Set Cover in the One-pass Edge-arrival Streaming Model

Sanjeev Khanna
Department of Computer and Information Science, University of Pennsylvania
Philadelphia, PA, US
sanjeev@cis.upenn.edu

Christian Konrad
Department of Computer Science, University of Bristol
Bristol, UK
christian.konrad@bristol.ac.uk

Cezar-Mihail Alexandru
Department of Computer Science, University of Bristol
Bristol, UK
ca17021@bristol.ac.uk

ABSTRACT

We study the Set Cover problem in the one-pass edge-arrival streaming model. In this model, the input stream consists of a sequence of tuples (S, u), indicating that element u is contained in set S. This setting captures the streaming Dominating Set problem and is more general and harder to solve than the Set Cover set-arrival setting, where entire sets with all their elements arrive in the stream one-by-one.

We prove the following results (n is the size of the universe, m is the number of sets):

(1) A work by [Khanna, Konrad, ITCS'22] on streaming Dominating Set implies a one-pass $\tilde{O}(\sqrt{n})$-approximation algorithm with space $\tilde{O}(m)$ for edge-arrival Set Cover in adversarially ordered streams. We show that this space bound is best possible up to poly-log factors in that every α-approximation algorithm, for $\alpha = \Omega(\sqrt{n})$, requires space $\tilde{\Omega}(\frac{mn^2}{\alpha^4})$ in adversarially ordered streams, even if the algorithm is only required to output an α-approximation of the size of an optimal cover.

(2) As our main result, we give a one-pass $\tilde{O}(\sqrt{n})$-approximation algorithm with space $\tilde{O}(\frac{m}{\sqrt{n}})$ for edge-arrival Set Cover in random order streams. This result together with the lower bound mentioned above establishes a strong separation between the adversarial and random order settings.

(3) Finally, in adversarial order streams, we show that non-trivial algorithms with space $o(m)$ can be achieved at the expense of increased approximation factors $\tilde{\Omega}(\sqrt{n})$, which is in contrast to the set-arrival setting, where space $\tilde{O}(n)$ is enough for a $\Theta(\sqrt{n})$-approximation, and space $\Omega(n)$ is needed for an $o(n/\log n)$-approximation. We give an α-approximation algorithm for one-pass edge-arrival Set Cover with space $\tilde{O}(\frac{mn}{\alpha^2})$, for every $\alpha = \tilde{\Omega}(\sqrt{n})$.

CCS CONCEPTS

• **Theory of computation → Streaming models**; **Streaming, sublinear and near linear time algorithms**.

KEYWORDS

Streaming Algorithms, Set Cover, Random Order, Lower Bounds

PODS '23, June 18–23, 2023, Seattle, WA, USA

ACM ISBN 979-8-4007-0127-6/23/06.
https://doi.org/10.1145/3584372.3588678

ACM Reference Format:
Sanjeev Khanna, Christian Konrad, and Cezar-Mihail Alexandru. 2023. Set Cover in the One-pass Edge-arrival Streaming Model. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '23), June 18–23, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3584372.3588678

1 INTRODUCTION

The advent of Big Data has fueled the need for algorithms that are able to process huge quantities of data while maintaining a memory that is much smaller than the size of the input. Data streaming algorithms fulfil this role and have received significant attention since more than two decades. A *data streaming algorithm* processes its input sequentially in a single pass (or few passes) and uses a memory of size sublinear in the input size. We are interested in how well fundamental problems can be solved in this model, in particular, how the space requirements of such algorithms depend on the desired solution quality, as well as on various other aspects of the model, such as the arrival order of the input data or the number of passes.

In this paper, we consider the Set Cover problem in the one-pass streaming model. In Set Cover, we are given a universe $\mathcal{U}$ of size n and a family $\mathcal{S} = \{S_1, \ldots, S_m\}$ of m subsets $S_i \subseteq \mathcal{U}$, $i \in [m]$, of the universe $\mathcal{U}$. The objective is to output a smallest subset $\mathcal{T} \subseteq \mathcal{S}$ that *covers* the entire universe, i.e., such that $\bigcup_{S \in \mathcal{T}} S = \mathcal{U}$, and a *cover certificate* $C : \mathcal{U} \to \mathcal{T}$, indicating for each element u a set in $\mathcal{T}$ that covers/contains u. We will consider approximation algorithms for Set Cover. We say that a (streaming) algorithm is an α-approximation algorithm if it outputs a cover of size at most α times the size of a smallest set cover.

Set Cover in the Set-arrival Model. Set Cover in the so-called set-arrival model has been extensively studied in the literature [1, 4, 10, 12, 13, 15, 22]. In the *set-arrival model*, a streaming algorithm sees a sequence of the input sets in arbitrary order, where each set arrives together with *all* its elements. The one-pass setting in the set-arrival model is fully understood: For any $\alpha = o(\sqrt{n})$, results by Assadi, Khanna, and Li [4] show that space $\tilde{\Theta}(\frac{mn}{\alpha})$[1] is necessary and sufficient. It is also known that an $\Theta(\sqrt{n})$-approximation can be computed with space $\tilde{O}(n)$ [10, 13]. Together, these results show that set-arrival Set Cover undergoes a phase transition at approximation factor $\alpha = \Theta(\sqrt{n})$, namely, $\tilde{O}(n)$ space is sufficient for $O(\sqrt{n})$-approximation but $\tilde{\Theta}(\frac{mn}{\alpha})$ space is needed for $\alpha = o(\sqrt{n})$. Furthermore, it is not hard to see that the lower bound for streaming Dominating Set by [19] also applies to set-arrival Set Cover and

[1]We write $\tilde{O}$, $\tilde{\Theta}$, and $\tilde{\Omega}$ to mean O, Θ and Ω, respectively, where poly log factors are suppressed.

shows that space $\Omega(n)$ is necessary for any approximation factor $o(n/\log n)$.

Set Cover in the Edge-arrival Model. In this paper, we study Set Cover in the one-pass *edge-arrival model*. In this model, the input stream consists of a sequence of tuples (S, u), indicating that element $u \in \mathcal{U}$ is contained in set S. Bateni et al. [6] were the first to consider this model and gave a p-pass $((1+\epsilon)\log n)$-approximation streaming algorithm with space $\tilde{O}(mn^{O(\frac{1}{p})} + n)$. The edge-arrival setting also appeared in a work by Indyk et al. [16] who observed that their multi-pass streaming algorithm for fractional Set Cover can also be implemented in the edge-arrival setting. Furthermore, Khanna and Konrad [19] studied the Dominating Set problem in the graph streaming model, which can be seen as a special case of edge-arrival Set Cover with $m = n$ sets. Their results imply the following algorithm:

Theorem 1 ([19] KK-algorithm). *There is a randomized one-pass $\tilde{O}(\sqrt{n})$-approximation streaming algorithm for edge-arrival Set Cover with space $\tilde{O}(m)$. We will refer to this algorithm as the KK-algorithm.*

Since the edge-arrival setting is more general than the set-arrival setting, lower bounds for Set Cover in the set-arrival setting, in particular, the $\tilde{\Omega}(mn/\alpha)$ space lower bound for $\alpha = o(\sqrt{n})$ by Assadi et al. [4], also applies to the edge-arrival setting. Furthermore, since the $\tilde{O}(\frac{mn}{\alpha})$-space algorithm by Assadi et al. can also be implemented in the edge-arrival setting (see the Appendix of [19] for details), the edge-arrival setting is also completely understood for $\alpha = o(\sqrt{n})$.

The KK-algorithm together with the $\tilde{\Omega}(\frac{mn}{\alpha})$ space lower bound by Assadi et al. imply that, similar to the set-arrival setting, Set Cover in the edge-arrival setting also undergoes a phase transition at approximation factor $\alpha = \tilde{\Theta}(\sqrt{n})$.

1.1 Our Results

While the regime $\alpha = o(\sqrt{n})$ is fully understood for edge-arrival Set Cover in the adversarial order setting, the space complexity for $\alpha = \Omega(\sqrt{n})$ is open. In our first result, we resolve the space complexity for $\alpha = \tilde{\Theta}(\sqrt{n})$ up to poly-logarithmic factors, showing that space $\tilde{\Omega}(m)$ is necessary, which renders the KK-algorithm (Theorem 1) best possible:

Theorem 2. *Let $\alpha \geq \sqrt{n}$. Then any randomized α-approximation one-pass streaming algorithm for edge-arrival Set Cover in adversarial order streams requires $\tilde{\Omega}(mn^2/\alpha^4)$ space, even if the algorithm is only required to output an α-approximation of the size of an optimal cover.*

The fact that our lower bound even holds for algorithms that only output an approximation of the optimal set cover size is a substantial strength. Indeed, the $\Omega(mn/\alpha)$ lower bound for $\alpha = o(\sqrt{n})$ by Assadi, Khanna, and Li [4] crucially relies on the fact that algorithms output a cover certificate, and it is an open problem whether this requirement can be lifted.

Next, as our main result, we give a one-pass $\tilde{O}(\sqrt{n})$-approximation streaming algorithm with space $\tilde{O}(\frac{m}{\sqrt{n}})$ for *random order streams*, i.e., streams where the arrival order of the tuples (S, u) is chosen uniformly at random. This result together with the lower bound stated in Theorem 2 establish a strong separation between the adversarial and random order settings.

Approx.	Space	Stream order	Ref.
$\alpha = o(\sqrt{n})$	$\tilde{\Theta}(\frac{mn}{\alpha})$	adversarial	[4]
$\alpha = \tilde{\Theta}(\sqrt{n})$	$\tilde{O}(m)$	adversarial	[19]
$\alpha = \tilde{\Omega}(\sqrt{n})$	$\tilde{\Omega}(\frac{mn^2}{\alpha^4})$ LB, $\tilde{O}(\frac{mn}{\alpha^2})$ UB	adversarial	**here**
$\alpha = \tilde{\Theta}(\sqrt{n})$	$\tilde{O}(\frac{m}{\sqrt{n}})$	random	**here**

Table 1: Set Cover in the one-pass edge-arrival model

Theorem 3. *Suppose that $m = \tilde{\Omega}(n^2) \cap \text{poly}(n)$. Then, there is a randomized one-pass $\tilde{O}(\sqrt{n})$-approximation streaming algorithm for edge-arrival Set Cover with space $\tilde{O}(\frac{m}{\sqrt{n}})$, when the input stream is in random order.*

Random order streams have received significant attention in the data streaming literature for a variety of problems, including matchings [2, 7, 17, 18, 20], ruling sets [3], frequency moments [8], and submodular maximization [14]. The random order model is considered to be a more realistic model than the worst-case (adversarial) order model since, in practice, data rarely arrives in the worst possible order. Regarding Set Cover, our paper is the first to study edge-arrival Set Cover in random order streams. In the set-arrival setting, it is known that the one-pass random order setting is not easier than the adversarial order setting for approximation factors $\alpha = o(\sqrt{n})$ [4].

Last, returning to the adversarial order setting, we show that the regime $\alpha = \tilde{\Omega}(\sqrt{n})$ is non-trivial in the edge-arrival setting, which is counter to the set-arrival setting, where space $\tilde{\Theta}(n)$ is enough for a $\Theta(\sqrt{n})$-approximation and necessary for an $o(n/\log n)$-approximation. We obtain the following result:

Theorem 4. *For any $\alpha = \tilde{\Omega}(\sqrt{n})$, there is a randomized one-pass streaming algorithm for Set Cover with expected approximation ratio α in the edge-arrival model with space $\tilde{O}(\frac{mn}{\alpha^2})$.*

While the lower bound of Theorem 2 and the upper bound of Theorem 4 match up to poly-logarithmic factors for $\alpha = \tilde{\Theta}(\sqrt{n})$, we leave it as an interesting open problem to close the gap between the two bounds for other values of $\alpha = \tilde{\Omega}(\sqrt{n})$.

Table 1 summarizes all results known on the edge-arrival setting.

1.2 Techniques

We describe below the techniques used behind our results. We will describe the techniques behind our main result, our random order streaming algorithm, last since this description builds upon the description of our adversarial order algorithm.

