

# Factorial Lower Bounds for (Almost) Random Order Streams

Ashish Chiplunkar  
IIT Delhi  
Delhi, India  
ashish.chiplunkar@cse.iitd.ac.in

John Kallaugher  
Sandia National Labs  
Albuquerque, USA  
jmkall@sandia.gov

Michael Kapralov  
EPFL  
Lausanne, Switzerland  
michael.kapralov@epfl.ch

Eric Price  
UT Austin  
Austin, USA  
ecprice@cs.utexas.edu

**Abstract**—In this paper we introduce and study the STREAMINGCYCLES problem, a random order streaming version of the Boolean Hidden Hypermatching problem that has been instrumental in streaming lower bounds over the past decade. In this problem the edges of a graph  $G$ , comprising  $n/\ell$  disjoint length- $\ell$  cycles on  $n$  vertices, are partitioned randomly among  $n$  players. Every edge is annotated with an independent uniformly random bit, and the players’ task is to output, for some cycle in  $G$ , the sum (modulo 2) of the bits on its edges, after one round of sequential communication.

Our main result is an  $\ell^{\Omega(\ell)}$  lower bound on the communication complexity of STREAMINGCYCLES, which is tight up to constant factors in the exponent. Applications of our lower bound for STREAMINGCYCLES include an essentially tight lower bound for component collection in (almost) random order graph streams, making progress towards a conjecture of Peng and Sohler [SODA’18] and the first exponential space lower bounds for random walk generation.

**Index Terms**—streaming algorithms, graph algorithms, fourier analysis, communication complexity

## I. INTRODUCTION

The streaming model of computation has been a core model for small space algorithms that process large datasets since the foundational paper of [1] showed how to (approximately) compute basic statistics of large datasets using space only polylogarithmic in the size of the input. While many such stream statistics are tractable to calculate in this model—including frequency moments, number of distinct elements, and heavy hitters—for graphs of  $n$  edges even some basic problems are known to require space  $\Omega(n)$  in streams in which the edge arrival order is chosen adversarially. At the same time, a recent line of work on testing graph properties [2], [3], [4], [5], [6], [7] shows that when the edges of the input graph are presented in a uniformly random order, one can approximate many fundamental graph properties (e.g., matching size, number of connected components, constant query testable properties in bounded degree graphs) in polylogarithmic or even constant space.

These algorithms make use of small space graph exploration primitives, including:

a) *Component collection.*: Given a graph  $G = (V, E)$  presented as a random order stream and a budget  $k$ , collect the connected components of a representative sample of vertices of  $G$  assuming that many vertices in  $G$  belong to components of size bounded by  $k^1$ . The work of [5] designs a component collection primitive that uses  $k^{O(k^3)}$  space, and then uses it to obtain an additive  $\varepsilon n$ -approximation to the number of connected components in  $G$ .

b) *Random walk generation.*: Given a graph  $G = (V, E)$  presented as a random order stream, a target walk length  $k$  and a budget  $s$ , generate a sample of (close to) independent random walks of length  $k$  from  $s$  vertices in  $G$  selected uniformly at random. The work of [8] designs a primitive that outputs such walks (with constant TVD distance to the desired distribution, say), using space  $2^{O(k^2)} s$ .

c) *k-disc estimation.*: Given a bounded degree graph  $G = (V, E)$  presented as a random order stream, an integer  $k$ , estimate  $k$ -disc frequencies<sup>2</sup>. The work of [4] designs a primitive for estimating  $k$ -disc frequencies and then uses it to show that any constant query testable property of bounded degree graphs is random order streamable.

<sup>1</sup>We note that our definition of component collection here is not formal by design. This is because our lower bound applies to a very weak formal version of component collection, in which one is promised that a constant fraction of vertices of the input graph belong to components of size at most  $k$ , and the task is to output one such vertex (see Definition IV.2 in Section IV). Our algorithmic results, on the other hand, solve the stronger version of the problem, namely output a ‘representative sample’ of vertices of the graph together with components that they belong to (as long as those are of size bounded by  $k$ ), in any graph. In particular, our algorithmic primitive naturally leads to an algorithm that additively approximates the number of connected components in the input graph, as in [5].

<sup>2</sup>A  $k$ -disc is the subgraph induced by vertices at shortest path distance at most  $k$  from a given vertex, and the set of  $k$ -disc frequencies corresponds to the numbers of occurrences of all such graphs, up to isomorphism.

The above primitives perform depth- $k$  exploration in random order graph streams using space exponential in  $k$ . Our work is motivated by the natural question:

Does depth- $k$  exploration in random order streams require space exponential in  $k$ ?

This question was originally raised by [5], who wrote

*...it will also be interesting to obtain lower bounds for random order streams. It seems to be plausible to conjecture that approximating the number of connected components requires space exponential in  $1/\varepsilon$ . It would be nice to have lower bounds that confirm this conjecture.*

In this paper we make progress towards this conjecture, showing that this dependence is indeed necessary, at least in graph streams that admit some amount of correlation. Our lower bound is based on a new communication problem that we refer to as the STREAMINGCYCLES problem, a relative of the well-studied Boolean Hidden Hypermatching problem. We introduce the STREAMINGCYCLES problem next, then give reductions from component collection and random walk generation. Our reduction generates instances of the component collection and random walk generation problem that are not quite random order streams, but rather allow for small *batches* of edges as opposed to edges themselves to arrive in a random order. We argue that this is in fact a very natural *robust* analog of the idealized random order streaming model, and show that corresponding random order streaming algorithms extend to the batch random order setting. Finally, we give an overview of our lower bound, which is the main technical contribution of the paper.

*d) The STREAMINGCYCLES problem.*: Our main technical contribution is a tight lower bound for the STREAMINGCYCLES problem, which we now define. In an instance of STREAMINGCYCLES( $n, \ell$ ) a graph  $G = (V, E)$  made up of  $n/\ell$  length- $\ell$  cycles is received as a stream of edges  $e$  with bit labels  $x_e$ . There are  $n$  players, indexed by the edges of  $G$ . Upon arrival of an edge  $e$  the corresponding bit label  $x_e$  is given to the corresponding player as private input, together with a message from the previous player. The edges  $e$  are posted on a common board as the edges arrive. The last player must return a vertex  $v \in V$  and the *parity* of the cycle  $C \subseteq E$  containing  $v$ , i.e.  $\sum_{e \in C} x_e$ . We consider a distributional version of the problem, in which the bits  $x_e$  are chosen independently and uniformly at random, and the ordering of the edges in the stream is uniformly random.

A naïve protocol for the STREAMINGCYCLES problem

is for the players to track the connected component of a vertex  $u \in G$ . This strategy succeeds if and only if the edges of the component arrive “in order”, which happens with probability  $\ell^{-\Theta(\ell)}$ . Thus, it suffices to track  $\ell^{O(\ell)}$  vertices, which results in an  $\ell^{O(\ell)}$  communication per player protocol. Our main result shows that this is essentially best possible:

**Theorem 1.1** (Main result; informal version of Theorem IV.1). *Any protocol for the STREAMINGCYCLES( $n, \ell$ ) problem that succeeds with  $2/3$  probability requires  $\min(\ell^{\Omega(\ell)}, n^{0.99})$  bits of communication from some player.*

*e) Relation to the Boolean Hidden Hypermatching problem.*: We note that this is related to the search version of the Boolean Hidden Hypermatching problem, in which a vector  $x \in \{0, 1\}^n$  is given to Alice, who sends a single message to Bob. Bob, in addition to the message from Alice, is given a perfect hypermatching with hyperedges of size  $\ell$  and must output the parity of  $x$  on one of the hyperedges. The Boolean Hidden Hypermatching problem admits a protocol with  $O(n^{1-1/\ell})$  communication (Alice simply sends Bob the values of  $x$  on a uniformly random subset of coordinates of size  $n^{1-1/\ell}$ ), and this bound is tight. Note that in the STREAMINGCYCLES problem the bits of  $x$  are associated with edges in the cycles, and therefore every cycle can naturally be associated with a hyperedge in Boolean Hidden Hypermatching problem. In contrast to the Boolean Hidden Hypermatching problem, in which the bits are presented first and then the hyperedges are revealed, in the STREAMINGCYCLES problem the bits and the identities of the hyperedges are gradually revealed to the algorithm. Similarly to the Boolean Hidden Hypermatching problem, a ‘sampling’ protocol turns out to be nearly optimal. The naïve protocol mentioned above, and considered in more detail in Section III, solves STREAMINGCYCLES( $n, \ell$ ) using  $O(\ell!)$  samples, and we prove a nearly matching lower bound of  $\min\{\ell^{\Omega(\ell)}, n^{0.99}\}$  bits.

