest here, *space*, is inherently different from the other resources considered in XOR lemmas such as circuit size, communication cost, or query complexity: unlike all these resources, *space is reusable*. As a result, it is pretty simple to show some examples where Theorem 1 is in fact optimal. For instance:

- *Example 1.* Let $f : \{0,1\}^{2m} \to \{0,1\}$ be the inner product of the first half and second half of $x \in \{0,1\}^{2m}$, i.e., $f(x) = \sum_{i=1}^{m} x_i \cdot x_{m+i} \mod 2$.

  It follows from standard communication complexity lower bounds of the inner product problem (see, e.g. [KN97]) that any single-pass $m/3$-space algorithm for f over the uniform distribution μ on $\{0,1\}^{2m}$ has probability of success $\leq 1/2 + 2^{-\Theta(m)}$.

  On the other hand, for the problem $f^{\oplus \ell}(x^1, \ldots, x^\ell)$, there is a trivial single-pass $(m + O(1))$-space algorithm that succeeds with probability one: Simply compute each $f(x^i)$ using m space and then maintain a running XOR of these values.

  This example shows that in any streaming XOR lemma, we cannot increase the space of the algorithm for $f^{\oplus \ell}$ by more than a factor of 3. As a result for any $\ell > 3$, there is no hope of getting any strong form of XOR lemma, at least for single pass algorithms.

- *Example 2.* Let f be a lopsided multi-party pointer chasing problem as follows:

$$f : [m^2]^m \times \underbrace{[m^2]^{m^2} \times \cdots [m^2]^{m^2}}_{r} \to \{0,1\} \,,$$

  and for a given $h : [m^2]^m \to [m^2]^{m^2}$ and $g^1, \ldots, g^r : [m^2]^{m^2} \to [m^2]^{m^2}$, compute the parity of

$$h(g^1(g^2(\cdots(g^r(1))))).$$

  It again follows from standard communication complexity lower bounds for pointer chasing (e.g. [Yeh16]; see also [GM08] for extension to multi-party version and streaming) that over the uniform distribution over $h, g^1, \ldots, g^r$ (with this order of arrival in the stream), any $(r-1)$-pass streaming algorithm requires $\Omega_r(m)$ space while any $(r-2)$-pass algorithm requires $\Omega_r(m^2)$ space and both bounds are tight (because size of h is small enough to be "shortcut" by an $\Omega(m)$ space algorithm).

  Again, let s be such that any $(r-1)$-pass algorithm for f with space s only succeeds with probability, say, $\leq 2/3$. It suffices to have $s = \alpha(r) \cdot m$ for a sufficiently small $\alpha(r)$ as a function

of r. Now, for the problem $f^{\oplus \ell}$, an algorithm that has $\ell \cdot m$ space for $\ell > \alpha(r)^{-1}$ can solve the problem even in $(r - 2)$ passes breaking a strong XOR Lemma.

This example shows that in any streaming XOR lemma, we cannot increase the space of the algorithm for $f^{\oplus \ell}$ by some large factor (which is at least proportional to the number of passes). Note that we chose f in this example so that the algorithm for $f^{\oplus \ell}$ has to still use $(r-2)$ passes; we can also play with number of h and g functions in definition of f to change this quantity to any other number of passes. This suggests that for *any* number of passes, one cannot hope for a strong streaming XOR Lemma.

The above examples point out the optimality of Theorem 1 in various cases. More conceptually, the intuition behind XOR Lemmas is that the naive algorithm that attempts to solve each subproblem individually (and thus independently), is more or less optimal. In most settings, this corresponds to a strong XOR Lemma. However, as above examples suggest, in the streaming, even the naive algorithm that solve each problem individually does not need to spend more resources per each subproblem as space is reusable. As such, it seems that semantically speaking, our Theorem 1 is the strongest type of XOR Lemma in our setting[9].

Remark B.1. *It is worth pointing out the work of Shaltiel [Sha03] who gave several nice examples in showing limitations of strong XOR Lemmas in general (e.g., circuit complexity, communication complexity, or query complexity). However, those examples and ours seem entirely different. In particular, Shaltiel's examples are based on working with distributions that are "often" easy and just become hard with small probability, which allows the algorithm for $f^{\oplus \ell}$ to spend more of its resources on a particular subproblem among the ℓ ones. Our examples do not have such a property and in fact they hold even for the notion of "fair" algorithms studied in [Sha03] that are algorithms that use the same amount of resources for each of the ℓ subproblems in $f^{\oplus \ell}$ (and break the examples of [Sha03]). We believe these differences are rooted in the different nature of space as a resource.*

Finally, let us conclude this section by a (rather unrelated) comment about Theorem 1.

Remark B.2. *Theorem 1 is stated as a distributional lower bound, which we found more natural for our purpose in this paper. However, it can also be stated as a worst-case lower bound by applying Yao's minimax principle twice: once to get a hard distribution for f, then applying Theorem 1, and another one to get a worst-case lower bound for $f^{\oplus \ell}$.*

C   An Optimal Algorithm for Noisy Gap Cycle Counting

We give a short and simple proof that demonstrate the asymptotic optimality of our lower bounds for noisy gap cycle counting in Theorem 2.