Lower Bound for Adversarial Order Streams. We first discuss our $\tilde{\Omega}(mn^2/\alpha^4)$ space lower bound for α-approximation algorithms.

Our lower bound is proved in the one-way multi-party communication setting, where each of overall t parties holds a portion of the input instance. The parties communicate via messages *in order*, that is, the first party sends a message M_1 to the second, who in turn sends a message M_2 to the third. This continues until the last party t has received a message M_{t-1} and then outputs the result of the protocol. It is well-known that a lower bound on the minimum message length of the longest individual message implies a space lower bound for one-pass streaming algorithms.

We first point out that, in order to prove lower bounds above $\tilde{\Theta}(n)$ for an approximation factor $\alpha = \Omega(\sqrt{n})$, we require $t = \Omega(\alpha^2/n)$ parties since there is a simple deterministic t-party protocol with approximation factor $2\sqrt{nt}$ and maximum message length $\tilde{O}(n)$ (omitted due to space restrictions).

Indeed, our lower bound construction uses $t = \Theta(\alpha^2 \log^2 n/n)$ parties and is a reduction from the t-party version of the Set-Disjointess problem. In t-party Set-Disjointess, each party holds a subset of a universe of size m. The parties are guaranteed that either their sets are all pairwise disjoint or there is a *unique* element that appears in all sets. The goal is to decide between these two cases, and it is known that every protocol solving t-party Set-Disjointness requires at least one message of size $\Omega(m/t^2)$.

We work with a family of random sets $T_1, \ldots, T_m$, each of size $\sqrt{nt}$, and partitions of each of these sets into t random subsets $T_i^1, \ldots, T_i^t$ of size $\sqrt{n/t}$ each such that $T_i = T_i^1 \dot\cup \ldots \dot\cup T_i^t$. Then, given a t-party Set-Disjointness instance $(S_1, \ldots, S_t)$ with $S_i \subseteq [m]$, every party p includes the partial set T_i^p into the Set Cover instance if and only if $i \in S_p$.

Observe that if the sets $(S_1, \ldots, S_t)$ are pairwise disjoint, then every set in the Set Cover instance is of size $\sqrt{n/t}$. On the other hand, if the sets $(S_1, \ldots, S_t)$ are uniquely intersecting, then there exists one set in the Set Cover instance of size $\sqrt{nt}$. Denote this set by T_j. We will argue that, since the parties cannot determine which is the case, most of the elements of T_j need to be covered using other partial sets $T_{j'}^p$. However, the sizes of these random sets are chosen such that with high probability, each partial set $T_{j'}^p$ can cover only $O(\log n)$ elements of T_j. Thus, an algorithm requires $\Omega(\sqrt{nt}/\log n)$ such sets to cover T_j while the single set T_j may in fact be part of the input. The approximation guarantee, therefore, is $\alpha = \Omega(\sqrt{nt}/\log n)$, which allows us to choose $t = \Theta(\alpha^2 \log^2 n/n)$ to yield a lower bound of $\Omega(m/t^2) = \Omega(mn^2/(\alpha^4 \log^4 n))$.

The implementation of this idea requires the last party to fork the execution of the algorithm in the reduction m times. In parallel run i, the set $\mathcal{U} \setminus T_i$ is added to the instance, which allows focusing on covering the elements T_i. See Section 3 for further details.

Algorithm for Adversarial Order Streams. We will now explain the ideas underlying our one-pass randomized α-approximation algorithm with space $\tilde{O}(mn/\alpha^2)$, for $\alpha = \tilde{\Omega}(\sqrt{n})$. Our algorithm constitutes an improvement over the KK-algorithm by Khanna and Konrad [19]. We will first discuss the KK algorithm and then our improvements.

The key challenge in designing algorithms for edge-arrival Set Cover is the fact that sets may be spread out over the input stream and algorithms cannot take decisions based on the entire content of a set. In the set-arrival setting where entire sets arrive in the stream, algorithms can greedily add sets to the solution if they cover enough not-yet covered elements. This strategy does not work here. The KK-algorithm provides a solution to this problem based on the use of *uncovered-degree* counters for the sets: Every tuple (S_i, u) arriving in the stream increments the uncovered-degree $d(S_i)$ of S_i if u is not yet covered by the algorithm. Intuitively, we only want to add a set to the solution if it covers enough yet-uncovered elements. However, since this information is not available at any one moment, a probabilistic inclusion process is used instead: Whenever the uncovered degree of a set reaches $i \cdot \sqrt{n}$, for any integral $i \geq 1$, the set is included in the solution with probability $2^i \cdot \frac{\sqrt{n}}{m}$, and, if included, covers then all elements contained in this set that arrive from this moment onward in the stream.

The key point of the analysis of the KK-algorithm is to show that the probabilistic inclusion process does not add too many sets to the solution. To see this, denote by $\mathcal{S}_i \subseteq \mathcal{S}$ the subset of *level-i sets*, which are those sets with uncovered-degree in $[i\sqrt{n}, (i+1)\sqrt{n})$ at the end of the stream. Then, according to the inclusion rule, we expect to add at least $|\mathcal{S}_i| \cdot 2^i \cdot \frac{\sqrt{n}}{m}$ level-i sets to the solution. In order for not too many sets to be included, this can only work if the number of level i sets decreases exponentially, and, indeed, it is shown in [19] that $\mathbb{E}\,|\mathcal{S}_i| \leq \frac{1}{2}\,\mathbb{E}\,|\mathcal{S}_{i-1}|$, for every i, which implies that each level only contributes $\tilde{O}(\sqrt{n})$ sets to the final solution.

The KK-algorithm requires space $\tilde{\Theta}(m)$ for storing the uncovered-degrees of all sets. To go below this space bound, we thus cannot maintain the uncovered-degrees of all sets. Instead, for each set, we maintain its current level instead of its uncovered-degree. This requires us to use a different *promotion strategy* for the sets to reach the next level since uncovered-degrees are no longer available. Whenever a tuple (S_i, t) arrives in the stream such that t is not yet covered, we increase the level of S_i with probability $\frac{1}{\alpha}$.

At a first glace, this strategy does not appear to allow us to decrease the space requirements since we need to maintain the current level of every set. However, we show that, for $\alpha = \tilde{\Omega}(\sqrt{n})$, only $\tilde{O}(mn/\alpha^2)$ sets reach the second level over the course of the algorithm. Hence, it is sufficient to explicitly store only the levels of those $\tilde{O}(mn/\alpha^2)$ sets that were promoted at least once, which achieves the desired space bound.

Algorithm for Random Order Streams. At a high-level, our random arrival order algorithm for set cover aims to simulate the KK algorithm [19], albeit this time by rotating sets through the memory in blocks of $O(m/\sqrt{n})$ sets at a time: at any given moment in time, the algorithm tracks arriving edges for only a predetermined collection of $O(m/\sqrt{n})$ sets. Intuitively speaking, even though each set effectively only sees an $O(1/\sqrt{n})$-fraction of the stream, the random arrival order of the stream should still make it possible to collect a statistical signal that highlights sets that can cover $\tilde{\Omega}(\sqrt{n})$ of yet uncovered elements. However, a number of challenges emerge in successfully implementing this high-level intuition. First, the length of the stream that needs to be examined to register such a signal for a set S depends on the number of new elements that S is able to cover. In particular, if this number is $\gamma\sqrt{n}$, it is crucial that any such set S is included in the solution by the time a $\tilde{O}(1/\gamma)$-fraction of the stream is consumed because otherwise, $\omega(\sqrt{n})$ elements that S could have covered would have already passed by in the stream. So the duration of the stream that should be assigned to processing a set needs to depend on how many elements it would end up covering in the solution. We address this by designing a family of algorithms that are run successively, where the first algorithm consumes only about a $\frac{1}{\sqrt{n}}$-fraction of the stream, and in general, the i_{th} algorithm consumes the next $\frac{2^i}{\sqrt{n}}$-fraction of the stream. This ensures that every *relevant* set gets detected in a timely manner, before too many of its incident edges have passed.

A second challenge emerges in ensuring that not too many sets are chosen in our implementation of the KK algorithm. With each successive level, the KK algorithm geometrically increases the rate at which a relevant set S (one that covers $\tilde{\Omega}(\sqrt{n})$ of yet uncovered elements) is sampled. It is then necessary that the number of sets considered for sampling at each successive level is geometrically decreasing. This goal is achieved in [19] via the following argument. If too many sets are being considered for sampling at level j, it must necessarily be the case that some *uncovered* element x is contributing many edges towards such sets. The original implementation of the KK algorithm where each set is being tracked at all times, naturally confers a *monotonicity* property that ensures that if a set S is chosen for sampling at a level j, then it must have been also chosen for sampling at all previous levels. The analysis in [19] then shows that with high probability, one of the sets containing the element x would have been sampled before level j starts, and hence element x would have been marked as covered when level j starts. Both these properties are somewhat difficult to ensure in our setting since the reduced $\tilde{O}(m/\sqrt{n})$ space requires working with very sparse information about the sets and elements, making it hard to ensure either monotonicity or coverage of elements that have many incident edges.

We ensure monotonicity by demanding an increasingly stronger statistical signal as we go from one level to the next. This ensures that if a set S is chosen for sampling at a level j, then even allowing for stochastic deviations, with high probability, it must have also been chosen at every previous level. However, an element x that would have been covered in the KK algorithm, may continue providing statistical signal for sampling other sets in our setting simply because while a set S containing the element x has been added to our solution, the actual edge (S, x) is yet to arrive in the stream. This can lead to sampling too many sets at level j. We overcome this by tracking another statistical signal with the goal of identifying elements that would have been covered in the KK algorithm in a timely manner. This task is made more difficult by the availability of $\tilde{O}(m/\sqrt{n})$ space. Nonetheless, we show that we can detect this early enough in the stream and optimistically mark such elements as covered, and hence prevent them from creating too many sets for sampling. These ideas together allow us to obtain a $\tilde{O}(\sqrt{n})$-approximation using only $\tilde{O}(m/\sqrt{n})$ space.

1.3 Further Related Work

Saha and Geetor [22] initiated the study of (set-arrival) Set Cover in the streaming model and gave a $O(\log n)$-approximation algorithm that makes $O(\log n)$ passes over the input and uses space $\tilde{O}(n)$. Emek and Rosén showed that an $O(\sqrt{n})$-approximation can be obtained in a single pass with the same amount of space. This has been extended to multiple passes by Chakrabarti and Wirth [10], who showed that an $O(n^{\frac{1}{p+1}})$-approximation can be achieved using p passes and space $\tilde{O}(n)$, as long as p is constant, which, as proved in their paper, is best possible.

Regarding algorithms that use substantially more space, results by Assadi [1] and Har-Peled et al. [15] show that, using a poly-logarithmic number of passes, an α-approximation can be achieved using space $\tilde{O}(mn^{\frac{1}{\alpha}})$ and this space bound is optimal.

Set Cover on massive inputs can be solved well in practice. Most practical approaches are based on efficient implementations of the Greedy Set Cover algorithm [11, 21, 23]. It is also known that the streaming algorithm of Emek and Rosén [13] only produces slightly larger covers as those produced by Greedy in practice, albeit using substantially less memory [5].

1.4 Outline

We present in Section 2 a graphical representation of Set Cover instances that is used throughout the paper. We then give our lower bound for adversarial order streams in Section 3. Next, we present our streaming algorithm for the random order setting in Section 4, and our algorithm for the adversarial order setting is presented in Section 5. Finally, we conclude in Section 6.

2 PRELIMINARIES

Throughout this paper, we assume that every element $u \in \mathcal{U}$ is contained in at least one set since otherwise the set cover instance would not be feasible.

We make use of a representation of the input Set Cover instance $(\mathcal{S}, \mathcal{U})$ with $m = |\mathcal{S}|$ and $n = |\mathcal{U}|$ as a bipartite graph. This representation is obtained by setting $\mathcal{S}$ and $\mathcal{U}$ as the two bipartitions of the graph, and for every set $S_i \in \mathcal{S}$ and every $u \in S_i$, we include the edge (S_i, u) in the graph. More formally, we define $G = (\mathcal{S}, \mathcal{U}, E)$ with $(S_i, u) \in E$ if and only if $u \in S_i$. Then, a cover $\mathcal{T} \subseteq \mathcal{S}$ of the input instance corresponds to a subset of vertices of the left bipartition whose neighborhood equals the entire right bipartition.