*f) Applications to component collection and random walk generation.*: We now give a natural way for the players to produce a graph stream based on their inputs to the communication game:

- Define the vertex set  $V' = V \times \{0, 1\}$ .
- On receiving the  $t^{\text{th}}$  edge  $(uv, b_{uv})$ , insert edges  $(u, 0)(v, b_{uv})$  and  $(u, 1)(v, \bar{b}_{uv})$  into the stream.

In other words, every edge of the graph  $G$  in the STREAMINGCYCLES problem becomes a pair of edges in  $G'$ . The order in which edges of  $G'$  are presented is not quite random as *pairs of edges* as opposed to individual edges arrive in a uniformly random order.

We refer to such streams, in which *batches* of edges arrive uniformly at random in the stream as opposed to edges themselves, as *(hidden-)batch* random order streams. Note that the reduction above generates a stream with batches of size two, corresponding to the pairs of edges arriving at the same time. We argue in Section III-A below that the batched model, in which arrival times of edges could be correlated, but the correlations are restricted by bounded size batches, is a very natural *robust* analog of the idealized uniformly random streaming model. In particular, we show that, surprisingly, some existing random order streaming algorithms for estimating graph properties are quite robust, and can be made to work even when the structure of the batches is not known to the algorithm, i.e. in the *hidden-batch* random order model.

Using the reduction above together with Theorem I.1, we get

**Theorem I.2** (Component collection lower bound; informal version of Theorem IV.3). *Component collection requires  $k^{\Omega(k)}$  bits of space in hidden-batch random order streams.*

*Proof.* If the component collection algorithm returns a vertex  $(v, b)$  together with a component of size  $\ell$  that contains  $(v, b)$ , return  $v$  and parity = 0. If it returns  $(v, b)$  and a component of size  $2\ell$  containing  $(v, b)$ , return  $v$  and parity = 1. Otherwise fail.  $\square$

The bound provided by Theorem I.2 is **tight up to constant factors in the exponent**. We give an algorithm with  $k^{O(k)}$  space complexity in Section 5 of the full version:

**Theorem I.3** (Component collection upper bound; informal version of Theorem 5.2 of the full version). *There exists a component collection algorithm in (hidden-batch) random order streams with space complexity  $k^{O(k)}$  (words).*

Similarly to the work of [5], our component collection algorithm can be used to estimate the number of connected components to additive precision  $\varepsilon n$ . The space complexity of our estimation algorithm is  $(1/\varepsilon)^{O(1/\varepsilon)}$ , similarly improving upon on [5]. The details are provided in Section 5 of the full version.

Similarly, we obtain exponential lower bounds for the random walk generation problem:

**Theorem I.4** (Random walk generation lower bound; informal version of Theorem IV.7). *Generation of a random walk of length  $k$  started at any vertex in a graph given as a hidden-batch random order stream requires*

*$k^{\Omega(\sqrt{k})}$  space. Generation of  $C \cdot 4^k$  random walks for a sufficiently large constant  $C > 0$  requires  $k^{\Omega(k)}$  space.*

*Proof.* For the first lower bound, let  $\ell = \sqrt{k/C}$  for a sufficiently large absolute constant  $C$ , so that  $k = C\ell^2$ . Generate a walk of length  $k$  with precision  $\varepsilon = 1/10$  in total variation distance. The walk loops around the cycle that it starts in with probability at least  $2/3$ . Let  $(v, b)$  denote the starting vertex. If the cycle is of length  $\ell$ , output  $v$  and parity = 0. If the cycle is of length  $2\ell$ , output  $v$  and parity = 1. Thus, random walk generation requires at least  $\ell^{\Omega(\ell)} = k^{\Omega(\sqrt{k})}$  space.

For the second bound, let  $\ell = k/2$  and run  $C4^k = C2^{2\ell}$  random walks of length  $k$  started at uniformly random vertices, with precision  $1/10$  in total variation distance (for the joint distribution), for a sufficiently large constant  $C > 0$ . With probability at least  $2/3$  at least one of the walks will loop around the cycle that it started in. Let  $(v, b)$  denote the starting vertex. If the cycle is of length  $\ell$ , output  $v$  and parity = 0. If the cycle is of length  $2\ell$ , output  $v$  and parity = 1.  $\square$

g) *Boolean Fourier Analysis for Many-Player Games:* Our lower bound for STREAMINGCYCLES is based on the techniques of *Boolean Fourier analysis*. The application of these techniques to one-way communication complexity goes back to [9], but previous applications have either involved two players or at most a small number relative to the size of the input, meaning that they can afford to lose factors polynomial or even exponential in the player count. Our application involves  $n$  players for an  $O(n)$ -sized input, requiring a careful consideration of how the Fourier coefficients associated with the players' messages evolve as each player passes to the next. We give an overview of these techniques in Sections II and III.

#### A. Related work

The random order streaming model has seen a lot of attention recently. Besides the aforementioned work of [5] that gives small space algorithms for component counting, small space approximations to matching size have been given in [2], [3], [4], [6] (naturally, the problem has also attracted significant attention in adversarial streams, but the space complexity of known algorithms in this model is significantly higher than in random order streams [10], [11], [12], [13], [14], [15], [16], [17]). The work of [4] shows that constant query testable graph properties can be tested in constant space in random order streams in bounded degree graphs.

The Fourier-analytic approach to proving communication complexity lower bounds pioneered by [9] has

been instrumental in lower bounds for many graph problems, including cycle counting [18], estimating MAX-CUT value [19], [20], [21] more general CSPs [22], [23], [24], and subgraph counting [25]. Communication problems inspired by the Boolean hidden matching problem (and therefore related to the STREAMINGCYCLES problem that forms the basis of our lower bound) have also been recently used to obtain lower bounds for multipass algorithms for several fundamental graph streaming problems [12], [26], [27]. Lower bounds for statistical estimation problems (e.g., distinct elements, frequency moments and quantile estimation) in random order streams were given in [28], [29].

## II. WARM-UP: BOOLEAN HIDDEN HYPERMATCHING WITH INTERLEAVING

Let  $\mathbf{X} \in \{0, 1\}^n$  be uniformly random, and revealed one bit at a time to our algorithm. Let our algorithm's state at time  $t$  be  $\mathbf{M}_t$ , which is at most  $c$  bits long. The Fourier-analytic lower bound approach studies quantities corresponding to the following question: what is known about the *parity* of each set of bits at time  $t$ ? For the indicator  $z \in \{0, 1\}^t$  of any subset of bits that arrive before time  $t$ , define

$$\tilde{\mathbf{F}}_t(z) := \mathbb{E}_{x|\mathbf{M}_t} [(-1)^{z \cdot x}] \quad (1)$$

which is  $\pm 1$  if the corresponding parity is specified by the message and 0 if it is completely unknown. One can think of  $\tilde{\mathbf{F}}_t^2(z) \in [0, 1]$  as an estimate of how well the parity  $z$  is remembered at time  $t$ . A method that stores  $c$  individual bits would have

$$\forall k \leq c, \sum_{|z|=k} \tilde{\mathbf{F}}_t^2(z) = \binom{c}{k},$$

where  $|z|$  denotes the Hamming weight of  $z$ , because it remembers exactly each subset of those bits. The foundation of the Fourier-analytic approach is that a similar bound typically holds for *any* protocol that generates  $c$ -bit messages  $\mathbf{M}_t$ :

$$\forall k \leq c, \sum_{|z|=k} \tilde{\mathbf{F}}_t^2(z) \leq \binom{O(c)}{k} \quad (2)$$

with very good probability over  $\mathbf{M}_t$ . This inequality (Lemma 3 in [9]) is a consequence of the hypercontractive inequality (see Lemma 3.4 in [30]).

