Maximum Coverage in Random-Arrival Streams

Rowan Warneke
School of Computing and Information Systems, The University of Melbourne, Australia

Farhana Choudhury
School of Computing and Information Systems, The University of Melbourne, Australia

Anthony Wirth
School of Computing and Information Systems, The University of Melbourne, Australia

Abstract

Given a collection of m sets, each a subset of a universe $\{1,\ldots,n\}$, *maximum coverage* is the problem of choosing k sets whose union has the largest cardinality. A simple greedy algorithm achieves an approximation factor of $1 - 1/e \approx 0.632$, which is the best possible polynomial-time approximation unless $\mathrm{P} = \mathrm{NP}$.

In the streaming setting, information about the input is revealed gradually, in an online fashion. In the set-streaming model, each set is listed contiguously in the stream. In the more general edge-streaming model, the stream is composed of set-element pairs, denoting membership. The overall goal in the streaming setting is to design algorithms that use sublinear space in the size of the input. An interesting line of research is to design algorithms with space complexity polylogarithmic in the size of the input (i.e., polylogarithmic in both n and m); we call such algorithms low-space. In the set-streaming model, it is known that $1/2$ is the best possible low-space approximation. In the edge-streaming model, no low-space algorithm can achieve a nontrivial approximation factor.

We study the problem under the assumption that the order in which the stream arrives is chosen uniformly at random. Our main results are as follows.

- In the random-arrival set-streaming model, we give two new algorithms to show that low space is sufficient to break the $1/2$ barrier. The first achieves an approximation factor of $1/2 + c_1$ using $\tilde{O}(k^2)$ space, where $c_1 > 0$ is a small constant and $\tilde{O}(\cdot)$ notation suppresses polylogarithmic factors; the second achieves a factor of $1 - 1/e - \varepsilon - o(1)$ using $\tilde{O}(k^2\varepsilon^{-3})$ space, where the $o(1)$ term is a function of k. This is essentially the optimal bound, as breaking the $1 - 1/e$ barrier is known to require high space.
- In the random-arrival edge-streaming model, we show for all fixed $\alpha > 0$ and $\delta > 0$, any algorithm that α-approximates maximum coverage with probability at least 0.9 in the random-arrival edge-streaming model requires $\Omega(m^{1-\delta})$ space (i.e., high space), even for the special case of $k = 1$.

2012 ACM Subject Classification Theory of computation → Random order and robust communication complexity; Theory of computation → Sketching and sampling

Keywords and phrases Maximum Coverage, Streaming Algorithm, Random Arrival, Greedy Algorithm, Communication Complexity

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.102

Acknowledgements Thanks to Andrew McGregor for suggesting this as a possible research direction.

1 Introduction

Maximum coverage is a classic NP-hard problem in computer science with a range of applications, including facility allocation [16], information retrieval [2], and content recommendation [18]. We are given a collection of m sets, $\mathcal{F}$, each a subset of a universe $[n] \coloneqq \{1,\ldots,n\}$, and a positive integer, k. The goal is to choose k sets from $\mathcal{F}$ such that the cardinality of the union of the chosen sets – known as their *coverage* – is maximised. A standard greedy algorithm achieves a $(1 - 1/e)$-approximation and unless $\mathrm{P} = \mathrm{NP}$, this is the best approximation possible in polynomial time [7].

The greedy algorithm performs well in practice, but requires access to the entire input during its operation, which scales poorly for massive data sets. In the last 15 years, there has been an increasing focus on *streaming algorithms* for maximum coverage [18, 15, 6, 11]. These algorithms process the input piece by piece and use substantially less space than would be required to simply read the entire input into memory and run a classical algorithm, which requires $O(mn)$ space in general. In the *set-streaming* model, each set arrives contiguously in the stream: all the elements in one set are listed, then all the elements in the next set, and so on. In the more general *edge-streaming* model, each entity in the stream is a set-element pair, so the full description of a set might be spread across the stream.