3 LOWER BOUND FOR ADVERSARIAL ORDER STREAMS

In this section, we will prove that every one-pass streaming algorithm for Set Cover in the edge-arrival model with approximation factor $\alpha \geq \sqrt{n}$ requires $\Omega(mn^2/(\alpha^4 \log^4 n))$ space.

Our lower bound is obtained by a reduction from the one-way t-party version of the Set Disjointness problem. In this problem, each party $1 \leq i \leq t$ holds a subset $S_i \subseteq \mathcal{U}$ of the universe $\mathcal{U} = [n]$. The players are promised that either all sets $S_1, \ldots, S_t$ are pairwise disjoint, i.e., $S_i \cap S_j = \emptyset$ holds, for every $i \neq j$, or the sets uniquely intersect, i.e., $|\bigcap_i S_i| = 1$ and $|S_i \cap S_j| = 1$, for every $i \neq j$, hold. The objective is to decide which is the case.

It is well-known that the one-way communication complexity of t-party Set Disjointness is $\Omega(n/t)$ [9], which implies that at least one of the $t-1$ messages involved needs to be of size $\Omega(n/t^2)$.

Theorem 5 ([9]). *Let $\mathcal{P}$ be a protocol that solves the t-party Set-Disjointness problem with error at most $1/4$. Then, at least one message in $\mathcal{P}$ is of size $\Omega(n/t^2)$.*

Our lower bound requires a family of sets with small pairwise intersection. We first establish the existence of such a set family in Lemma 1, and then give our lower bound in Theorem 2.

Lemma 1. *Let m, t, n be integers with $t \leq n$ and $m = O(poly(n))$. Then, there exists a family of sets $T_1, \ldots, T_m \subseteq [n]$, each of size $s = \sqrt{n \cdot t}$, and partitions of the sets such that, for every $1 \leq i \leq m$ and $1 \leq r \leq t$, $T_i = T_i^1 \dot\cup \ldots \dot\cup T_i^t$ and $|T_i^r| = \frac{s}{t} = \sqrt{n/t}$ holds, and*

for every i, j and r with $i \neq j$:

$$|T_i^r \cap T_j| = O(\log n) \ .$$

Proof. We will show that random sets $T_1, \ldots, T_m$ fulfil this lemma with non-zero probability, which implies the existence of such a set family.

To this end, suppose that each T_i is a random subset of $[n]$ of size $s = \sqrt{n \cdot t}$, and let $T_i = T_i^1 \dot\cup \ldots \dot\cup T_i^t$ be a random partition of T_i. Fix now indices i, j and r, with $i \neq j$, and let $T_j = \{x_1, \ldots, x_s\}$. Then:

$$\mathbb{E}[|T_i^r \cap T_j|] = \sum_{k=1}^{s} \Pr[x_k \in T_i^r] = s \cdot \frac{s/t}{n} = \frac{s^2}{n \cdot t} = 1 \ .$$

By Chernoff bounds, we obtain that $|T_i^r \cap T_j| = O(\log n)$ with probability $1 - \frac{1}{n^C}$, for any constant C. Taking a union bound over all indices i, j and r ($i \neq j$), we see that the previous statement holds for all such indices with probability $1 - m^2 \cdot t \cdot \frac{1}{n^C}$, which can be bounded from below by $1 - \frac{1}{n}$ by choosing C large enough and using the fact that $m = O(\text{poly}(n))$. □

Theorem 2. *Let $\alpha \geq \sqrt{n}$. Then, every α-approximation one-pass streaming algorithm for Set Cover in the edge-arrival model with error probability at most $1/(4m)$ uses space $\Omega(mn^2/\alpha^4 \log^4 n)$, even if the algorithm only outputs an α-approximation of the size of an optimal cover.*

Proof. Let t be an integer. Let $T_1, T_2, \ldots, T_m \subseteq [n]$ with $|T_i| = s = \sqrt{n \cdot t}$, for all i, be a set family as in the statement of Lemma 1, i.e., each set T_i can be partitioned into subsets $T_i^1, \ldots, T_i^t$, each of size s/t, such that, for any i, j, r with $i \neq j$, $|T_i \cap T_j^r| = O(\log n)$ holds.

Let $\mathcal{A}$ be an algorithm as in the statement of the theorem, and let $(S_1, \ldots, S_t)$ be an instance of t-party Set-Disjointness with $S_i \subseteq [m]$. The t parties use $\mathcal{A}$ to solve the instance. To this end, every party i includes all the partial sets T_b^i with $b \in S_i$ in the Set Cover instance. The parties then execute $\mathcal{A}$ on this instance by forwarding the memory state. More concretely, the first party runs $\mathcal{A}$ on the sets T_b^1, for every $b \in S_1$, and sends the memory state of the algorithm to the second party. Upon receiving the memory state of $\mathcal{A}$ from party $i-1$ ($i \geq 2$), party i continues the execution of $\mathcal{A}$ on the sets T_b^i, for every $b \in S_i$.

The last party behaves as follows. After having executed $\mathcal{A}$ on her share of the input, the party forks the execution of $\mathcal{A}$ and runs m parallel executions. In the parallel execution j, the last party continues the execution of $\mathcal{A}$ on set $\overline{T}_j = [n] \setminus T_j$.

Observe that if $(S_1, \ldots, S_t)$ uniquely intersect then there exists a cover of size 2 in the parallel run $j = \bigcap_i S_i$. This is because the sets T_j and $\overline{T}_j$ are part of the input in run j and form such a cover.

Suppose now that $(S_1, \ldots, S_t)$ are pairwise disjoint. Then, for any $1 \leq j \leq m$, an optimal cover in parallel run j is of size at least $\text{OPT}_0 = O(\frac{s - s/t}{\log n})$, since the s elements in T_j need to be covered, which can be achieved by the at most one set T_j^k, for some k, and then via other sets which, by the properties of the set family, have an intersection with T_j of size at most $O(\log n)$.

The last party generates the output as follows: If one of the parallel runs returns an estimate of the size of the optimal cover of at most $\text{OPT}_0 - 1$, then the party reports *uniquely intersecting*. If no such run exists then the party reports *pairwise disjoint*.

For the algorithm $\mathcal{A}$ to output a solution of size at most $\text{OPT}_0 - 1$ when there is a solution of size 2, it is required that its approximation factor α is such that:

$$\alpha \cdot 2 \leq \text{OPT}_0 - 1 = O(\frac{s - s/t}{\log n}) \ ,$$

which yields $\alpha = O(s/\log n) = O(\sqrt{n \cdot t}/\log n)$. We can thus choose a value $t = \Theta(\alpha^2 \log^2 n/n)$. Then, by Theorem 5, we obtain that $\mathcal{A}$ uses space $\Omega(\frac{m}{t^2}) = \Omega(\frac{mn^2}{\alpha^4 \log^4 n})$.

In order to have correctly invoked Theorem 5, we require that the probability that none of the parallel runs fail to be at least 3/4. Using the union bound, we see that this is achieved if the error probability of $\mathcal{A}$ is $\frac{1}{4m}$. □

In the full version of this paper, we give a t-party protocol with approximation factor $\alpha = 2\sqrt{nt}$ and maximum message length $\tilde{O}(n)$. The existence of such a protocol highlights the need for $t = \Omega(\alpha^2/n)$ parties in order to prove our lower bound.

Remark. While the lower bound above is stated for algorithms with a success probability of at least $1 - 1/4m$, we note that any algorithm $\mathcal{A}$ with success probability of at least 3/4 can be converted into an algorithm with a success probability of at least $1 - 1/(4m)$ by running $O(\log m)$ parallel copies of $\mathcal{A}$, and outputting the smallest answer over all runs. Thus we can also conclude an $\Omega(mn^2/(\alpha^4 \log^4 n \log m))$ space lower bound for algorithms that are only required to succeed with probability at least 3/4.

4 ALGORITHM FOR RANDOM ORDER STREAMS

In this section, we present our algorithm for random order streams. We will first present the algorithm and discuss some of its key properties in Subsection 4.1. Three key invariants that hold throughout the algorithm and the main theorem are then presented in Subsection 4.2. Due to space resitrictions, the proofs of two of these invariants are deferred to the appendix.

4.1 Algorithm

The key idea behind our algorithm (see Algorithm 1 for a listing) is to process the input sets $\mathcal{S}$ in $\sqrt{n}$ batches $\mathcal{S}_1, \ldots, \mathcal{S}_{\sqrt{n}}$, each of size $\frac{m}{\sqrt{n}}$, focusing on at most one batch at any one moment. This allows us to reduce the space complexity from $\tilde{O}(m)$ to $\tilde{O}(\frac{m}{\sqrt{n}})$. We assume that the number of sets m and the size of the universe n are known to the algorithm, and, w.l.o.g., we also assume that the input stream length N is known. This is without loss of generality for the following reason: First, we can assume that $N \geq \frac{m}{\sqrt{n}}$ since otherwise the entire stream would fit into memory. Furthermore, we also know that $N \leq m \cdot n$ since every set is of size at most n. We can therefore run a logarithmic number of executions of our algorithm in parallel, using the guess $2^i \frac{m}{\sqrt{n}}$ for the value N in run i. Since our algorithm is not sensitive to the exact value of N, the run with the guess closest to N will therefore produce a valid solution.

Algorithm 1 Random Order $\tilde{O}(\frac{m}{\sqrt{n}})$-space $\tilde{O}(\sqrt{n})$-approx. Alg.

Require: Integers m, n, N, large constant C, $\mathcal{S}$ is arbitrarily partitioned into subsets $\mathcal{S}_1, \ldots, \mathcal{S}_{\sqrt{n}}$

1: Sol $\leftarrow \{\}$ {output set cover to be computed}
2: *Throughout the algorithm:*
3: Keep track of marked (as covered) elements using $O(n)$ space
4: For every element, store first set that contains element using space $\tilde{O}(n)$
5: *Run epoch 0:*
6: Sample every set w.p. $p_0 = C \cdot \frac{\sqrt{n}}{m} \log m$, add to Sol if sampled
7: Detect and mark every element of degree $\geq 1.1\frac{m}{\sqrt{n}}$ by processing first $\Theta(\frac{\sqrt{n}N \log m}{m})$ edges of the stream (see Lemma 6)
8: *Run Algorithms* $\mathsf{A}^{(1)} \ldots \mathsf{A}^{(\frac{1}{2}\log(n) - 3\log\log(m) - 2)}$:
9: **for** $i \leftarrow 1 \ldots K := \frac{1}{2}\log(n) - 3\log\log(m) - 2$ **do**
10: $\tilde{Q} \leftarrow$ sample each element of $\mathcal{S}$ with probability $q_0 = \frac{1}{n}$ {Tracked special sets}
11: *Run epoch j of algorithm* $\mathsf{A}^{(i)}$:
12: **for** $j \leftarrow 1 \ldots \log m - \frac{1}{2}\log n$ **do**
13: $T \leftarrow \emptyset$ {Data structure for tracking}
14: $\tilde{Q}' \leftarrow \emptyset$ {Subsampled special sets of current epoch j}
15: *Run subepoch k of epoch j of algorithm* $\mathsf{A}^{(i)}$:
16: **for** $k \leftarrow 1 \ldots \sqrt{n}$ **do**
17: $C[S] \leftarrow 0$, for every $S \in \mathcal{S}_k$ {Counter for each set}
18: **for** $\ell_i := \frac{2^i \cdot N}{n \log m}$ times **do** {length of subepoch is ℓ_i}
19: Let (u, S) be next edge in stream
20: **if** $S \in$ Sol **then**
21: mark u, store S as the witness for covering u and **continue** with next edge in stream
22: **if** u marked **then continue** with next edge in stream
23: {u not yet marked and $S \notin$ Sol}
24: **if** $S \in \tilde{Q}$ **then** {S was a special set in epoch $j-1$ and sampled for tracking}
25: Add edge (u, S) to T
26: **if** $S \in \mathcal{S}_k$ **then**
27: $C[S] \leftarrow C[S] + 1$
28: **if** $C[S] = j \cdot \log^6(m)$ **then** {S is special, eligible for sampling into Sol and $\tilde{Q}'$}
29: Add S to Sol with probability $p_j = C \cdot \frac{2^j\sqrt{n}}{m} \log m$
30: Add S to $\tilde{Q}'$ with probability $q_j = \min\{\frac{2^j}{n}, 1\}$
31: Mark every yet unmarked u if T contains at least $1.085 \cdot \frac{m2^{i-1}}{n^2 \log m}$ edges incident to u
32: $\tilde{Q} \leftarrow \tilde{Q}'$ {update subsampled set of special elements}
33: *Cover yet uncovered elements in remainder of stream:*
34: **for** remaining edges (u, S) in stream **do**
35: **if** $S \in$ Sol and u does not have a covering certificate **then**
36: Mark u, store S as the witness for covering u
37: *Post-processing/Patching phase:*
38: For every element u that does not yet have a covering witness (this may include marked elements): Cover u using the first set that u was incident, see Line 4
39: **return** Sol and cover certificate