In Boolean Hidden Hypermatching, one first receives the bits  $x$  and then receives  $\Theta(n/\ell)$  “important” sets  $z^{(i)}$ , each of size  $\ell$ . Since the sets are uniform and independent of the message  $\mathbf{M}_t$  (and so, of  $\tilde{\mathbf{F}}_t$ ), the expected amount known about them is

$$\mathbb{E}_{z^{(i)}} \left[ \sum_{i \in [n/\ell]} \tilde{\mathbf{F}}_t^2(z^{(i)}) \right]$$

which in turn is

$$(n/\ell) \frac{1}{\binom{n}{\ell}} \sum_{|z|=\ell} \tilde{\mathbf{F}}_t^2(z)$$

which is bounded by

$$(n/\ell) \frac{\binom{O(c)}{\ell}}{\binom{n}{\ell}}$$

finally giving us a bound of

$$(n/\ell) \left( \frac{O(c)}{n} \right)^\ell. \quad (3)$$

If  $c \ll n^{1-1/\ell}$ , this is  $o(1)$  so the algorithm probably does not remember any of the important parities.

The challenge we face in adapting this approach is that our important sets (the components of the graph) are revealed over time, interleaved with the bits of  $x$  rather than at the end.

To see how this can be an issue, consider a two-stage version of Boolean Hidden Hypermatching: the first  $n/2$  bits of  $x$  are given, then at time  $s = n/2$  we receive  $z_{\leq s}^{(i)}$  (elements  $z^{(i)}$  with indices at most  $s$ ) for each  $i$ , then the rest of  $x$ , and finally at time  $t = n$  we receive the rest of the important indices  $z_{[s+1:t]}^{(i)}$  (elements  $z^{(i)}$  with indices between  $s+1$  and  $t$ ). For simplicity, suppose  $|z_{\leq s}^{(i)}| = |z_{[s+1:t]}^{(i)}| = \ell/2$  always. The algorithm that stores a random subset of bits still needs  $c \gtrsim n^{1-1/\ell}$ . Solving either half of the stream (determining the parity of one of the half-sets  $z_{\leq s}^{(i)}, z_{[s+1:t]}^{(i)}$ ) requires only  $n^{1-2/\ell}$  space in general, but how can we get a tight  $n^{1-1/\ell}$  bound?

The problem is that (2) does not give strong enough control over the higher-order moments to show (3). The sets  $z^{(i)}$  at the end are no longer independent of  $\tilde{\mathbf{F}}_t$ , because the algorithm's behavior in the second half can depend on the  $z_{\leq s}$ . One could instead apply (2) to each half of the stream and take the product, getting

$$\sum_z \tilde{\mathbf{F}}_s^2(z_{\leq s}) \tilde{\mathbf{F}}_t^2(0^s z_{[s+1:t]})$$

to be equal to

$$\left( \sum_{|z_{\leq s}|=\ell/2} \tilde{\mathbf{F}}_s^2(z_{\leq s}) \right) \left( \sum_{|z_{[s+1:t]}|=\ell/2} \tilde{\mathbf{F}}_t^2(0^s z_{[s+1:t]}) \right)$$

which is in turn bounded by

$$\left( \frac{O(c)}{\ell/2} \right)^2$$

and so, on average over  $z$ ,

$$\sum_{i \in [n/\ell]} \tilde{\mathbf{F}}_s^2(z_{\leq s}^{(i)}) \tilde{\mathbf{F}}_t^2(0^s z_{[s+1:t]}^{(i)}) \leq (n/\ell) \left( \frac{O(c)}{n} \right)^\ell.$$

For algorithms that store individual bits this implies (3), since in that case

$$\tilde{\mathbf{F}}_t(z) = \tilde{\mathbf{F}}_t(z_{\leq s} 0^{t-s}) \tilde{\mathbf{F}}_t(0^s z_{[s+1:t]})$$

and

$$\tilde{\mathbf{F}}_t^2(z_{\leq s} 0^{t-s}) \leq \tilde{\mathbf{F}}_s^2(z_{\leq s}).$$

However, for general algorithms,

$$\tilde{\mathbf{F}}_t(z) \neq \tilde{\mathbf{F}}_t(z_{\leq s} 0^{t-s}) \tilde{\mathbf{F}}_t(0^s z_{[s+1:t]}).$$

To solve this, we need to relate  $\tilde{\mathbf{F}}_t(z)$  to bounds involving  $z_{\leq s}$  and  $z_{[s+1:t]}$  individually. We define

$$\tilde{\mathbf{r}}_{s,t}(z_{[s+1:t]}) := \mathbb{E}_{x|\mathbf{M}_s, \mathbf{M}_t, \mathbf{B}_t} \left[ (-1)^{z_{[s+1:t]} \cdot x_{[s+1:t]}} \right]$$

as a “double-ended” version of (1): it asks about the knowledge of  $H$  given the states before and after  $H$  arrives, as well as the “board”  $\mathbf{B}_t$  at time  $t$  (which is the information about important sets revealed by time  $t$ , namely the  $z_{\leq s}^{(i)}$ ). Since  $x$  is independent of  $\mathbf{B}_t$ , this is specified by  $2c$  bits ( $\mathbf{M}_s$  and  $\mathbf{M}_t$ ), so it also satisfies (2). Our key observation, Lemma III.5, is that

$$\tilde{\mathbf{F}}_t(z) = \mathbb{E}_{\mathbf{M}_s|\mathbf{M}_t, \mathbf{B}_t} \left[ \tilde{\mathbf{F}}_s(z_{\leq s}) \tilde{\mathbf{r}}_{s,t}(z_{[s+1:t]}) \right]. \quad (4)$$

This lets us relate  $\tilde{\mathbf{F}}_t(z)$  to  $\tilde{\mathbf{F}}_s(z_{\leq s})$  and  $\tilde{\mathbf{r}}_{s,t}(z_{[s+1:t]})$ , each of which are bounded by (2).

Specifically, for any fixed index  $i$ , the average amount that is remembered about  $z^{(i)}$  is:

$$\begin{aligned} & \mathbb{E}_{z^{(i)}, \mathbf{M}_s, \mathbf{M}_t} \left[ \tilde{\mathbf{F}}_t(z^{(i)})^2 \right] \\ & \leq \mathbb{E}_{\mathbf{M}_s, \mathbf{M}_t} \left[ \mathbb{E}_{z^{(i)}} \left[ \tilde{\mathbf{F}}_s^2(z_{\leq s}^{(i)}) \tilde{\mathbf{r}}_{s,t}^2(z_{[s+1:t]}^{(i)}) \right] \right] \\ & = \mathbb{E}_{\mathbf{M}_s, \mathbf{M}_t} \left[ \mathbb{E}_{z_{\leq s}^{(i)}} \left[ \tilde{\mathbf{F}}_s^2(z_{\leq s}^{(i)}) \right] \mathbb{E}_{z_{[s+1:t]}^{(i)}} \left[ \tilde{\mathbf{r}}_{s,t}^2(z_{[s+1:t]}^{(i)}) \right] \right] \\ & \leq \frac{\binom{O(c)}{\ell/2}}{\binom{n/2}{\ell/2}} \cdot \frac{\binom{O(2c)}{\ell/2}}{\binom{n/2}{\ell/2}} = \left( \frac{O(c)}{n} \right)^\ell. \end{aligned}$$

Thus, for the two-stage version of Boolean Hidden Hypermatching, we need  $c \gtrsim n^{1-1/\ell}$  to remember one of the  $n/\ell$  important sets on average.

These equations, (4) and (2), form the Fourier-analytic basis of our lower bound. The rest of the challenge for our setting comes from random order streams having much more complicated combinatorics than two-stage hypermatching, spread across  $n$  stages. As edges arrive, components appear, extend, and merge to eventually form the final cycles.