Most existing work on streaming algorithms for maximum coverage has focused on the *arbitrary-arrival* model, in which the stream may arrive in any order. In particular, the stream may arrive in a *worst-case* order, so any theoretical guarantees made about an algorithm in this model must be made for all possible stream orderings. In practice, however, it is often reasonable to assume that the data stream is randomly ordered;[1] in the *random-arrival* model, all possible stream orderings are considered equally likely, and theoretical guarantees about an algorithm in this model need only be made as an *average* over these orderings. The random-arrival model has been studied for many important problems, including quantile estimation [10], maximum matching [13, 4], and submodular maximisation [17, 1], often revealing improved space-accuracy trade-offs.

In this paper, we study maximum coverage in the random-arrival model. In the *random-arrival set-streaming* model,[2] the order in which the sets appear in the stream is uniformly random, but we assume nothing about the order in which the elements in each set arrive. By contrast, in the *random-arrival edge-streaming* model, the edges arrive in a uniformly random order, so large sets are more likely to be represented early in the stream. For each model, our goal is to determine whether the random-arrival assumption permits a better approximation factor than is possible in the corresponding arbitrary-arrival model, when very low space is available (i.e., polylogarithmic in both n and m). We answer this question in the affirmative for random-arrival set-streaming: we present two low-space algorithms, each achieving an approximation factor better than $1/2$ (the best possible approximation in the arbitrary-arrival model). For random-arrival edge-streaming, we demonstrate a near-tight space lower bound, hence answering in the negative: nearly linear in m space is required to achieve a nontrivial approximation.

1.1 Related work

Table 1 summarises relevant past work on streaming algorithms for maximum coverage. Note that any algorithm that guarantees an approximation factor in the arbitrary-arrival model trivially also guarantees this approximation factor in the random-arrival model.

The multi-pass variant of streaming maximum coverage has also been studied, in which a small number of passes over the stream are allowed [15, 12]. In this paper, however, we focus exclusively on single-pass algorithms.

Unless otherwise specified, the following results apply to the arbitrary-arrival model.

[1] See Guha and McGregor [10] §1.1 for a discussion on justifications for this assumption.

[2] Note that we consider both "random-arrival" and "set-streaming" to be part of the model specification. When we talk about *the* random-arrival model, we mean both the random-arrival set-streaming and random-arrival edge-streaming models.

Set-streaming. Saha and Getoor [18], who introduced the set-streaming model, gave a 1/4-approximation algorithm called SOPS,[3] which uses $\tilde{O}(nk)$ space and explicitly returns the k chosen sets. Yu and Yuan [20] studied the problem under the more relaxed ID-reporting output specification, in which only the *IDs* of the k chosen sets must be returned by the algorithm.[4] They gave an algorithm, GOPS, which achieves an approximation factor of approximately 0.3 and uses $\tilde{O}(n)$ space, where $\tilde{O}(\cdot)$ notation suppresses dependence on polylogarithmic factors. Further progress came indirectly from Badanidiyuru et al. [5], who gave a $(1/2 - \varepsilon)$-approximation algorithm for the related submodular maximisation problem. A careful adaptation to maximum coverage uses $\tilde{O}(n\varepsilon^{-1})$ space [15].

More recently, McGregor and Vu [15] developed the first set-streaming maximum coverage algorithms that use sublinear-in-n space. Using *subsampling*, which effectively discards much of the universe, $[n]$, they substantially reduce the space consumption of their algorithms with minimal loss in solution quality. Of the four new algorithms they developed, the fourth, which we call MV-4 after the authors, is the most relevant to our work, as it is both low-space and single-pass. MV-4 achieves an approximation factor of $1/2 - \varepsilon$ using just $\tilde{O}(k\varepsilon^{-3})$ space. The algorithm is explained in detail in Section 2.1. The other two MV algorithms that are also single-pass are also included in Table 1 (although note that they are not low-space).

McGregor and Vu [15] also proved the first nontrivial space lower bound for set-streaming maximum coverage. They showed that achieving an approximation factor better than $1 - 1/e$ requires $\Omega(mk^{-2})$ space in the arbitrary-arrival model and $\Omega(mk^{-3})$ space in the random-arrival model. Feldman et al. [8] extended this result to approximation factors better than 1/2 for the arbitrary-arrival model.[5] Note that all lower bounds discussed in this work apply to the problem of simply *estimating* the optimal coverage.