Our algorithm gradually adds sets to the initially empty set Sol (Line 1), which constitutes the output set cover when the algorithm terminates. In particular, a set that is added to Sol is never removed. The algorithm also maintains the set of *marked as covered* elements, i.e., elements that are either covered by one of the sets in Sol or that are likely to be covered at a later stage (Line 3). We also store, for each element, a *cover certificate*, i.e., a set that covers the element. Furthermore, for each element u, we remember the first set in the stream that covers u (Line 4). These sets are required in the post-processing stage in order to cover any elements that have not been covered over the course of the algorithm.

The algorithm then enters *epoch* 0. In epoch 0, the algorithm adds every set with probability $p_0 = C \cdot \frac{\sqrt{n}}{m} \log m$ to Sol (Line 6). This ensures that elements with degree $\Omega(\frac{m}{\sqrt{n}})$ are covered by one of these sets with high probability. We therefore next identify elements of such large degree by detecting their signal in only a small fraction of the stream and mark these elements as covered (Line 7). Observe that we have not yet necessarily observed an edge that covers such an element u, however, with high probability, such an edge will be observed at a later stage.

Next, the algorithm runs $K = \frac{1}{2}\log(n) - 3\log\log(m) - 2$ algorithms $\mathsf{A}^{(1)}, \mathsf{A}^{(2)}, \ldots, \mathsf{A}^{(K)}$ sequentially. Each algorithm $\mathsf{A}^{(i)}$ consists of $\log m - \frac{1}{2}\log n$ epochs, and each epoch of $\sqrt{n}$ subepochs. The $\sqrt{n}$ subepochs are used for processing the sets $\mathcal{S}_1, \ldots, \mathcal{S}_{\sqrt{n}}$ in turn. Each subepoch of algorithm $\mathsf{A}^{(i)}$ processes $\ell_i = \frac{2^i N}{n \log m}$ input edges, which implies that, overall,

$$\sum_{i=1}^{K} (\log m - \frac{1}{2}\log n) \cdot \sqrt{n} \cdot \ell_i \leq \sum_{i=1}^{K} \log(m) \cdot \sqrt{n} \cdot \frac{2^i N}{n \log m}$$
$$= \frac{N}{\sqrt{n}} \cdot \sum_{i=1}^{K} 2^i < \frac{N}{\sqrt{n}} \cdot 2^{\frac{1}{2}\log(n) - 3\log\log(m) - 1} = N/(2 \cdot \log^3 m) \quad (1)$$

input edges are processed in algorithms $\mathsf{A}^{(1)}, \mathsf{A}^{(2)}, \ldots, \mathsf{A}^{(K)}$. Epoch 0 runs on only $\Theta(\frac{\sqrt{n}N \log m}{m}) = o(N/\log^3 m)$ edges. Hence, when algorithm $\mathsf{A}^{(K)}$ finishes, only $N/(2 \cdot \log^3 m) + o(N/\log^3 m) \leq N/\log^3 m$ edges have been processed.

Algorithm $\mathsf{A}^{(i)}$ focuses on those sets that are incident to at least $\Omega(\frac{n}{2^i})$ yet uncovered elements. More specifically, in epoch j of algorithm $\mathsf{A}^{(i)}$, the algorithm counts the number of edges observed between every set S and the yet unmarked elements (Line 27). Since this step would require $\Omega(m)$ space, this is implemented via running $\sqrt{n}$ subepochs sequentially and only the $m/\sqrt{n}$ sets in $\mathcal{S}_r$ are considered in subepoch r. We call a set in epoch j *special* if we have observed at least $j \cdot \log^6 m$ edges between the set and yet unmarked elements. Each special set is then added to Sol with probability $p_j = 2^j \cdot p_0$ (Line 29) and to a set $\tilde{Q}'$ with probability $q_j = \frac{2^j}{n}$ (Line 30). Before illustrating the purpose of $\tilde{Q}'$, we give more intuition as to why the sampling probability $p_j = 2^j \cdot p_0$ for adding special sets to Sol is suitable. In Lemma 8, we will prove that the number of special sets in epoch j is bounded by $1.1\frac{m}{2^j}$ with high probability. Hence, since each of the special sets is included in Sol with probability $p_j = C \cdot \frac{2^j\sqrt{n}\log m}{m}$, with high probability, we

add only $\tilde{O}(\sqrt{n})$ sets to Sol in every epoch of every algorithm $A^{(i)}$, and, thus, Sol contains at most $\tilde{O}(\sqrt{n})$ sets when the last algorithm $A^{(K)}$ has finished.

We will next discuss the purpose of $\tilde{Q}'$. Recall that every special set in epoch j of algorithm $A^{(i)}$ is included in $\tilde{Q}'$ with probability $q_j = \frac{2^j}{n}$. The set $\tilde{Q}'$, which becomes set $\tilde{Q}$ in epoch $j+1$ (see Line 32), is used for *tracking*. Our analysis crucially relies on being able to identify elements that are incident to at least $1.1 \cdot \frac{m}{2^j\sqrt{n}}$ special sets in epoch j. Since special sets are included in Sol with probability $p_j = C \cdot \frac{2^j\sqrt{n}\log m}{m}$, such elements are covered by special sets that are added to Sol with high probability and we can thus mark these elements as covered (observe that this is very similar to epoch 0). On an intuitive level, once an element is marked, the element cannot continue to contribute to increasing the counters of sets, which implies that we expect fewer and fewer sets to become special as the algorithm proceeds. This observation also justifies that the inclusion probabilities double between epochs, i.e., $p_j = 2 \cdot p_{j-1}$. In order to identify elements of such high degree, we make use of the fact that the special elements in epoch j are a subset of the special elements in epoch $j-1$ with high probability (Lemma 5), a property implied by the fact that, for a set to be special in epoch $j+1$, we require to observe $\log^6 m$ edges towards yet unmarked elements more than in epoch j, and a probabilistic argument shows that it is extremely unlikely that a set exceeds the higher threshold in epoch $j+1$ but not the lower threshold in epoch j. We therefore take a uniform random sample of the special elements in epoch j (the set $\tilde{Q}'$, which becomes $\tilde{Q}$ in epoch $j+1$) and track/store all edges between the sampled sets and the edges of epoch $j+1$. We prove in Lemma 6 that this sample is indeed enough to detect elements that are incident to at least $1.1 \cdot \frac{m}{2^{j+1}\sqrt{n}}$ special elements in epoch $j+1$. These elements are then marked as covered in Line 31.

Once algorithm $A^{(K)}$ has finished, we process the remaining edges and mark yet unmarked elements if they are incident to sets in Sol and store their covering witnesses. In the post-processing step, we cover the yet uncovered elements at a rate of one set per element in order to make sure that we indeed output a legal cover (Line 38). We observe that some marked elements may not have found a covering witness prior to the post-processing stage. This can happen when a high degree element is marked as covered in Line 31, but all its incident edges towards sets in Sol have appeared earlier in the stream. Our analysis accounts for this via the notion of *missed edges*. We prove that, for every set S added to Sol, there are only $\tilde{O}(\sqrt{n})$ missed edges incident to S.

4.2 Analysis

We use N to denote the input stream length. We assume that the number of sets m is larger than $C \cdot n^2 \log^3 m$, for a sufficiently large constant C, and polynomially bounded (in n), i.e., $m = \text{poly } n$.

For any set S, we will denote by $N(S)$ the set of elements that are contained in S. Thus the stream contains edges of the form (S, x) for all $x \in N(S)$. For any set S, and a subset $X \subseteq [n]$, we will denote by (S, X) the set of all edges of the form (S, x) for $x \in X$.

Denote by $\text{Sol}^{(i)}$ the variable Sol at the moment when algorithm $A^{(i)}$ finished, and let $U^{(i)} \subseteq \mathcal{U}$ be the set of elements that are not covered by $\text{Sol}^{(i)}$, i.e., $U^{(i)} = \mathcal{U} \setminus \cup_{S \in \text{Sol}^{(i)}} S$. Note that even if an element x can be covered by a set $S \in \text{Sol}^{(i)}$, it may be that the edge (S, x) appeared in the stream before the set S was included in our solution; in this case the element x may remain marked as uncovered but x will not be included in $U^{(i)}$. We will refer to such edges as *missed edges*.

Our analysis relies on proving the three invariants **(I1)**, **(I2)**, and **(I3)** from which our main result is derived. We will next state these invariants, then state and prove our main result. For space reasons, Invariants **(I2)** and **(I3)** are proved in the appendix.

- **(I1):** At the end of $A^{(i)}$, with probability at least $1 - (i)/m^5$, for any set $S \notin \text{Sol}^{(i)}$, the set S can cover at most $(n/2^i) \cdot \log^9 m$ elements in $U^{(i)}$.
- **(I2):** If a set S is included in $\text{Sol}^{(i)}$ during the execution of $A^{(i)}$, then with probability at least $1-1/m^3$, there are $O(\sqrt{n}\log^9 m)$ missed edges that are incident on the set S only.
- **(I3):** With probability at least $1 - 1/m^3$, the total number of sets added to Sol during the execution of $A^{(i)}$ is $O(\sqrt{n}\log^2 m)$.

We note at this occasion that we have not attempted to minimize the poly-log factors appearing in our analysis.

Theorem 3. *Let $m = \tilde{\Omega}(n^2) \cap \text{poly}(n)$. Then, Algorithm 1 is a one-pass $\tilde{O}(\sqrt{n})$-approximation streaming algorithm for edge-arrival Set Cover in the random order setting that uses space $\tilde{O}(\frac{m}{\sqrt{n}})$ and succeeds with high probability over the random ordering of the input and the random coin flips of the algorithm.*

Proof. We first analyse the space requirements of the algorithm and then establish the approximation factor.

Space Analysis. We start by observing that for any i, all variables used by algorithm $A^{(i)}$ except $\tilde{Q}, \tilde{Q}', T$ and Sol are of size at most $\tilde{O}(m/\sqrt{n})$. We will see in Lemma 8 that the number of special sets in epoch j is at most $1.1\frac{m}{2^j}$ with high probability. Since the tracked sets in epoch j are the special sets from epoch $j-1$ subsampled with probability q_{i-1}, by Chernoff bounds, we track at most $\tilde{O}(q_{j-1}\frac{m}{2^{j-1}}) = \tilde{O}(\frac{m}{n})$ special sets in epoch j with high probability. This bounds the sizes of $\tilde{Q}'$ and $\tilde{Q}$ by $\tilde{O}(\frac{m}{n})$. Furthermore, we will prove in Lemma 6 that $|T| = \tilde{O}(\frac{m}{\sqrt{n}})$. Last, the algorithm can easily be modified so that $|\text{Sol}| \leq n$ always holds. Indeed, if $|\text{Sol}|$ reaches the size n then we report a trivial cover that covers every element with a single set (recall the algorithm stores for every element the first set that it is contained in). This establishes the space complexity of the algorithm.