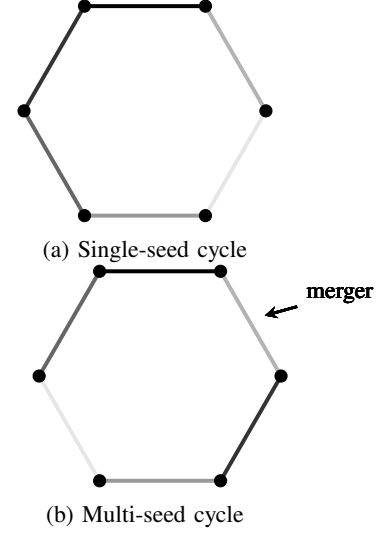


Fig. 1: Illustration of two possible edge arrival orders. Lighter edges arrive later.

### III. TECHNICAL OVERVIEW

*a) A basic “sampling” protocol.* Suppose we only permit ourselves to remember one parity (so using one bit of space, in addition to whatever space we need to know which parity this is). We want to eventually learn the parity of one cycle in the stream, and so the natural strategy is as follows:

- 1) Arbitrarily choose some edge to start with, and record its parity.
- 2) Whenever we see a new edge that is incident to the parity we are storing, add that edge to the parity, and hope our parity eventually grows to encompass an entire cycle.

This strategy will succeed with probability  $\ell^{-\Theta(\ell)}$ , as it only works if no edge of the cycle arrives before a path to it from the first edge has already arrived (we refer to such a cycle as a “single-seed” cycle—see Fig. 1a for an illustration). So we would have to repeat this process  $\ell^{\Theta(\ell)}$  times in order to achieve a constant probability of success.

Can we hope to do better by not considering the parities independently? Suppose we maintain the parities of  $c$  paths at a time, which may merge with each other as we process the stream. We now have some chance of finding “multi-seed” cycles, i.e. cycles in which several disjoint paths arrive before eventually being merged by later edge arrivals—see Fig. 1b for an illustration. If we happen to have remembered the parity of each of the components that eventually merged into a given multi-seed cycle, we will find the parity of the cycle. The

chance that any  $k$  of them are from the same cycle is  $\sim \binom{c}{k} (\ell/n)^{k-1}$ , and the chance of any given cycle having only  $k$  seeds is  $(\ell/k)^{-\Theta(\ell)}$ , as it requires  $k$  paths of average length  $\ell/k$  to arrive in order. Until  $c$  is  $n^{1-\Theta(1/\ell)}$ , the probability of finding the parity of a cycle will therefore be dominated by the single-seed case.

However, so far we have assumed we can only store individual parities. To extend these arguments to algorithms that maintain arbitrary state, we make use of the tools of *Boolean Fourier analysis*.

b) *Fourier-analytic lower bound for STREAMING-CYCLES.*: We construct a hard instance for the problem in which the bit labels  $\mathbf{X}$  for the edges are chosen uniformly at random, and in order to simplify the analysis we allow the algorithm to remember which edges it has seen for free (although not the bit labels). We say that these edges are posted on the “board”  $\mathbf{B}$ . The state of the board at time  $t$ , i.e. after receiving  $t$  edges, is denoted by  $\mathbf{B}_t$ .

For any subset  $z \in \{0,1\}^t$  of the edges that have arrived so far (given by the appropriate bit mask), we can associate the expectation of the parity of  $z$  given  $\mathbf{M}_t$  with the *normalized Fourier coefficient*

$$\tilde{\mathbf{F}}_t(z) := \mathbb{E}_{x|\mathbf{M}_t} [(-1)^{z \cdot x}].$$

Note that if the algorithm returns  $v$ , and  $C$  is the cycle containing  $v$  (written as an element of  $\{0,1\}^n$ ), the algorithm’s best guess for the parity of  $C$  will be 1 if  $\tilde{\mathbf{F}}_n(C) > 0$  and  $-1$  otherwise. Moreover, the probability that this guess will be correct is  $\frac{1+|\tilde{\mathbf{F}}_n(C)|}{2}$ . Therefore, for a lower bound, it will suffice to prove that with good probability

$$|\tilde{\mathbf{F}}_n(C)| = o(1).$$

c) *Fourier mass on collections of component types.*: Writing  $\mathbb{Z}_+^{\{\}}$  for the set of multisets of integers, for  $\beta \in \mathbb{Z}_+^{\{\}}$  and a  $z \in \{0,1\}^t$  we define

$$z \sim_t \beta$$

to be true iff  $z$  corresponds to a set of edges which contains  $\beta[a]$  components (i.e., paths or cycles) of length  $a$  for each  $a$  (here for  $\beta \in \mathbb{Z}_+^{\{\}}$  and an integer  $a$  we write  $\beta[a]$  to denote the number of occurrences of  $a$  in  $\beta$ ). Our main object of study in the lower bound proof is

$$\mathbf{H}_\beta^t := \sum_{z \in \{0,1\}^t, z \sim_t \beta} \tilde{\mathbf{F}}_t(z)^2,$$

which can be viewed as the amount of certainty that the algorithm has about parities of unions of components whose sizes are prescribed by  $\beta$ .

In order to build some intuition, we consider our prototypical sampling protocol from the start of Section III. Let  $c$  be the number of edges sampled at the beginning and for every  $j \geq 1$  let  $Y_j^t$  denote the number of components of size  $j$  (i.e., with  $j$  edges) that the sampling algorithm was able to construct at time  $t$ . Then for every multiset  $\beta$  and every  $t$  one simply has

$$\mathbf{H}_\beta^t = \prod_a \binom{Y_a^t}{\beta[a]}.$$

and

$$\mathbf{H}_{\{1\}}^c = c,$$

where the last equality holds because the sampling algorithm grows components out of the first  $c$  arriving edges, and for every  $j$  we have that  $\mathbf{H}_{\{j\}}^t$  is the number of components of size  $j$  that the algorithm knows the parity of at time  $t$ . In order to prove that the sampling protocol does not succeed, we need to prove that  $Y_\ell^n = 0$ , and for general protocols we need to prove

**Lemma III.1.** *For all  $\varepsilon > 0$ , there is a  $D > 0$  depending only on  $\varepsilon$  such that, if  $c < \min(\ell^{D/D}, n^{1-\varepsilon})$  and  $D < \ell < D^{-1} \log n$ ,*

$$\mathbb{E}_{\mathbf{X}, \mathbf{B}_n} [\mathbf{H}_{\{\ell\}}^n] \leq \varepsilon.$$

In order to establish Lemma III.1, we bound the expected evolution of  $\mathbb{E}[\mathbf{H}_\beta^t]$  as a function of  $t$ . In what follows we first analyze the evolution of  $\mathbf{H}_\beta^t$  for the simple “sampling” protocol, and then present the main ideas of our analysis.

d) *Evolution of Fourier coefficients of the “sampling” protocol.*: Fix a component of size  $j$ . At time  $t$  the probability that it gets extended is about  $\frac{2}{n-t}$ , and therefore

$$\mathbb{E}[Y_j^t | \mathbf{B}_{t-1}, \mathbf{F}_{t-1}] \approx Y_j^{t-1} + \frac{2}{n-t} \cdot Y_{j-1}^{t-1}$$

+ (contribution from merges of smaller components).