Proposition C.1. *For any $p, k \in \mathbb{N}^+$ such that k divides $2p$, there is a p-pass streaming algorithm for Noisy Gap Cycle Counting $\mathbf{NGC}_{n,k}$ that uses $O_{p,k}((n/k)^{1-2p/k} \cdot \log n)$ bits of space and outputs the correct answer with probability at least $2/3$.*

For simplicity of exposition, we presented Proposition C.1 for the case when $2p \mid k$. One can trivially extends the bounds to an $(n/k)^{1-O(p/k)}$ algorithm using this result (by picking the $\tilde{p} < p$ such that $2\tilde{p} \mid k$ and running the algorithm for $\tilde{p}$ passes). A bit more careful calculation can also give a sharper bound of $O_{p,k}((n/k)^{1-\frac{1}{\lceil k/2p \rceil}} \log n)$ for every $p \leq k$; we omit the details.

---

[9] Although we should note that one can consider more general type of Theorem 1 which, for instance, is defined by *interleaving* the streams of subproblems instead of *concatenating* them; such an approach is not particularly meaningful for our application in this paper but is an interesting question on its own.

Proof of Proposition C.1. Consider the following algorithm:

ALGORITHM 1: An optimal algorithm for the noisy gap cycle counting problem

---

1: Sample each vertex independently with probability $q := 36 \cdot (n/k)^{-2p/k}$ to get a set U – if $|U| > 100q \cdot n$ terminate and return FAIL.

2: For every vertex $u_i \in U$, find the depth-p BFS tree of u_i in G in p passes.

3: If there is a k-cycle among the visited BFS trees, output G is a "k-cycle case"; otherwise output it is a "$(2k)$-cycle case".

The pass complexity of Algorithm 1 is clearly p. For space complexity, considering the graph consists of only cycles and paths, every depth-p BFS tree can have at most $O(p)$ vertices and so the total space needed by the algorithm is $O(p|U|\log n) = O_{p,k}((n/k)^{1-2p/k}\log n)$ bits by the condition on terminating the algorithm.

We now prove the correctness. In the following, for simplicity, we remove the condition in the algorithm that terminates whenever U is large; as by Markov bound this event happens with probability $1/100$ anyway, this can only change the final probability of success calculated this way by a value of $-1/100$. Moreover, considering the algorithm has a one-sided error, i.e., never outputs "k-cycle" on a graph that does not have a k-cycle, we only need to prove that when the graph has a k-cycle, the algorithm finds it with probability at least $2/3$, which we do in the following.

Recall that $n = 6t \cdot k$ in $\mathbf{NGC}_{n,k}$ and we are interested in the case G has $2t$ vertex-disjoint k-cycles. Let $C_1, \ldots, C_{2t}$ denote these cycles. Fix an arbitrary cycle $C = (v_0, v_1, \ldots, v_{k-1})$ from this list. Pick the following $\beta := (k/2p)$ vertices from C: $v_0, v_{2\cdot p}, v_{4\cdot p}, \ldots, v_{2\beta \cdot p}(= v_k)$. The distance between any consecutive pairs of vertices in this list inside G is at most $2p$ and thus their depth-p BFS trees intersect with each other. As such, if we sample all these vertices in the algorithm, the set of BFS trees returned by the algorithm contains a k-cycle. As each vertex is sampled independently with probability q, the probability that this event happens is q^β.

On the other hand, since all cycles $C_1, \ldots, C_\beta$ are vertex-disjoint, the above event happens for each of them independently. As such, the probability that we do not see a k-cycle is at most

$$(1 - q^\beta)^{2t} \leq \exp\left(-q^\beta \cdot 2t\right) = \exp\left(-(36 \cdot (k/n)^{2p/k})^{k/2p} \cdot (n/3k)\right) \leq \exp\left(-12\right) < 1/100.$$

As such Algorithm 1 outputs the correct answer with probability at least $1 - 2/100 > 2/3$. ∎

D A Technical Comparison with the Lower Bound of [AKSY20]

[AKSY20] previously proved a multi-pass lower for the (no-noise) gap cycle counting problem. Their work starts by proving a lower bound for a generalization of pointer chasing wherein the goal is to chase multiple pointers simultaneously; this lower bound is also in the low-probability regime, i.e., $1/\text{poly}(n)$ advantage over random guessing, but *only works for single-pass algorithms*. This is then plugged into an intricate *round/pass-elimination* argument that, informally speaking, shows that solving the problem in p passes for depth-k pointers is as hard as solving the problem in $p-1$ passes but for depth $k/2$ instead. Applying this step inductively gives an $\Omega(\log k)$ pass lower bound for $n^{o(1)}$-space algorithms. A quick technical summary of [AKSY20] is then the following: prove a "strong" single-pass lower bound first and then increase the number of passes without "weakening" the lower bound too much. This somewhat the exact opposite of what we do in this paper: we prove a "weak" multi-pass lower bound first and then plug it into our streaming XOR Lemma to

amplify its hardness and obtain a "strong" lower bound via a reduction that slightly reduces the number of passes. This natural hardness amplification approach leads to exponentially stronger lower bound of $\Omega(k)$ passes for $n^{o(1)}$-space algorithms with a considerably simpler proof.

It is worth pointing out two other differences (both technical and conceptual) as well. Firstly, owing to the introduction of noise, we obtain a clear reduction from pointer chasing instead of the implicit connection in [AKSY20] that is scattered throughout the proof. Secondly, we directly consider streaming algorithms as opposed to working with the two-party communication model in [AKSY20] and reducing it to streaming only at the end; this is critical in our work as our XOR Lemma is for streaming algorithms and not (two-party) communication (obtaining XOR Lemmas in communication complexity is an interesting open question; see, e.g. [Wei15, Open Problem 6.3]).