Approximation Ratio Analysis. Assuming the three invariants **(I1)**, **(I1)**, and **(I3)** hold, we can show that, with probability at least $1 - 1/m$, our algorithm outputs a solution that is at most $O(\sqrt{n}\log^{12} m)$ times the size of the optimal solution. To see this, we first observe that by Invariant (**I3**), with probability at least $1-1/m^2$, the total number of sets added to Sol by $A^{(1)}, A^{(2)}, ..., A^{(K)}$, is bounded by $O(\sqrt{n}\log^3 m)$. We next observe that with probability at least $1 - 1/m^2$, by Invariant (**I2**), we have that for every set S included in Sol during the execution of the algorithms $A^{(1)}, A^{(2)}, ..., A^{(K)}$, there are only $O(\sqrt{n}\log^9 m)$ missed edges. Finally, by Invariant (**I1**), we know that when $A^{(K)}$ terminates, no

set outside of Sol can cover more than $O(\sqrt{n}\log^{12} m)$ elements in $U^{(K)}$. So during the patching phase, the total number of sets added by our algorithm is at most $O(\sqrt{n}\log^{12} m)$ times more than the number of sets used by an optimal solution.

Putting together, we obtain that with probability at least $1-1/m$, we output a solution whose size is at most $O(\sqrt{n}\log^{12} m) = \tilde{O}(\sqrt{n})$ times the optimal solution. □

4.3 Concentration Result

The random order assumption allows us to prove a concentration result that we will use throughout our analysis.

Lemma 2. *Let $I \subseteq [N]$ be a subset of positions in the stream with $|I| = \ell$. Let $S \in \mathcal{S}$ be any set and let $X \subseteq S$ be a fixed subset of S. Then, with probability at least $1 - \frac{1}{m^{20}}$, we have that the total number of edges of the form (S, x) with $x \in X$ that appear in the locations I is:*

1. *at least $0.99 \cdot \frac{\ell}{N}|X|$ and at most $1.01 \cdot \frac{\ell}{N}|X|$, if $|I| \leq 0.001N$ and $\frac{\ell}{N}|X| \geq C\log m$, for a large enough constant C;*
2. *at most $C\log(m) \cdot \max\{\frac{\ell}{N}|X|, 1\}$, for some large constant C, if $\ell \leq \frac{N}{2}$;*
3. *at least $\frac{\ell}{N}|X|(1-\frac{1}{\sqrt{n}}) - \log(m)\sqrt{\frac{\ell}{N}|X|(1-\frac{1}{\sqrt{n}})}$ and at most $\frac{\ell}{N(1-\frac{1}{\sqrt{n}})}|X| + \log(m)\sqrt{\frac{\ell}{N(1-\frac{1}{\sqrt{n}})}|X|}$, if $\ell \leq \frac{N}{\sqrt{n}}$ and $\frac{\ell}{N}|X| \geq \log^6 m$.*

Due to space restrictions, the proof of this lemma is deferred to the appendix.

4.4 Proof of Invariant (I1)

Lemma 3 (**Invariant (I1)**). *At the end of $\mathsf{A}^{(i)}$, with probability at least $1 - i/m^5$, for any set $S \notin \mathsf{Sol}^{(i)}$, the set S can cover at most $(n/2^i) \cdot \log^9 m$ elements in $U^{(i)}$.*

Proof. We will prove invariant (**I1**) by induction. At the start of $\mathsf{A}^{(i)}$, with probability at least $1-(i-1)/m^5$, we have that for any set $S \notin \mathsf{Sol}^{(i)}$, the set S can cover at most $(n/2^{i-1})\log^9 m$ elements. This is clearly true when the algorithm $\mathsf{A}^{(1)}$ starts as no set can cover more than n elements. Now suppose this invariant holds at the end of the algorithm $\mathsf{A}^{(i-1)}$. Assume by way of contradiction, that there is a set S such that S does not get sampled during $\mathsf{A}^{(i)}$ but S contains at least $(n/2^i) \cdot \log^9 m$ elements in $U^{(i)}$. Let $X = N(S) \cap U^{(i)}$; by our assumption, we have $|X| \geq (n/2^i) \cdot \log^9 m$.

Consider any epoch $\mathsf{E}_j^{(i)}$ (the jth epoch of $\mathsf{A}^{(i)}$) and let us analyze the sub-epoch of $\mathsf{E}_j^{(i)}$ that is devoted to the processing of set S. Let $I \subseteq [N]$ denote the set of indices (positions in the stream) during which we processed set S so far. Since $S \notin \mathsf{Sol}^{(i)}$, S has only been observed in epoch 0, in all subepochs of the algorithms $\mathsf{A}^{(1)}$ through $\mathsf{A}^{(i-1)}$ dedicated to processing S, in the first $(j-1)$ subepochs of $\mathsf{A}^{(i)}$ dedicated to processing S, and part of the j_{th} epoch of $\mathsf{A}^{(i)}$. We can thus bound $|I|$ by

$$\begin{aligned}|I| &\leq \frac{C\sqrt{n}N\log m}{m} + \left(\sum_{r=1}^{i-1}\log m \cdot \ell_r\right) + j \cdot \ell_i \\ &\leq \frac{N}{2\cdot\sqrt{n}} + \sum_{r=1}^{K}\log m \cdot \ell_r \\ &= \frac{N}{2\cdot\sqrt{n}} + \sum_{r=1}^{K}\log m \cdot \frac{2^r N}{n\log m} \\ &\leq \frac{N}{2\cdot\sqrt{n}} + \frac{2^{K+1}N}{n} \leq \frac{N}{\sqrt{n}}\,,\end{aligned}$$

where we used the bound $2^K = o(\sqrt{n})$ and $m = \Omega(n^2)$.

By Lemma 2, we know that with probability at least $1-1/m^{20}$, the total number of edges in $(S, N(S))$ that appear in I can be bounded by

$$1.01 \cdot \frac{|N(S)|}{N} \cdot |I| \leq 1.01 \cdot \frac{|N(S)|}{N} \cdot \frac{N}{\sqrt{n}} \leq 1.01 \cdot \frac{|N(S)|}{\sqrt{n}} \leq 1.01\sqrt{n}\,,$$

where we used the trivial bound $|N(S)| \leq n$.

Now note that the indices in I are the only locations where the algorithm has thus far observed edges incident on S. Thus if we fix the set J of all the locations in the stream where edges incident on S appear, and the set $Y \subseteq [n]$ of elements such that only edges in (S, Y) appear in the stream at indices in I, then all edges in $(S, N(S) \setminus Y)$ appear in a uniformly at random permutation on indices in $J \setminus I$. In particular, this means that all elements in $X \setminus Y$ appear as in a uniformly at random chosen permutation over indices in $J \setminus I$. We also note at this stage that the algorithm would process these elements since, as we show in Lemma 7, uncovered elements are not marked with high probability.

Again invoking Lemma 2, we know that with probability at least $1 - 1/m^{20}$, the total number of edges of the form (S, X) that appear in the sub-epoch of $\mathsf{E}_j^{(i)}$ devoted to processing set S can be bounded from below by

$$\begin{aligned}0.99 \cdot \frac{|X \setminus Y|}{N} \cdot \ell_i &= 0.99 \cdot \frac{(n/2^i)\cdot\log^9 m - 2\sqrt{n}}{N} \cdot \frac{2^i N}{n\log m} \\ &\geq 0.99\log^8 m - o(1)\,.\end{aligned}$$

Thus with probability at least $1 - 1/m^{20}$, during the sub-epoch of $\mathsf{E}_j^{(i)}$ devoted to processing the set S, the set S gets selected for sampling with probability p_j (since $0.99\log^8 m - o(1) \geq j \cdot \log^6 m$). Moreover, by taking a union bound over all epochs, we know that this assertion holds for set S in every epoch $\mathsf{E}_j^{(i)}$ with probability at least $1 - 1/m^{19}$. Thus conditioned on this event, the set S is guaranteed to get sampled with probability 1, and hence included in the solution before $\mathsf{A}^{(i)}$ terminates. By taking a union bound over all sets, we conclude that the probability that Invariant (**I1**) is violated during $\mathsf{A}^{(i)}$ is bounded by $1/m^{18}$, completing the proof. □

5 ALGORITHM FOR ADVERSARIAL ORDER STREAMS

We will now present our $\tilde{O}(mn/\alpha^2)$ space α-approximation streaming algorithm, for $\alpha = \tilde{\Omega}(\sqrt{n})$, for edge-arrival Set Cover.

Our algorithm is depicted in the listing of Algorithm 2. In this listing, we assume that we have a function Coin(p) to our disposal, which evaluates to 1 with probability p and to 0 with probability $1-p$. The algorithm proceeds as follows:

While processing the stream, the algorithm computes a partial cover consisting of sets $D_0, \ldots, D_{\log m}$, where D_i is the partial cover of level i. In parallel to the computation of these sets, for each element $u \in \mathcal{U}$, the algorithm stores an arbitrary set $R(u)$ that contains u (lines 9 and 10). Then, in a post-processing step, if an element u is not covered in the partial cover $\bigcup_i D_i$ then the set $R(u)$ is added to the cover certificate (line 25), thereby ensuring that the output indeed covers all elements.

Next, we will discuss how the sets D_i are computed. Every set is assigned a *level*, which is initialized to 0. The map L, defined in Line 3, stores the levels of all those sets whose level is at least 1. Then, whenever a tuple (S, u) arrives in the stream such that u is not yet covered, the level of S is increased by 1 with probability $\frac{1}{\alpha}$ in Line 18. Furthermore, whenever the level of a set is increased to ℓ, then the set is included in D_i with probability $p_\ell = \frac{\alpha^{2\ell+1}}{mn^\ell}$ in Line 21, which completes the description of the algorithm.

Algorithm 2 Single-pass Streaming Algorithm for Set Cover in the Edge-arrival Setting

Require: Bipartite input graph $G = (\mathcal{S}, \mathcal{U}, E)$ with $|\mathcal{S}| = m$ and $|\mathcal{U}| = n$

1: $D_1, D_2, \ldots, D_{\log m} \leftarrow \{\}$
2: for every $u \in \mathcal{U}$: $R(u) \leftarrow \perp$ {we store an arbitrary set that covers u, for every $u \in \mathcal{U}$}
3: $L \leftarrow$ empty map with keys in $\mathcal{S}$ and values in $[\log m]$ {for storing sets with level ≥ 1}
4: $U \leftarrow \emptyset$ {keep track of dominated nodes ($U \subseteq \mathcal{U}$ always holds)}
5: For every $u \in \mathcal{U} : C(u) \leftarrow \perp$ {output cover certificate}
6: Let $D_0 \subseteq \mathcal{S}$ such that every set is included in D_0 with probability $p_0 := \frac{\alpha}{m}$
7: **while** stream not empty **do**
8: Let (S, u) be the next edge in the stream
9: **if** $R(u) = \perp$ **then**
10: $R(u) \leftarrow S$ {store an arbitrary set that covers u}
11: **if** $u \in U$ **then** {ignore edge if incident to already covered element from $\mathcal{U}$}
12: Continue with next edge in stream
13: {element u is not yet covered}
14: $\ell \leftarrow 0$ {level of set S}
15: **if** $S \in L$ **then** {look up level in L}
16: $\ell \leftarrow L[S]$
17: **if** Coin$(\frac{1}{\alpha})$ **then**
18: $\ell \leftarrow \ell + 1$ {increase level of S}
19: $L[S] \leftarrow \ell$ {store level in L}
20: **if** Coin$\left(p_\ell := \frac{\alpha^{2\ell+1}}{mn^\ell} = \left(\frac{\alpha^2}{n}\right)^\ell p_0\right)$ **then**
21: $D_\ell \leftarrow D_\ell \cup \{S\}$
22: **if** $S \in \bigcup_{i \geq 0} D_i$ **then** {u is dominated by S}
23: $U \leftarrow U \cup \{u\}$
24: $C(u) \leftarrow S$
25: For every $u \in \mathcal{U} \setminus U : C(u) \leftarrow R(u)$
26: **return** Cover $\bigcup_{i=0}^{\log m} D_i \cup \{R(u) : u \in \mathcal{U} \setminus U\}$ and cover certificate C

Analysis

Due to space restrictions, we defer the analysis to the full version of this paper. We obtain the following theorem:

Theorem 4. *For any $\alpha \geq 2\sqrt{n}$, Algorithm 2 is a randomized one-pass semi-streaming algorithm for Set Cover in the edge-arrival model with expected approximation factor $O(\alpha \log m)$ and space requirements $\tilde{O}(\frac{mn}{\alpha^2})$.*

Remark. The space bound claimed in the previous theorem a priori only holds in expectation. However, this bound can easily be turned into a high probability bound or even into a worst-case bound at the expense of a marginal $o(1)$ increase in the expected approximation ratio.