In order to derive the asymptotics of  $Y_j^t$ , we first ignore the contribution of merges, and later verify that they do not affect the result significantly. In particular, ignoring the contribution of merges, we get

$$\mathbb{E}[Y_j^t | \mathbf{B}_{t-1}, \mathbf{F}_{t-1}] \approx Y_j^{t-1} + \frac{2}{n-t} \cdot Y_{j-1}^{t-1}.$$

We assume for intuition that  $t \leq n/2$ , i.e. we are only looking at the first half of the stream. Since the initial conditions are (essentially)  $Y_1^1 = c$  and  $Y_j^1 = 0$  for  $j > 1$ , because the algorithm can remember  $c$  single edges at the beginning of the stream, and no larger components (since they typically do not form at the very beginning of the stream). This now yields that

$$Y_j^t \leq c \cdot 4^{j-1} (t/n)^{j-1} / (j-1)!$$

for  $t \leq n/2$  and all  $j \geq 1$ . This is because  $Y_j^1 = c$  as required, and for  $j \geq 2$

$$\begin{aligned} \mathbb{E}[Y_j^t] &\leq \sum_{s=1}^{t-1} \frac{2}{n-s} Y_{j-1}^s \\ &\leq (c \cdot 4^{j-1}/(j-2)!) \frac{1}{n} \sum_{s=1}^{t-1} (t/n)^{j-2} \\ &\approx c \cdot 4^{j-1}/(j-2)! \cdot \int_0^{t/n} x^{j-2} dx \\ &= c \cdot 4^{j-1} (t/n)^{j-1}/(j-1)! \end{aligned} \quad (5)$$

This in particular implies that  $Y_\ell^{t/2} \ll 1$  if  $c = \ell^{o(\ell)}$ , and in general that for the sampling protocol we have, at least for  $t \leq n/2$ ,

$$\begin{aligned} \mathbb{E}[\mathbf{H}_\beta^t] &= \mathbb{E} \left[ \prod_{j \in \beta} Y_j^t \right] \\ &\approx \prod_{j \in \beta} c \cdot 4^{j-1} (t/n)^{j-1}/(j-1)! \\ &\lesssim \left( \prod_{j \in \beta} \frac{1}{j!} \right) \cdot Q^{|\beta|_*} \cdot \left( \frac{t}{n} \right)^{|\beta|_* - |\beta|} \cdot c^{|\beta|}, \end{aligned}$$

where  $Q$  is an absolute constant,  $|\beta|_* = \sum_{i \in \beta} i$  and  $|\beta|$  is the number of elements in the multiset  $\beta$  (counting multiplicities).

*e) Evolution of Fourier coefficients of a general protocol.* The outline of the simple “sampling” protocol above provides a good model for our general proof. Specifically, in Lemma 4.37 of the full version we show that there exists a constant  $Q > 0$  such that for (almost) all  $\beta$  and  $t$  (the near-end of the stream and going from  $\beta = \{\ell-1\}$  to  $\beta = \{\ell\}$  require some special treatment) one has

$$\mathbb{E}[\mathbf{H}_\beta^t] \lesssim \left( \prod_{j \in \beta} \frac{1}{j!} \right) \cdot Q^{|\beta|_*} \cdot \left( \frac{t}{n} \right)^{|\beta|_* - \nu(\beta)} \cdot c^{|\beta|}, \quad (6)$$

where  $|\beta|_* = \sum_{i \in \beta} i$  and  $\nu(\beta) = \sum_{i \in \beta} \lceil \frac{i}{2} \rceil$ . Lemma III.1 then follows by essentially summing the above bound over all component types (the proof is presented in Section 4.8 of the full version).

The result of Lemma III.1 can then be seen to imply that the chance of successfully guessing the parity of any cycle is  $o(1)$  whenever  $c = \ell^{o(\ell)}$ , and specifically that the STREAMINGCYCLES problem requires  $\ell^{\Omega(\ell)}$  space.

To establish (6), we bound the (expected) evolution of  $\mathbf{H}_\beta^t$  as a function of  $t \in [n]$ . Specifically, we show in Section 4.7 of the full version that for every  $t \in [n]$  the expectation of  $\mathbf{H}_\beta^t$  can be upper bounded in terms

of expectations of  $\mathbf{H}_\alpha^s$  for  $s < t$  and  $\alpha$  corresponding to “subsets” of  $\beta$ . This is a natural extension of our analysis of the “sampling” protocol above. In full generality, however, this requires showing that if the algorithm has limited information about parities of collections of type  $\alpha$  at time  $s$  (i.e.,  $\mathbf{H}_\alpha^s$  is small), then it is unlikely to know too much about collections of type  $\beta$  obtained as a result of merging several components in  $\alpha$  or extending them by edges arrived between  $s$  and  $t$ .

Crucially, the probability that a collection of components of type  $\alpha$  grows into a collection of components of type  $\beta$  at any given time depend only on the *collection type* (i.e., the multisets  $\alpha$  and  $\beta$ ). Specifically, for  $s \in [n]$ , a pair of collection types  $\alpha, \beta \in \mathbb{Z}_+^{\{\cdot\}}$  such that  $\alpha[1] = \beta[1]$  (the number of single edge components in  $\alpha$  and  $\beta$  is the same) and a realization  $B_s$  of the board  $\mathbf{B}_s$  at time  $s$  we write

$$p_s(\alpha, \beta, B_s) = \Pr_{\mathbf{B}_{s+1}} [z \cdot 1 \sim_{s+1} \beta | \mathbf{B}_s = B_s]$$

for any  $z \in \{0, 1\}^s$  such that  $z \sim_s \alpha$  to denote the probability that a collection of type  $\alpha$  at time  $s$  becomes a collection of type  $\beta$  at time  $s+1$  through one of the following “growth events”:

**Extension** An edge arrives at time  $s+1$  that is incident to exactly one component in the collection.

**Merge** An edge arrives at time  $s+1$  that is incident to two components in the collection.

We will use

**Lemma III.2** (Informal version of Lemma 4.18 of the full version). *With high probability over the board state  $\mathbf{B}_s$ , for  $s$  not too close to  $n$  one has for (almost) every  $\alpha, \beta$*

$$p_s(\alpha, \beta, \mathbf{B}_s) \leq \frac{O(\alpha[a])}{n}$$

*if  $\alpha \rightarrow \beta$  is an extension of a path of size  $a$  and*

$$p_s(\alpha, \beta, \mathbf{B}_s) \leq \frac{O(\alpha[a] \cdot \alpha[b])}{(n-s)^2}$$

*if  $\alpha \rightarrow \beta$  is a merge of paths of size  $a$  and  $b$ .*

We will also need

**Definition III.3** (Down set of  $\beta \in \mathbb{Z}_+^{\{\cdot\}}$ ). *For  $\alpha, \beta \in \mathbb{Z}_+^{\{\cdot\}}$  we write  $\alpha \in \beta-1$  if  $\beta$  can be obtained from  $\alpha$  by either an extension or a merge followed by possibly adding an arbitrary number of 1’s to  $\alpha$ .*

*For  $\beta \in \mathbb{Z}_+^{\{\cdot\}}$  and  $\alpha \in \beta-1$  we write  $|\beta-\alpha|$  to denote the number of ones that need to be added to  $\alpha$  after a merge or extension to obtain  $\beta$ .*

Equipped with the above, and writing  $T$  for the set of all edge arrival times, we can state our main bound on the evolution of Fourier coefficients:

**Lemma III.4.** For every  $t \in T$  one has for  $\beta \in \mathbb{Z}_+^{\ell}$  that contain at least one component of size more than 1 and have  $|\beta|_* \leq \ell - 2$ ,  $\mathbb{E}_{\mathbf{X}, \mathbf{B}_t} [\mathbf{H}_\beta^t]$  is at most

$$\sum_{s=1}^{t-1} \sum_{\alpha \in \beta-1} q(|\beta - \alpha|) \cdot \mathbb{E}_{\mathbf{X}, \mathbf{B}_s} [\mathbf{H}_\alpha^s \cdot p(\alpha, \beta, \mathbf{B}_s)]$$

and for  $\beta$  with all components of size 1

$$\mathbb{E}_{\mathbf{X}, \mathbf{B}_t} [\mathbf{H}_\beta^t] \leq q(|\beta|),$$

where

$$q(k) = \begin{cases} \left(\frac{8c}{k}\right)^k & \text{if } k \leq c \\ 2^c & \text{otherwise.} \end{cases}$$

The function  $q(k)$  comes from hypercontractivity (as in (2), [30]), and bounds the total amount of Fourier mass a  $c$ -bit message can place on size- $k$  parities; when  $\alpha \rightarrow \beta$ ,  $q(|\beta - \alpha|)$  appears because the term involves remembering  $|\beta - \alpha|$  of the isolated edges that arrive between  $s$  and  $t$ .