The results of McGregor and Vu [15] and Feldman et al. [8] imply that the best possible approximation factor that can be achieved using low space in the arbitrary-arrival model is 1/2, where by "low space" we mean space polylogarithmic in both n and m.[6] In the random-arrival model, however, there is a knowledge gap for approximation factors between 1/2 and $1 - 1/e$: can some low-space algorithm achieve an approximation factor in this range, or is high space always required?

Edge-streaming. Bateni et al. [6] gave the first edge-streaming maximum coverage algorithm, which achieves an approximation factor of $1 - 1/e - \varepsilon$ using $\tilde{O}_\varepsilon(m)$ space, where the subscript suppresses dependence on ε (which was not analysed by the authors). Indyk and Vakilian [11] improved this result by showing that the optimal space bound is $\tilde{\Theta}(\alpha^2 m)$, where $\alpha > 0$ is the approximation factor. This result implies that achieving a nontrivial (i.e., nonzero) approximation factor using low space is impossible in the arbitrary-arrival edge-streaming model.

[3] The algorithm was given this name in a later work [20].
[4] IDs appear explicitly in the edge-streaming model as part of the set-element pairs. In the set-streaming model, we assume for convenience that the ID of each set is listed directly before the elements in the set, but this is a largely unimportant implementation detail.
[5] The authors actually studied the more general submodular maximisation problem. However, the submodular function constructed for the hardness instance is also a coverage function, so the result applies to maximum coverage as well.
[6] Other reasonable definitions exist, but we choose to focus on n and m since these are the parameters that can cause a maximum coverage problem instance to require a huge amount of space to store, which is the motivation for using streaming algorithms in the first place.

Table 1 A summary of known results and our new results for the space complexity of single-pass streaming approximation algorithms for maximum coverage. Each of the parameters $\alpha, \delta, \varepsilon$ is positive, and c_1 is a small positive constant.

Stream	Name	Arrival	Approximation	Space	Ref.
Set	SOPS	Arbitrary	$1/4$	$O(nk)$	[18]
	GOPS	Arbitrary	~ 0.3	$\tilde{O}(n)$	[20]
	MCSS	Arbitrary	$1/2 - \varepsilon$	$\tilde{O}(n\varepsilon^{-1})$	[5, 15]
	MV-1	Arbitrary	$1 - 1/e - \varepsilon$	$\tilde{O}(m\varepsilon^{-2})$	[15]
	MV-3	Arbitrary	$1 - \varepsilon$	$\tilde{O}(m\varepsilon^{-2} \cdot \min(k, \varepsilon^{-1}))$	[15]
	MV-4	Arbitrary	$1/2 - \varepsilon$	$\tilde{O}(k\varepsilon^{-3})$	[15]
	–	Arbitrary	$> 1 - 1/e$	$\Omega(mk^{-2})$	[15]
	–	Arbitrary	$> 1/2$	$\Omega(mk^{-3})$	[8]
	–	Random	$> 1 - 1/e$	$\Omega(mk^{-3})$	[15]
	GS-SALSA	Random	$1/2 + c_1$	$\tilde{O}(k^2)$	**Here**
	GS-SMC$^+$	Random	$1 - 1/e - \varepsilon - o(1)$	$\tilde{O}(k^2\varepsilon^{-3})$	**Here**
Edge	–	Arbitrary	$1 - 1/e - \varepsilon$	$\tilde{O}_\varepsilon(m)$	[6]
	–	Arbitrary	$\alpha > 0$	$\tilde{\Theta}(\alpha^2 m)$	[11]
	–	Random	Any fixed $\alpha > 0$	$\Omega(m^{1-\delta})$	**Here**

1.2 Our contributions

The results of this paper are included in Table 1.

Set-streaming. In the arbitrary-arrival model, the best approximation factor that can be achieved using low space is 1/2. We present two low-space algorithms that break the 1/2 barrier in the random-arrival model.