Last, similar to [19], we could also turn the expected approximation guarantee into a high probability guarantee at the expense of an additional $\log m$ factor in the approximation ratio. See [19] for details.

6 CONCLUSIONS

In this paper, we showed that the $\tilde{O}(\sqrt{n})$-approximation $\tilde{O}(m)$ space KK-algorithm by Khanna and Konrad [19] for edge-arrival Set Cover in the adversarial order setting is optimal up to poly-logarithmic factors in that every α-approximation streaming algorithm requires space $\tilde{\Omega}(mn^2/\alpha^4)$. For the regime when $\alpha = \omega(\sqrt{n})$, we improved upon the KK-algorithm by showing that an α-approximation can be achieved in one pass using only $\tilde{O}(mn/\alpha^2)$ space. As our main result, we showed that the space barrier of $\tilde{\Theta}(m)$ for $\tilde{O}(\sqrt{n})$-approximation algorithms can be broken when the stream is in random order. For this setting, we gave a $\tilde{O}(\sqrt{n})$-approximation streaming algorithm that uses space $\tilde{O}(\frac{m}{\sqrt{n}})$.

We conclude with two natural open problems.

First, while our space lower bound of $\tilde{\Omega}(mn^2/\alpha^4)$ and the space bound of $\tilde{O}(mn/\alpha^2)$ required by our algorithm match for $\alpha = \tilde{\Theta}(\sqrt{n})$ up to poly-logarithmic factors, the bounds differ for larger values of α. Can we close the gap between the two bounds for all values of $\alpha = \tilde{\Omega}(\sqrt{n})$?

Second, our $\tilde{O}(\sqrt{n})$-approximation algorithm for random order streams uses space $\tilde{O}(\frac{m}{\sqrt{n}})$ and thus breaks the $\tilde{\Theta}(m)$ space bound for adversarial order streams. However, can we further reduce the space requirements in the random order setting or can prove a matching lower bound? We conjecture that the right answer is $\tilde{O}(\frac{m}{\sqrt{n}})$, that is, the space bound achieved here is optimal up to poly-logarithmic factors.

ACKNOWLEDGMENTS

Sanjeev Khanna is supported in part by NSF awards CCF-1934876 and CCF-2008305. Christian Konrad is supported by EPSRC New Investigator Award EP/V010611/1. Cezar-Mihail Alexandru is supported by EPSRC DTP studentship EP/T517872/1.

REFERENCES

[1] Sepehr Assadi. 2017. Tight Space-Approximation Tradeoff for the Multi-Pass Streaming Set Cover Problem. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, Emanuel Sallinger, Jan Van den Bussche, and Floris Geerts (Eds.). ACM, 321–335. https://doi.org/10.1145/3034786.3056116

[2] Sepehr Assadi and Soheil Behnezhad. 2021. Beating Two-Thirds For Random-Order Streaming Matching. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference) (LIPIcs, Vol. 198)*, Nikhil Bansal, Emanuela Merelli, and James Worrell (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 19:1–19:13.

[3] Sepehr Assadi and Aditi Dudeja. 2021. Ruling Sets in Random Order and Adversarial Streams. In *35th International Symposium on Distributed Computing (DISC 2021) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 209)*, Seth Gilbert (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 6:1–6:18. https://doi.org/10.4230/LIPIcs.DISC.2021.6

[4] Sepehr Assadi, Sanjeev Khanna, and Yang Li. 2021. Tight Bounds for Single-Pass Streaming Complexity of the Set Cover Problem. *SIAM J. Comput.* 50, 3 (2021). https://doi.org/10.1137/16M1095482

[5] Michael Barlow, Christian Konrad, and Charana Nandasena. 2021. Streaming Set Cover in Practice. In *Proceedings of the Symposium on Algorithm Engineering and Experiments, ALENEX 2021, Virtual Conference, January 10-11, 2021*, Martin Farach-Colton and Sabine Storandt (Eds.). SIAM, 181–192. https://doi.org/10.1137/1.9781611976472.14

[6] MohammadHossein Bateni, Hossein Esfandiari, and Vahab Mirrokni. 2017. Almost Optimal Streaming Algorithms for Coverage Problems. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures* (Washington, DC, USA) *(SPAA '17)*. Association for Computing Machinery, New York, NY, USA, 13–23. https://doi.org/10.1145/3087556.3087585

[7] Aaron Bernstein. 2020. Improved Bounds for Matching in Random-Order Streams. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference) (LIPIcs, Vol. 168)*, Artur Czumaj, Anuj Dawar, and Emanuela Merelli (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 12:1–12:13.

[8] Vladimir Braverman, Emanuele Viola, David P. Woodruff, and Lin F. Yang. 2018. Revisiting Frequency Moment Estimation in Random Order Streams. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 107)*, Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella (Eds.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 25:1–25:14. https://doi.org/10.4230/LIPIcs.ICALP.2018.25

[9] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. 2003. Near-Optimal Lower Bounds on the Multi-Party Communication Complexity of Set Disjointness. In *18th Annual IEEE Conference on Computational Complexity (Complexity 2003), 7-10 July 2003, Aarhus, Denmark*. IEEE Computer Society, 107–117. https://doi.org/10.1109/CCC.2003.1214414

[10] Amit Chakrabarti and Anthony Wirth. 2016. Incidence Geometries and the Pass Complexity of Semi-Streaming Set Cover. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, Robert Krauthgamer (Ed.). SIAM, 1365–1373. https://doi.org/10.1137/1.9781611974331.ch94

[11] Graham Cormode, Howard J. Karloff, and Anthony Wirth. 2010. Set cover algorithms for very large datasets. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, Jimmy Huang, Nick Koudas, Gareth J. F. Jones, Xindong Wu, Kevyn Collins-Thompson, and Aijun An (Eds.). ACM, 479–488. https://doi.org/10.1145/1871437.1871501

[12] Erik D. Demaine, Piotr Indyk, Sepideh Mahabadi, and Ali Vakilian. 2014. On Streaming and Communication Complexity of the Set Cover Problem. In *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8784)*, Fabian Kuhn (Ed.). Springer, 484–498. https://doi.org/10.1007/978-3-662-45174-8_33

[13] Yuval Emek and Adi Rosén. 2016. Semi-Streaming Set Cover. *ACM Trans. Algorithms* 13, 1 (2016), 6:1–6:22. https://doi.org/10.1145/2957322

[14] Moran Feldman, Paul Liu, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. 2022. Streaming Submodular Maximization Under Matroid Constraints. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 229)*, Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 59:1–59:20. https://doi.org/10.4230/LIPIcs.ICALP.2022.59

[15] Sariel Har-Peled, Piotr Indyk, Sepideh Mahabadi, and Ali Vakilian. 2016. Towards Tight Bounds for the Streaming Set Cover Problem. 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*, Tova Milo and Wang-Chiew Tan (Eds.). ACM, 371–383. https://doi.org/10.1145/2902251.2902287

[16] Piotr Indyk, Sepideh Mahabadi, Ronitt Rubinfeld, Jonathan R. Ullman, Ali Vakilian, and Anak Yodpinyanee. 2017. Fractional Set Cover in the Streaming Model. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA (LIPIcs, Vol. 81)*, Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 12:1–12:20. https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2017.12

[17] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. 2014. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, Chandra Chekuri (Ed.). SIAM, 734–751. https://doi.org/10.1137/1.9781611973402.55

[18] Michael Kapralov, Slobodan Mitrovic, Ashkan Norouzi-Fard, and Jakab Tardos. 2020. Space Efficient Approximation to Maximum Matching Size from Uniform Edge Samples. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, Shuchi Chawla (Ed.). SIAM, 1753–1772. https://doi.org/10.1137/1.9781611975994.107

[19] Sanjeev Khanna and Christian Konrad. 2022. Optimal Bounds for Dominating Set in Graph Streams. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA (LIPIcs, Vol. 215)*, Mark Braverman (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 93:1–93:23. https://doi.org/10.4230/LIPIcs.ITCS.2022.93

[20] Christian Konrad, Frédéric Magniez, and Claire Mathieu. 2012. Maximum Matching in Semi-streaming with Few Passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings (Lecture Notes in Computer Science, Vol. 7408)*, Anupam Gupta, Klaus Jansen, José D. P. Rolim, and Rocco A. Servedio (Eds.). Springer, 231–242. https://doi.org/10.1007/978-3-642-32512-0_20

[21] C. L. Lim, Alistair Moffat, and Anthony Ian Wirth. 2014. Lazy and Eager Approaches for the Set Cover Problem. In *Thirty-Seventh Australasian Computer Science Conference, ACSC 2014, Auckland, New Zealand, January 2014 (CRPIT, Vol. 147)*, Bruce Thomas and Dave Parry (Eds.). Australian Computer Society, 19–27. http://crpit.scem.westernsydney.edu.au/abstracts/CRPITV147Lim.html

[22] Barna Saha and Lise Getoor. 2009. On Maximum Coverage in the Streaming Model & Application to Multi-topic Blog-Watch. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*. SIAM, 697–708. https://doi.org/10.1137/1.9781611972795.60

[23] Stergios Stergiou and Kostas Tsioutsiouliklis. 2015. Set Cover at Web Scale. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Sydney, NSW, Australia) *(KDD '15)*. Association for Computing Machinery, New York, NY, USA, 1125–1133. https://doi.org/10.1145/2783258.2783315

A RANDOM ORDER ALGORITHM

A.1 Proof of Concentration Result: Lemma 2

Proof. The experiment described in the statement of the lemma can be seen as follows. We are given a ground set of size N and a subset of the ground set X of size $|X|$. We now pick a random subset of the ground set of size ℓ. We would like to show that the number of elements of X in this subset is concentrated with high probability.