The proof of Lemma III.4 is based on a function  $\mathbf{r}$  that functions similarly to the  $\mathbf{r}_{s,t}$  used in the Section II warm-up. Lemma III.4 allows us to bound the Fourier mass on various collections of components as a function of their evolution in the stream. We consider two prototypical examples now.

**Example 1: single-seed components.** To obtain some intuition for Lemma III.4, we first consider a simplified setting where  $p(\alpha, \beta, \mathbf{B}_s) = 0$  unless  $\alpha \rightarrow \beta$  is an extension, i.e., we ignore the effect of merges. Without merges, since we eventually care about the single-element set  $\{ \ell \}$ , we only need to track the mass on other single-element sets  $\beta = \{a\}$ , so  $\beta - 1 = \{a - 1\}$ . Then  $\mathbb{E}_{\mathbf{X}, \mathbf{B}_t} [\mathbf{H}_\beta^t]$  is at most

$$\begin{aligned} & \sum_{s=1}^{t-1} \sum_{\substack{\alpha \in \beta-1 \\ \alpha \rightarrow \beta \text{ is an extension}}} q(|\beta - \alpha|) \cdot \mathbb{E}_{\mathbf{X}, \mathbf{B}_s} [\mathbf{H}_\alpha^s \cdot p(\alpha, \beta, \mathbf{B}_s)] \\ &= \sum_{s=1}^{t-1} q(0) \mathbb{E}_{\mathbf{X}, \mathbf{B}_s} [\mathbf{H}_{\{a-1\}}^s \cdot p(\{a-1\}, \{a\}, \mathbf{B}_s)] \\ &\leq \sum_{s=1}^{t-1} 1 \cdot \frac{O(1)}{n} \mathbb{E}_{\mathbf{X}, \mathbf{B}_s} [\mathbf{H}_{\{a-1\}}^s]. \end{aligned}$$

where the last step uses the first bound from Lemma III.2. One observes that the above recurrence is quite similar to (5) and can be upper bounded similarly by  $c \frac{2^{O(a)}}{a!} (t/n)^a$ —as needed for (6).

**Example 2: multi-seed components.** To illustrate the way Lemma III.4 handles merges, suppose we want to

bound  $H_{\{4\}}$ . There are three paths from  $\{4\}$  via down-set relations to our base cases:

$$\begin{aligned} \{4\} &\rightarrow \{3\} \rightarrow \{2\} \rightarrow \{1\} \\ \{4\} &\rightarrow \{3\} \rightarrow \{1, 1\} \\ \{4\} &\rightarrow \{2, 1\} \rightarrow \{1\} \end{aligned}$$

The first is a series of extensions, so bounded by about  $c/a!$  according to Example 1; the other two involve merges, and we show give negligible contribution. We show this for the  $\{4\} \rightarrow \{2, 1\} \rightarrow \{1\}$  path here.

The only path to  $\beta = \{2, 1\}$  is an extension from  $\{1\}$ , so Lemma III.4 and Lemma III.2 show that  $\mathbb{E}_{\mathbf{X}, \mathbf{B}_t} [\mathbf{H}_\beta^t]$  is at most

$$\begin{aligned} & \sum_{s=1}^{t-1} \sum_{\alpha \in \beta-1} q(|\beta - \alpha|) \cdot \mathbb{E}_{\mathbf{X}, \mathbf{B}_s} [\mathbf{H}_\alpha^s \cdot p(\alpha, \beta, \mathbf{B}_s)] \\ &= \sum_{s=1}^{t-1} q(1) \cdot \mathbb{E}_{\mathbf{X}, \mathbf{B}_s} [\mathbf{H}_{\{1\}}^s \cdot p(\{1\}, \{2, 1\}, \mathbf{B}_s)] \\ &\leq \sum_{s=1}^{t-1} 8c \cdot \frac{O(1)}{n} \cdot \mathbb{E}_{\mathbf{X}, \mathbf{B}_s} [\mathbf{H}_{\{1\}}^s] \\ &\lesssim c^2(t/n). \end{aligned}$$

Then the contribution to  $\mathbb{E}_{\mathbf{X}, \mathbf{B}_s} [\mathbf{H}_\beta^t]$  for  $\beta = \{4\}$  from the  $\{4\} \rightarrow \{2, 1\} \rightarrow \{1\}$  path is at most

$$\sum_{s=1}^{t-1} q(|\{4\} - \{2, 1\}|) \cdot \mathbb{E}_{\mathbf{X}, \mathbf{B}_s} [\mathbf{H}_{\{2,1\}}^s \cdot p(\{2, 1\}, \{4\}, \mathbf{B}_s)]$$

which is bounded by

$$\begin{aligned} & \sum_{s=1}^{t-1} 1 \cdot O(c^2(s/n)) \cdot \frac{O(1)}{(n-s)^2} \lesssim c^2 \sum_{s=1}^{t-1} \frac{1}{(n-s)^2} \\ & \approx \frac{c^2}{n-t} \end{aligned}$$

For almost the entire stream, say  $t < n - n^{0.99} < n - \omega(c)$ , this term is much less than the  $\Theta(c)$  contribution from extensions. The fact that merges are about a  $c/n$  factor less likely can be used in general to show that the evolution is ultimately dominated by extensions and establish (6), for  $t < n - n^{0.99}$ .

*f) Handling the end of the stream.:* Our bound (6) gives roughly a  $\frac{c}{a!}$  bound on  $H_{\{a\}}^t$ , but only up to time  $t = n - n^{0.99}$ . By this time, however, probably every single cycle will be missing only  $O(1)$  edges, and so have at most  $O(1)$  components. Thus each cycle will have an  $\Omega(\ell)$ -long component that has arrived, and (6) gives a  $\frac{c^{O(1)}}{\Omega(\ell)!} \ll \varepsilon / \text{poly}(\ell)$  bound for the Fourier mass of its collection type at  $t$ . Summing over the  $\text{poly}(\ell)$  possible types leads to Lemma III.1.

We now discuss a key technical insight that allows us to establish Lemma III.4. It is the one illustrated in the warm-up example of Section II.

*g) Key tool in proving Lemma III.4: decomposition of a typical message.*: In order to establish Lemma III.4, we need an approach to expressing the Fourier transform of the typical message  $\mathbf{F}_t$  at time  $t$  in terms the Fourier transform of the typical messages  $\mathbf{F}_s$  for  $s < t$ . This is achieved by Lemma III.5 below. Intuitively, this lemma allows us to exploit communication bottlenecks arising at every  $s < t$  that preclude various Fourier coefficients from becoming large.

Let  $\mathbf{r}(x_{[s+1:t]}; F_s, F_t)$  denote the indicator function for  $x_{[s+1:t]}$  taking  $F_s$  to  $F_t$ . Note that  $\mathbf{r}$  is a random function depending only on  $\mathbf{B}_t$  ( $F_s$  is a function on  $s$  bits and  $F_t$  on  $t$  bits, so its dependence on  $\mathbf{B}_t$  is given implicitly by its arguments). The decomposition of typical messages is given by

**Lemma III.5.** *For every  $s, t \in T$  with  $s < t$ , and any  $z \leq t$ , we have:*

$$\tilde{\mathbf{F}}_t(z \leq t) = \mathbb{E}_{\mathbf{F}_s} \left[ \tilde{\mathbf{F}}_s(z \leq s) \cdot \tilde{\mathbf{r}}(z_{[s+1:t]}; \mathbf{F}_s, \mathbf{F}_t) \middle| \mathbf{F}_t, \mathbf{B}_t \right].$$

Since  $\mathbf{r}$  is a function partitioning its input based on at most  $2c$  bits, its normalized Fourier transform  $\tilde{\mathbf{r}}$  is subject to a bound similar to (2), which enables us to establish Lemma III.4.

#### A. Hidden-Batch Random Order Streams

We introduce a new random order streaming model that allows for (limited) correlations whose structure is unknown to the algorithm (the hidden-batch random order streaming model, Definition III.6 below). We give new algorithms for estimating local graph structure using small space in this model, and show that existing results in this space translate to our model with only a very mild loss in parameters.