We first prove the upper bound. To this end, consider the process of repeatedly (ℓ times) drawing elements from the ground set without replacement. Denote by Y_i the number of elements of X that were drawn within the first i steps. Then, the probability that the $(i+1)$th element drawn is from X is:

$$\frac{|X| - Y_i}{N - i} \leq \frac{|X|}{N - \ell} .$$

This process is stochastically dominated by a sequence of independent Bernoulli trials with success probability $p = \frac{|X|}{N-\ell}$. This process has the expected value $\mu = p \cdot \ell = \frac{\ell}{N-\ell}|X|$. Hence, by a Chernoff bound, $Y_\ell \leq C \log m \cdot \max\{\frac{\ell}{N-\ell}|X|, 1\}$, for a large constant C, which implies the second result. To prove the upper bound of the first result, we use the assumptions $\ell \leq 0.001N$ and $\frac{\ell}{N}|X| \geq C \log m$, for some large constant C. Then, the expected value μ is bounded

as

$$\mu = \frac{\ell}{N-\ell}|X| \geq \frac{\ell}{N}|X| \geq C \log m\,,$$

for a large constant C. Hence, by a Chernoff bound, we have that with probability at least $1 - \frac{1}{m^{21}}$,

$$Y_\ell \leq 1.001\frac{\ell}{N-\ell}|X| \leq 1.001\frac{\ell}{N-0.001N}|X| = \frac{1.001}{1-0.001}\cdot\frac{\ell}{N}|X| \leq 1.01\cdot\frac{\ell}{N}|X|\,.$$

Next, we prove the upper bound of result 3. We use the assumptions $\ell \leq \frac{N}{\sqrt{n}}$ and $\frac{\ell}{N}|X| \geq \log^6 m$. In this case, the expected value μ is at least $\log^6 m$. By a Chernoff bound, the probability that the outcome is larger by an additive $\delta\mu$ term is at most $e^{-\frac{\delta^2}{2+\delta}\mu}$. Hence, we can choose $\delta = \frac{\log m}{\sqrt{\mu}}$ to obtain an error probability of at most $\frac{1}{m^{20}}$ (for large enough m). We thus conclude that with probability $1 - \frac{1}{m^{20}}$, we have

$$Y_\ell \leq \frac{\ell}{N-\ell}|X| + \log(m)\sqrt{\frac{\ell}{N-\ell}|X|} \leq \frac{\ell}{N(1-\frac{1}{\sqrt{n}})}|X| + \log(m)\sqrt{\frac{\ell}{N(1-\frac{1}{\sqrt{n}})}|X|}\,.$$

Next, we prove the lower bound results, and we first focus on result 1. We condition on the event that the upper bound holds, i.e., $Y_\ell \leq 1.01\cdot\frac{\ell}{N}|X|$. We then bound the inclusion probability of item $i+1$ as follows:

$$\frac{|X|-Y_i}{N-i} \geq \frac{|X| - 1.01\cdot\frac{\ell\cdot|X|}{N}}{N} = \frac{|X|}{N}\left(1 - 1.01\cdot\frac{\ell}{N}\right) \geq \frac{|X|}{N}(1 - 0.0001\cdot 1.01)\,.$$

Our process thus stochastically dominates a sequence of independent Bernoulli trials with success probability $p = \frac{|X|}{N}(1 - 0.0001\cdot 1.01)$. Similar to the calculation above, by Chernoff bounds, we obtain that $Y_\ell \geq 0.99\cdot\frac{\ell|X|}{N}$ holds with probability at least $1 - \frac{1}{m^{21}}$. By a union bound. both upper and lower bounds hold with probability $1 - \frac{1}{m^{20}}$.

We now turn to the lower bound stated as result 3. By the same calculation as for result 1 (observe that we even use the upper bound from result 1 here), we see that the resulting process stochastically dominates a sequence of ℓ independent Bernoulli trials with success probability $p = \frac{|X|}{N}(1 - \frac{1.01}{\sqrt{n}})$ (using $\ell \leq \frac{N}{\sqrt{n}}$). The expected number of successes of this process is therefore $\mu = \frac{\ell}{N}|X|(1 - \frac{1.01}{\sqrt{n}})$, which, by the assumption of result 3, is at least $\log^6 m(1 - \frac{1.01}{\sqrt{n}})$. Denote by Z the number of observed successes. Then, by a Chernoff bound,

$$\Pr[Z \leq \mu - \delta\mu] \leq e^{\mu\delta^2/2}\,.$$

Hence, setting $\delta = \frac{\log m}{\sqrt{\mu}}$, we obtain a failure probability of $e^{\log^2 m/2}$, which is at most $\frac{1}{m^{20}}$ for large enough m. The result follows. □

A.2 Proof of Invariant (I2)

Lemma 4 (**Invariant (I2)**). *If a set S is included in $\mathrm{Sol}^{(i)}$ during the execution of $\mathsf{A}^{(i)}$, then with probability at least $1 - 1/m^3$, there are only $O(\sqrt{n}\log^9 m)$ missed edges that are incident on the set S.*

Proof. Fix any set $S \in \mathrm{Sol}^{(i)}$. W.l.o.g. assume that the set S was added to $\mathrm{Sol}^{(i)}$ during the execution of $\mathsf{A}^{(i)}$. We also assume that $|S| \geq C'\cdot\sqrt{n}\log^3 m$, for some large C', since otherwise the statement is trivially true.

Let Γ be the set of slots in the stream in which edges incident on S appear during the execution of epoch 0 and algorithms $\mathsf{A}^{(1)}, ..., \mathsf{A}^{(i)}$. To prove invariant (**I2**), it suffices to show that with probability at least $1 - 1/m^3$, the number of elements in $(S \cap U^{(i)})$ that appear in Γ is bounded by $O(\sqrt{n}\log^3 m)$, as this is precisely the set of missed edges incident on S. To bound this quantity, we will partition Γ into two sets, Γ_1 and Γ_2, where $\Gamma_1 \subseteq \Gamma$ denotes the set of slots in epoch 0 and in the executions of $\mathsf{A}^{(1)}, ..., \mathsf{A}^{(i)}$ which reside in the sub-epochs of these algorithms devoted to the processing of the set S. We will refer to these slots as *observed slots* of S as they are the slots where the algorithm observes which elements incident on S appear. We will refer to the set of remaining slots, namely $\Gamma_2 = \Gamma \setminus \Gamma_1$, as the *unobserved slots* of S. Note that once we fix the set of elements in S that appear in the observed slots, the execution of the algorithms $\mathsf{A}^{(1)}, ..., \mathsf{A}^{(i)}$, is *oblivious* to the remaining elements of S that appear in the unobserved slots. Moreover, each remaining element of set S is equally likely to appear in any of the unobserved slots.

Now to prove invariant (**I2**), we will bound the number of elements in $(S \cap U^{(i-1)})$ that appear in Γ as this is an upper bound on the set of missed edges incident on set S. By the same argument as in the proof of **(I1)**, we have $|\Gamma_1| \leq 1.01\sqrt{n}$ with probability $1 - \frac{1}{m^{20}}$, and we will prove that the number of elements in $(S \cap U^{(i-1)})$ that appear in Γ_2 is $O(\sqrt{n}\log^3 m)$. These two properties together complete the proof.

We now bound $|\Gamma_2|$. We have

$$E[|\Gamma_2|] \leq \left(\frac{2}{\sqrt{n}} + \frac{4}{\sqrt{n}} + ... + \frac{2^i}{\sqrt{n}}\right)\cdot|S| \leq \left(\frac{2^{i+1}}{\sqrt{n}}\right)\cdot|S|\,,$$

where each $\frac{2^r}{\sqrt{n}}$ corresponds to the fraction of the stream observed by algorithm $\mathsf{A}^{(r)}$. Note that we also have the lower bound $E[|\Gamma_2|] \geq \frac{2^i}{\sqrt{n}}\cdot|S| \geq C'2^i\log^3 m$. Hence, by Lemma 2, with probability at least $1 - 1/m^{20}$, the total number of edges incident on S that appear during the execution of $\mathsf{A}^{(1)}, ..., \mathsf{A}^{(i)}$ is at most $1.01\cdot\frac{2^{i+1}}{\sqrt{n}}\cdot|S|$.

We now note that every unobserved element in the set S is equally likely to appear in any unobserved slot. Thus, the expected number of elements in $(S \cap U^{(i-1)})$ that appear in the unobserved slots Γ_2 can be bounded from above by

$$1.01\frac{2^{i+1}}{\sqrt{n}}\cdot|S|\cdot\frac{|S\cap U^{(i)}|}{|S|} = 1.01\frac{2^{i+1}}{\sqrt{n}}\cdot|S\cap U^{(i)}|\,.$$

By Invariant (**I1**), with probability at least $1 - 1/m^4$, we know that $|S \cap U^{(i-1)}| \leq (n/2^{i-1})\log^9 m$. Thus with probability at least $1 - 1/m^4$, the expected number of elements in $(S \cap U^{(i-1)})$ that

appear in Γ_2 is at most

$$1.01\frac{2^{i+1}}{\sqrt{n}} \cdot |S \cap U^{(i)}| \leq 1.01\frac{2^{i+1}}{\sqrt{n}} \cdot (n/2^{i-1}) \log^9 m \leq 5\sqrt{n}\log^9 m\,.$$

Finally, using Chernoff bounds for negatively correlated random variables, we conclude that with probability at least $1 - 1/m^3$, the number of elements in $(S \cap U^{(i)})$ that appear in Γ_2 is at most $10\sqrt{n}\log^9 m$.

It now follows that with probability at least $1 - 1/m^3$, the total number of elements in $(S \cap U^{(i)})$ that appear during the execution of $\mathsf{A}^{(1)}, ..., \mathsf{A}^{(i)}$ in slots $\Gamma = \Gamma_1 \cup \Gamma_2$, is $O(\sqrt{n}\log^9 m)$, completing the proof of the invariant (**I2**). □

A.3 Proof of Invariant (I3)

Before presenting the proof of Invariant **(I3)**, we require some technical lemmas, which rely on the notion of *forward-edges*:

Definition 1. *We say that, at a given position in the stream, an element $u \in \mathcal{U}$ is* forward-incident *to a set S if the edge (S, u) appears in the remainder of the stream. We then also say that (S, u) is a* forward-edge.

Our first lemma states that if a set is special in epoch j of algorithm $\mathsf{A}^{(i)}$, then the set was also special in epoch $j-1$ of $\mathsf{A}^{(i)}$. This monotonicity property is a consequence of the gradually increasing thresholds $j \cdot \log^6 m$. If a set is special in epoch j, i.e., it has seen $j \cdot \log^6 m$ incident edges towards uncovered elements, then it is extremely unlikely that it has seen less than $(j-1)\log^6 m$ incident edges towards uncovered elements in epoch $j-1$.

Lemma 5. *For every set S, every algorithm $\mathsf{A}^{(i)}$, and every epoch $j \geq 2$, with probability $1 - \frac{1}{m^{10}}$, if set S is chosen for sampling in epoch j of $\mathsf{A}^{(i)}$, then the set S was also chosen for sampling in epoch $(j-1)$ of $\mathsf{A}^{(i)}$.*

Proof. Fix any set S, any algorithm $\mathsf{A}^{(i)}$, and any epoch $j \geq 2$. Now consider the moment at the beginning of epoch $(j-1)$ of algorithm $\mathsf{A}^{(i)}$ and let $S' \subseteq S$ denote the yet unmarked *forward-incident elements* of S, i.e., the elements $u \in S$ that are not yet marked such that (S, u) is a forward-edge.

We will prove that if $|S'| \leq \frac{(j-\frac{1}{2})\cdot n \cdot \log^7 m}{2^i}$ then S would not be sampled in epoch j with high probability, and on the other hand, if $|S'| \geq \frac{(j-\frac{1}{2})\cdot n \cdot \log^7 m}{2^i}$ then S would be sampled in epoch $(j-1)$ with high probability. Hence, no matter what the exact size of $|S'|$ is, the event that S would be sampled in epoch j but not sampled in epoch $j-1$ can only happen with vanishingly small probability.

Suppose thus that $|S'| \leq \frac{(j-\frac{1}{2})\cdot n \cdot \log^7 m}{2^i}$. As argued in Inequality 1, when algorithm $\mathsf{A}^{(K)}$ finishes, then at most a $\frac{1}{\log^3 m}$ fraction of the stream has been processed. Let N' denote the length of the stream that has not yet been processed. Then, $N' \geq N \cdot (1 - \frac{1}{\log^3 m})$. Observe further that $\ell_i \leq \frac{N'}{\sqrt{n}}$ holds for all $1 \leq i \leq K$. Then, by Lemma 2, at most

$$\begin{aligned}
&\frac{\ell_i}{N'} \cdot \frac{(j-\frac{1}{2})\cdot n \cdot \log^7 m}{2^i \cdot (1-\frac{1}{\sqrt{n}})} + \log(m) \cdot \sqrt{\frac{\ell_i}{N'} \cdot \frac{(j-\frac{1}{2})\cdot n \cdot \log^7 m}{2^i \cdot (1-\frac{1}{\sqrt{n}})}} \\
&= \frac{(j-\frac{1}{2})\log^6 m}{(1-\frac{1}{\log^3 m})\cdot(1-\frac{1}{\sqrt{n}})} + \log(m) \cdot \sqrt{\frac{(j-\frac{1}{2})\log^6 m}{(1-\frac{1}{\log^3 m})\cdot(1-\frac{1}{\sqrt{n}})}} \\
&\leq (j-\frac{1}{3})\log^6 m + \log m\sqrt{(j-\frac{1}{3})\log^6 m} \\
&\leq (j-\frac{1}{3})\log^6 m + \log m\sqrt{\log^7 m} \leq (j-\frac{1}{3})\log^6 m + \log^{4.5} m
\end{aligned}$$

elements of S' would appear in the subepoch j dedicated to processing S with probability $1 - \frac{1}{m^{20}}$, and the set would therefore not be sampled (since the threshold in the algorithm is $j \cdot \log^6 m$).