**Definition III.6** (Hidden-batch random order stream model; informal). *In the  $(b, w)$ -hidden-batch random order stream model the edges of the input graph  $G = (V, E)$  are partitioned adversarially into batches of size bounded by  $b$ , after which every batch is presented to the algorithm in a time window of length  $w \geq 0$  starting at a uniformly distributed time in the interval  $[0, 1]$ .*

To motivate this model consider observing, say, a network traffic stream or a stream of friendships in a social network. In each case, there are many events (say, a login attempt, or a group of people meeting each other at a party) that will trigger a bounded number of updates (the back and forward of packets in a login protocol, or people adding friends they met) that might

have very complicated temporal correlations with each other, but that occur over a bounded period and are mostly independent of other events being observed in the stream.

To simulate this, we think of the division of observations into events (our batches) being adversarial but limited by a maximum batch size  $b$ , while the times of the events are chosen at random but the observations associated with the events are adversarially distributed about the event time, subject to the event duration limit  $w$ . Note that this means that observations from multiple events may (and often will be) interleaved—more than one person may be logging onto the same network at the same time and more than one party may be taking place at once.

It is worth stressing that the partitioning of edges into batches is unknown to the algorithm (consequently, we refer to our model as the *hidden batch model*).

Some existing random order streaming algorithm can be readily ported to the hidden-batch random order streaming model.

**Theorem III.7** (Component Collection; informal version of Theorem 5.2 of the full version). *There is a  $(b, w)$ -hidden batch streaming algorithm that, if at least a  $\Omega(1)$  fraction of the vertices of  $G$  are in components of size at most  $\ell$ , returns a vertex in  $G$  and the component containing it with probability  $9/10$  over its internal randomness and the order of the stream, using  $\ell^{O(\ell)}(b + wm)$  polylog  $n$  bits of space.*

We show that results of [5] on counting connected components in random graph streams can be easily extended to our hidden-batch random order model with only a mild loss in parameters. Specifically, in [5] it was shown that the number of connected components  $c(G)$  in a graph  $G$  can be approximated up to an  $\varepsilon n$  additive term using  $(1/\varepsilon)^{O(1/\varepsilon^3)}$  words of space. We show that this can be improved to  $(1/\varepsilon)^{O(1/\varepsilon)}$  and that (up to log factors) it can be extended to hidden-batch streaming with only linear loss in the parameters.

**Theorem III.8** (Counting Components; informal version of Theorem 5.1 of the full version). *For all  $\varepsilon \in (0, 1)$ , there is a  $(b, w)$ -hidden batch streaming algorithm that achieves an  $\varepsilon n$  additive approximation to  $c(G)$  with  $9/10$  probability, using  $(1/\varepsilon)^{O(1/\varepsilon)}(b + wm)$  polylog( $n$ ) bits of space.*

We note that the  $w m$  term in the space complexity corresponds to the expected number of edges arriving in a given time window of length  $w$ . Since the arrival times of edges are adversarially chosen in a window of length  $w$  started at the arrival time of the corresponding batch,

it is natural to expect the algorithm to store these edges.

Our algorithm above, similarly to the approach of [5], proceeds by first sampling a few nodes in the graph uniformly at random, and then constructing connected components incident on those nodes explicitly. Such a sample of component sizes for a few vertices selected uniformly at random from the vertex set of the input graph can then be used to obtain an estimator for the number of connected components. The details are given in Section 5 of the full version.

#### IV. LOWER BOUNDS

In this section we prove our main result, which is a lower bound for STREAMINGCYCLES, even when the edge arrival order and bit labels are chosen uniformly at random.

**Theorem IV.1** (STREAMINGCYCLES Lower Bound). *For all constants  $\varepsilon > 0$ , solving the distributional version of the STREAMINGCYCLES( $n, \ell$ ) problem with probability at least  $2/3$  requires at least one player to send a message of size at least  $\min(\ell^{\Omega(\ell)}, n^{1-\varepsilon})$ .*

We use this to prove a lower bound for the component collection problem.

**Definition IV.2** (Component Collection). *In the  $(\beta, \ell)$  component collection problem, we are given a graph  $G$  as a stream of edges, with at least  $\beta|V(G)|$  of its vertices in components of size at most  $\ell$ , and we must return a vertex  $v \in V(G)$  and the size of the component containing  $v$ .*

Our lower bound is given by Theorem IV.3:

**Theorem IV.3** (Component Collection Lower Bound). *For all constants  $C, \varepsilon > 0$ , solving the  $(1, \ell)$  component estimation problem in the  $(2, 0)$ -batch random order streaming model with probability at least  $2/3$  requires at least  $\min(\ell^{\Omega(\ell)}, n^{1-\varepsilon})$  space.*

The theorem gives a tight lower bound for the  $(1, \ell)$  component collection problem in  $(2, 0)$ -hidden batch streams.

We also prove a lower bound for the random walk generation problem in random streams, which we define formally first. Our definition matches the one in [8].

**Definition IV.4** (Pointwise  $\varepsilon$ -closeness of distributions). *We say that a distribution  $p \in \mathbb{R}_+^{\mathcal{U}}$  is  $\varepsilon$ -close pointwise to a distribution  $q \in \mathbb{R}_+^{\mathcal{U}}$  if for every  $u \in \mathcal{U}$  one has*

$$p(u) \in [1 - \varepsilon, 1 + \varepsilon] \cdot q(u).$$

We now define the notion of an  $\varepsilon$ -approximate sample of a  $k$ -step random walk:

**Definition IV.5** ( $\varepsilon$ -approximate sample). *Given  $G = (V, E)$  and a vertex  $u \in V$  we say that  $(X_0, X_1, \dots, X_k)$  is an  $\varepsilon$ -approximate sample of the  $k$ -step random walk started at  $u$  if the distribution of  $(X_0, X_1, \dots, X_k)$  is  $\varepsilon$ -close pointwise to the distribution of the  $k$ -step walk started at  $u$  (see Definition IV.4).*

**Definition IV.6** (Random walk generation). *In the  $(k, s, \varepsilon, \delta)$ -random walk generation problem one must, given a graph  $G = (V, E)$  presented as a stream, generate  $s$  independent  $\varepsilon$ -approximate samples of the walk of length  $k$  in  $G$  started at a uniformly random vertex, with error bounded by  $\delta$  in the total variation distance.*

The work of [8] designs a primitive that outputs such walks using space  $(1/\varepsilon)^{O(k)} 2^{O(k^2)} s$ , with  $\delta = 1/10$ , say.

**Theorem IV.7** (Random Walk Generation Lower Bound). *There exists an absolute constant  $C > 1$  such that for sufficiently large  $k \geq 1$ , (1) solving the  $(k, 1, 1/10, 1/10)$  component estimation problem in the  $(2, 0)$ -batch random order streaming model requires at least  $\min(k^{\Omega(\sqrt{k})}, n^{0.99})$  space and (2) solving the  $(k, C4^k, 1/10, 1/10)$  random walk generation problem in the  $(2, 0)$ -batch random order streaming model requires at least  $\min(k^{\Omega(k)}, n^{0.99})$  space.*

#### ACKNOWLEDGEMENTS

Ashish Chiplunkar was partially supported by the Pankaj Gupta New Faculty Fellowship. John Kallaugher and Eric Price were supported by NSF Award CCF-1751040 (CAREER). Michael Kapralov was supported by ERC Starting Grant 759471.

John was also supported by Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. Also supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Accelerated Research in Quantum Computing program.

#### NOTICE

The following applies to John Kallaugher's authorship in this work: This article has been authored by an employee of National Technology & Engineering Solutions of Sandia, LLC under Contract No. DE-NA0003525 with the U.S. Department of Energy (DOE). The employee owns all right, title and interest in

and to the article and is solely responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan <https://www.energy.gov/downloads/doe-public-access-plan>.

## REFERENCES

- [1] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, G. L. Miller, Ed. ACM, 1996, pp. 20–29. [Online]. Available: <https://doi.org/10.1145/237814.237823>
- [2] M. Kapralov, S. Khanna, and M. Sudan, "Approximating matching size from random streams," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, C. Chekuri, Ed. SIAM, 2014, pp. 734–751. [Online]. Available: <https://doi.org/10.1137/1.9781611973402.55>
- [3] G. Cormode, H. Jowhari, M. 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, Vienna, Austria*, ser. LIPIcs, K. Pruhs and C. Sohler, Eds., vol. 87. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017, pp. 29:1–29:15. [Online]. Available: <https://doi.org/10.4230/LIPIcs.ESA.2017.29>
- [4] M. Monemizadeh, S. Muthukrishnan, P. Peng, and C. Sohler, "Testable bounded degree graph properties are random order streamable," in *44th International Colloquium on Automata, Languages, and Programming, IICALP 2017, July 10-14, 2017, Warsaw, Poland*, ser. LIPIcs, I. Chatzigiannakis, P. Indyk, F. Kuhn, and A. Muscholl, Eds., vol. 80. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017, pp. 131:1–131:14. [Online]. Available: <https://doi.org/10.4230/LIPIcs.ICALP.2017.131>
- [5] P. Peng and C. Sohler, "Estimating graph parameters from random order streams," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '18. USA: Society for Industrial and Applied Mathematics, 2018, pp. 2449–2466.
- [6] M. Kapralov, S. Mitrovic, A. Norouzi-Fard, and J. 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*, S. Chawla, Ed. SIAM, 2020, pp. 1753–1772. [Online]. Available: <https://doi.org/10.1137/1.9781611975994.107>
- [7] A. Czumaj, H. Fichtenberger, P. Peng, and C. 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*, ser. LIPIcs, J. Byrka and R. Meka, Eds., vol. 176. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2020, pp. 16:1–16:20. [Online]. Available: <https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2020.16>
- [8] J. Kallaugher, M. Kapralov, and E. Price, "Simulating random walks in random streams," in *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022 (to appear)*, 2022.
- [9] D. Gavinsky, J. Kempe, I. Kerenidis, R. Raz, and R. de Wolf, "Exponential separations for one-way quantum communication complexity, with applications to cryptography," in *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, D. S. Johnson and U. Feige, Eds. ACM, 2007, pp. 516–525. [Online]. Available: <https://doi.org/10.1145/1250790.1250866>
- [10] H. Esfandiari, M. T. Hajiaghayi, V. Liaghat, M. Monemizadeh, and K. 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, San Diego, CA, USA, January 4-6, 2015*, P. Indyk, Ed. SIAM, 2015, pp. 1217–1233. [Online]. Available: <https://doi.org/10.1137/1.9781611973730.81>
- [11] M. Bury and C. Schwiegelshohn, "Sublinear estimation of weighted matchings in dynamic data streams," in *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, ser. Lecture Notes in Computer Science, N. Bansal and I. Finocchi, Eds., vol. 9294. Springer, 2015, pp. 263–274. [Online]. Available: [https://doi.org/10.1007/978-3-662-48350-3\\_23](https://doi.org/10.1007/978-3-662-48350-3_23)
- [12] S. Assadi, S. Khanna, and Y. 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*, P. N. Klein, Ed. SIAM, 2017, pp. 1723–1742. [Online]. Available: <https://doi.org/10.1137/1.9781611974782.113>
- [13] A. McGregor and S. 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, New Orleans, LA, USA*, ser. OASICS, R. Seidel, Ed., vol. 61. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018, pp. 14:1–14:4. [Online]. Available: <https://doi.org/10.4230/OASICS.SOSA.2018.14>
- [14] M. Bury, E. Grigorescu, A. McGregor, M. Monemizadeh, C. Schwiegelshohn, S. Vorotnikova, and S. Zhou, "Structural results on matching estimation with applications to streaming," *Algorithmica*, vol. 81, no. 1, pp. 367–392, 2019. [Online]. Available: <https://doi.org/10.1007/s00453-018-0449-y>
- [15] A. McGregor and S. Vorotnikova, "Planar matching in streams revisited," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, ser. LIPIcs, K. Jansen, C. Mathieu, J. D. P. Rolim, and C. Umans, Eds., vol. 60. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 17:1–17:12. [Online]. Available: <https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2016.17>
- [16] R. Chitnis, G. Cormode, H. Esfandiari, M. Hajiaghayi, A. McGregor, M. Monemizadeh, and S. 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, Arlington, VA, USA, January 10-12, 2016*, R. Krauthgamer, Ed. SIAM, 2016, pp. 1326–1344. [Online]. Available: <https://doi.org/10.1137/1.9781611974331.ch92>
- [17] H. Esfandiari, M. Hajiaghayi, and M. Monemizadeh, "Finding large matchings in semi-streaming," in *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, C. Domeniconi, F. Gullo, F. Bonchi, J. Domingo-Ferrer, R. A. Baeza-Yates, Z. Zhou, and X. Wu, Eds. IEEE Computer Society, 2016, pp. 608–614. [Online]. Available: <https://doi.org/10.1109/ICDMW.2016.0092>
- [18] E. Verbin and W. 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, San Francisco, California, USA, January 23-25, 2011*, D. Randall, Ed. SIAM, 2011, pp. 11–25. [Online]. Available: <https://doi.org/10.1137/1.9781611973082.2>

- [19] M. Kapralov, S. Khanna, and M. 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*, P. Indyk, Ed. SIAM, 2015, pp. 1263–1282. [Online]. Available: <https://doi.org/10.1137/1.9781611973730.84>
- [20] M. Kapralov, S. Khanna, M. Sudan, and A. 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*, P. N. Klein, Ed. SIAM, 2017, pp. 1703–1722. [Online]. Available: <https://doi.org/10.1137/1.9781611974782.112>
- [21] M. Kapralov and D. Krachun, “An optimal space lower bound for approximating MAX-CUT,” in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, M. Charikar and E. Cohen, Eds. ACM, 2019, pp. 277–288. [Online]. Available: <https://doi.org/10.1145/3313276.3316364>
- [22] V. Guruswami, A. Velingker, and S. 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*, ser. LIPIcs, K. Jansen, J. D. P. Rolim, D. Williamson, and S. S. Vempala, Eds., vol. 81. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, pp. 8:1–8:19. [Online]. Available: <https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2017.8>
- [23] V. Guruswami and R. 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*, ser. LIPIcs, D. Achlioptas and L. A. Végh, Eds., vol. 145. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 5:1–5:12. [Online]. Available: <https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2019.5>
- [24] C.-N. Chou, S. Golovnev, and S. Velusamy, “Optimal streaming approximations for all boolean max-2csp and max-ksat,” in *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, 2020, pp. 330–341.
- [25] J. Kallaugher, M. Kapralov, and E. Price, “The sketching complexity of graph and hypergraph counting,” in *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, M. Thorup, Ed. IEEE Computer Society, 2018, pp. 556–567. [Online]. Available: <https://doi.org/10.1109/FOCS.2018.00059>
- [26] S. Assadi, G. Kol, R. R. Saxena, and H. Yu, “Multi-pass graph streaming lower bounds for cycle counting, MAX-CUT, matching size, and other problems,” in *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*. IEEE, 2020, pp. 354–364. [Online]. Available: <https://doi.org/10.1109/FOCS46700.2020.00041>
- [27] S. Assadi and V. N., “Graph streaming lower bounds for parameter estimation and property testing via a streaming XOR lemma,” *STOC*, 2021.
- [28] A. Chakrabarti, G. Cormode, and A. McGregor, “Robust lower bounds for communication and stream computation,” in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, C. Dwork, Ed. ACM, 2008, pp. 641–650. [Online]. Available: <https://doi.org/10.1145/1374376.1374470>
- [29] A. Chakrabarti, T. S. Jayram, and M. Patrascu, “Tight lower bounds for selection in randomly ordered streams,” in *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, S. Teng, Ed. SIAM, 2008, pp. 720–729. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1347082.1347161>
- [30] J. Kahn, G. Kalai, and N. Linial, “The influence of variables on boolean functions,” in *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, ser. SFCS ’88. USA: IEEE Computer Society, 1988, pp. 68–80. [Online]. Available: <https://doi.org/10.1109/SFCS.1988.21923>