Suppose now that $|S'| \geq \frac{(j-\frac{1}{2})\cdot n \cdot \log^7 m}{2^i}$. Similar to the reasoning above, by Lemma 2, at least

$$\begin{aligned}
&\frac{\ell_i}{N'} \cdot \frac{(j-\frac{1}{2})\cdot n \cdot \log^7(m)(1-\frac{1}{\sqrt{n}})}{2^i} \\
&\quad - \log(m) \cdot \sqrt{\frac{\ell_i}{N'} \cdot \frac{(j-\frac{1}{2})\cdot n \cdot \log^7(m)(1-\frac{1}{\sqrt{n}})}{2^i}} \\
&\quad \geq (j-\frac{2}{3})\log^6 m - \log m\sqrt{(j-\frac{2}{3})\log^6 m} \\
&\quad \geq (j-\frac{2}{3})\log^6 m - \log^{4.5} m \geq (j-1)\log^6 m
\end{aligned}$$

elements of S' would appear in subepoch $j-1$ with probability $1 - \frac{1}{m^{20}}$ and the set would therefore be sampled.

Taking a union bound over the error probabilities incurred for all sets, algorithms $\mathsf{A}^{(i)}$, and epochs, the result follows. □

Next, we take the perspective of an unmarked element. We show that every unmarked element is forward-incident to at most $1.1\frac{m}{2^j\sqrt{n}}$ elements at the end of epoch j with high probability. This implies that, the higher the epoch, elements can contribute less and less to increasing the counters of sets. Consequently, the number of special sets decreases from epoch to epoch. This is proved in Lemmas 6 and 8. In Lemma 7, we argue that uncovered elements are not marked with high probability. This property is used in the proof of **(I1)**. Since the reasoning follows from the proof of Lemmas 6, we give the lemma here.

Lemma 6. *For any algorithm $\mathsf{A}^{(i)}$ and any epoch $j \geq 0$, with probability at least $1 - \frac{j+1}{m^8}$, at the end of epoch j of algorithm $\mathsf{A}^{(i)}$, every not yet marked as covered element is forward-incident to at most $1.1\frac{m}{2^j\sqrt{n}}$ special sets of epoch j.*

Proof. We consider the initial sampling of sets with probability p_0 as epoch 0 of any of the algorithms $\mathsf{A}^{(i)}$ (i.e., all algorithms $\mathsf{A}^{(1)}, \mathsf{A}^{(2)}, \ldots$ have the same epoch 0). Denote by $\mathsf{Sol}^{(0)}$ the sampled sets. After the initial sampling, the number of occurrences of all elements in a substream of length $C\frac{\sqrt{n}N\log m}{m}$ are computed. Consider an element of degree at least $1.1\frac{m}{\sqrt{n}}$. Then, by Lemma 2, the element appears at least $0.99 \cdot 1.1\frac{m}{\sqrt{n}} \cdot C\frac{\sqrt{n}N\log m}{m\cdot N} = 0.99 \cdot 1.1C\log m =$

$1.089C \log m$ times in this sample with probability $1 - \frac{1}{m^{20}}$. With a similar reasoning, an element of degree at most $1.07\frac{m}{\sqrt{n}}$ appears at most $1.01 \cdot 1.07C \log m = 1.0807C \log m$ times in this sample with probability $1 - \frac{1}{m^{20}}$. We will therefore mark as covered every element that appears at least $1.085C \log m$ times in this sample. This guarantees that all elements with degree at least $1.1\frac{m}{\sqrt{n}}$ are marked while all elements with degree at most $0.7\frac{m}{\sqrt{n}}$ remain unmarked. This proves the statement for $j = 0$.

Next, consider the state of $\mathsf{A}^{(i)}$ after epoch j and suppose that the lemma holds for this epoch. Let Q_j denote the set of special elements from epoch j. Let $u \in \mathcal{U}$ be any element that is not yet marked as covered, and let $\mathrm{fd}(u, Q_j)$ denote the forward-degree of u to Q_j, i.e., the number of forward-edges between u and Q_j. Then, by the induction hypothesis, $\mathrm{fd}(u, Q_j) \leq 1.1\frac{m}{2^j\sqrt{n}}$ with probability $1 - \frac{j+1}{m^8}$.

Our algorithm in epoch j subsamples the sets in Q_j each with probability $q_j = \min\{\frac{2^j}{n}, 1\}$ and stores the sampled sets in variable $\tilde{Q}'$ (which becomes variable $\tilde{Q}$ in epoch $j+1$). We denote the subsampled set by $\tilde{Q}_j$. The algorithm then tracks all edges of epoch $j+1$ between $\tilde{Q}_j$ and the yet unmarked vertices.

We will now bound the number of edges tracked in epoch $j+1$. By Lemma 2, the number of forward-edges between an element u and Q_j in epoch j is $O(\frac{\ell_i}{N}\mathrm{fd}(u, Q_j) \log m)$ with probability at least $1 - \frac{1}{m^{20}}$, and, by a Chernoff bound, with probability $1 - \frac{1}{m^{C_2}}$, for any large constant C_2, there are

$$O(\frac{\ell_i}{N}\mathrm{fd}(u, Q_j) \log m \cdot q_j \log m) = \tilde{O}(\frac{2^i}{\sqrt{n}\log n} \cdot \mathrm{fd}(u, Q_j) \cdot \frac{2^j}{n}) = \tilde{O}(\mathrm{fd}(u, Q_j) \cdot \frac{2^j}{n}),$$

forward-edges between u and $\tilde{Q}_j$ in epoch $j+1$ (using the inequality $2^i \leq \sqrt{n}$). By the induction hypothesis, we know that $\mathrm{fd}(u, Q_j) \leq 1.1\frac{m}{2^j\sqrt{n}}$ for any element. Hence, the total number of tracked edges is:

$$\sum_{u \in \mathcal{U}} \tilde{O}(\mathrm{fd}(u, Q_j) \cdot \frac{2^j}{n}) = n \cdot \tilde{O}(\frac{m}{2^j\sqrt{n}} \cdot \frac{2^j}{n}) = \tilde{O}(\frac{m}{\sqrt{n}}).$$

We can thus track all these elements with space $\tilde{O}(\frac{m}{\sqrt{n}})$.

Next, we claim that the sampling allows us to distinguish between uncovered elements $u \in \mathcal{U}$ with forward-degree $\mathrm{fd}(u, Q_j) \geq 1.1\frac{m}{2^{j+1}\sqrt{n}}$ and those elements v with forward-degree $\mathrm{fd}(v, Q_j) \leq 1.07\frac{m}{2^{j+1}\sqrt{n}}$. Indeed, consider an element u with forward-degree at least $1.1\frac{m}{2^{j+1}\sqrt{n}}$. Then, by a Chernoff bound, with probability at least $1 - \frac{1}{m^{20}}$, the forward-degree between u and $\tilde{Q}_j$ is $\mathrm{fd}(u, \tilde{Q}_j) \geq 1.099\frac{m}{2^{j+1}\sqrt{n}} \cdot \frac{2^j}{n} = 1.099\frac{m}{2n^{1.5}}$. Then, by Lemma 2, with probability at least $1 - \frac{1}{m^{20}}$, there will be at least

$$0.99 \cdot 1.099\frac{m}{2n^{1.5}} \cdot \frac{2^i}{\sqrt{n}\log m} = 0.99 \cdot 1.099\frac{m2^{i-1}}{n^2 \log m}$$

forward-edges in epoch j (we assumed here that m is large enough, e.g., $m = \Omega(n^2 \log^3 m)$, to get concentration, which allows us to apply Lemma 2). By a similar reasoning, for an element v with $\mathrm{fd}(v, Q_j) \leq 1.07\frac{m}{2^{j+1}\sqrt{n}}$, there will be at most

$$1.01 \cdot 1.071\frac{m2^{i-1}}{n^2 \log m}$$

forward-edges in epoch j between v and $\tilde{Q}_j$. We can thus distinguish the two cases and mark as covered all elements that appear at least $1.085 \cdot \frac{m2^{i-1}}{n^2 \log m}$ times (observe that $1.01 \cdot 1.071 < 1.085 < 0.99 \cdot 1.099$).

The argument is then completed by the fact that the special elements in epoch $j+1$ are a subset of those in epoch j. Last, bounding the error probabilities in the induction step with a union bound, including the error of $\frac{j+1}{m^8}$ from the induction hypothesis, we see that the statement holds with probability $1 - \frac{j+2}{m^8}$. □

Lemma 7. *With probability $1 - \frac{1}{m^{18}}$, an element $u \in U^{(i)}$ is not marked when algorithm $\mathsf{A}^{(i)}$ terminates.*

Proof. The only possibility for an element $u \in \mathcal{U}^{(i)}$ to be marked is if u is marked in Line 31 but none of the special sets that contain u are added to Sol. The analysis of the previous lemma reveals that if u is marked then u was incident to at least $1.01 \cdot 1.07\frac{m}{2^{j+1}\sqrt{n}}$ special sets, and each special set is added to Sol with probability $p_j = C \cdot \frac{2^j\sqrt{n}}{m} \log m$. By Chernoff bounds, the probability that none of these sets are added to Sol is at most $\frac{1}{m^{30}}$. The result follows. □

Lemma 8. *For any i, j, there are at most $1.1 \cdot \frac{m}{2^j}$ special sets in epoch j of algorithm $\mathsf{A}^{(i)}$ with probability $1 - \frac{1}{m^5}$.*

Proof. Consider the state of the algorithm at the end of epoch $j-1$. Then, by Lemma 6, every unmarked element u has a forward-degree to the set of special elements Q_j in epoch j of at most $1.1\frac{m}{2^j\sqrt{n}}$ with probability $1 - \frac{1}{m^6}$. By Lemma 2, for each element, with probability $1 - \frac{1}{m^{20}}$, at most

$$1.01 \cdot 1.1\frac{m}{2^j\sqrt{n}} \cdot \frac{2^i}{\sqrt{n}\log m} \leq 1.01 \cdot 1.1\frac{m}{2^j\sqrt{n}\log m} \leq \frac{m}{2 \cdot 2^j\sqrt{n}}$$

of these forward-edges appear in epoch j. Observe that, any of these edges (u, S) appears in the subepoch of epoch j that processes S with probability $\frac{1}{\sqrt{n}}$. Hence, by concentration bounds, only $2 \cdot \frac{m}{2 \cdot 2^j\sqrt{n}} \cdot \frac{1}{\sqrt{n}} = \frac{m}{2^j n}$ of these edges contribute to increasing the counters. Since there are n elements, the sum of the counters can reach at most $\frac{m}{2^j}$, and since a set becomes special once a counter reaches $j \cdot \log^6 m \geq 1$, the result follows. □

Lemma 9 (**Invariant (I3)**). *With probability at least $1 - 1/m^3$, the total number of sets added to* Sol *during the execution of $\mathsf{A}^{(i)}$ is $O(\sqrt{n}\log^2 m)$.*

Proof. By Lemma 8, for any i, j, there are $O(\frac{m}{2^j})$ special sets in epoch j of algorithm $\mathsf{A}^{(i)}$ with probability $1 - \frac{1}{m^5}$, and by a union bound, this statement holds for all i, j with probability $1 - \frac{1}{m^3}$. Since each of these sets is added to the solution with probability $p_i = C \cdot \frac{2^i\sqrt{n}\log m}{m}$, by a Chernoff bound, at most $O(\sqrt{n}\log m)$ sets are added. Hence, overall $O(\sqrt{n}\log^2 m)$ sets are added in algorithm $\mathsf{A}^{(i)}$. □