- The first algorithm, GS-SALSA, uses $\tilde{O}(k^2)$ space and achieves an approximation factor of $1/2 + c_1$ in-expectation, where $c_1 > 0$ is a small absolute constant.
- The second algorithm, GS-SMC$^+$, uses $\tilde{O}(k^2\varepsilon^{-3})$ space and achieves an approximation factor of $1 - 1/e - \varepsilon - o(1)$ in-expectation, where the $o(1)$ term is a function of k. For large k, this is essentially the optimal low-space approximation.[7]

Both algorithms are based on existing state-of-the-art random-arrival algorithms for the related *submodular maximisation* problem [17, 1]. Our main contribution is a generalisation of the subsampling technique introduced by McGregor and Vu [15] in the design of MV-4, which we apply to these existing submodular maximisation algorithms to achieve our results.

Edge-streaming. Indyk and Vakilian [11] showed that $\tilde{\Theta}(\alpha^2 m)$ space is necessary and sufficient for α-approximating maximum coverage in the arbitrary-arrival model. An immediate corollary is that for all *fixed* $\alpha > 0$, the problem requires $\Omega(m)$ space. We prove an almost matching hardness result for the random-arrival case.

[7] Throughout this work, we typically do *not* consider the approximation factor of an algorithm (or hardness result) as a function of k, and instead consider the *worst-case* value of k (just as we take a worst-case collection of sets $\mathcal{F}$). We make an exception for GS-SMC$^+$ due to the quality of its approximation for large values of k. Note that the hardness results of McGregor and Vu [15] and Feldman et al. [8] apply as k approaches infinity, so the algorithm really is near-optimal for large k.

▶ **Theorem 1.** *For all fixed $\alpha > 0$ and $\delta > 0$, any algorithm that α-approximates maximum coverage with probability at least 0.9 in the random-arrival edge-streaming model requires $\Omega(m^{1-\delta})$ space, even for the special case of $k = 1$.*

Our result implies that unlike set-streaming, edge-streaming maximum coverage is *not* made easier by the random-arrival assumption, in the sense that the assumption does not increase the best possible low-space approximation. Our proof of Theorem 1 is a careful combination of ideas from the proof for the arbitrary-arrival case by Indyk and Vakilian [11] with ideas from Andoni et al. [3], who proved a random-arrival space lower bound for the *frequency moment estimation* problem.

2 Beating 1/2 for Set-Streaming

In this section, we show that the 1/2 barrier can be broken for random-arrival set-streaming maximum coverage using low space. Our main contribution is a generalised version of the subsampling technique introduced by McGregor and Vu [15], which allows us to replace the core subroutine of MV-4 with *any* set-streaming maximum coverage algorithm. In particular, replacing the subroutine with state-of-the-art streaming algorithms for the related *submodular maximisation* problem yields two new maximum coverage algorithms with approximation factors better than 1/2 in the random-arrival model. The first, GS-SALSA, achieves an approximation factor of $1/2 + c_1$ for a small constant $c_1 > 0$. The second, GS-SMC$^+$, achieves an approximation factor of $1 - 1/e - \varepsilon - o(1)$, where the $o(1)$ term is a function of k. For large k, this is essentially the optimal bound.

2.1 Preliminaries: Design of MV-4

We start by providing a high-level overview of the MV-4 algorithm designed by McGregor and Vu [15], which achieves an approximation factor of $1/2 - \varepsilon$ using $\tilde{O}(k\varepsilon^{-3})$ space in the arbitrary-arrival model, which is essentially the optimal low-space approximation.

At the heart of MV-4 is a subroutine, $\mathcal{A}$, which achieves an approximation factor of $1/2-\varepsilon$ using high space. This seeming contradiction is avoided by only providing a small sample of the input to $\mathcal{A}$; MV-4 is equipped with a binary hash function $h : [n] \to \{0, 1\}$, and as each element e arrives in the stream, it is passed to the subroutine $\mathcal{A}$ only if $h(e) = 1$. The subroutine is therefore run on a *subsampled* problem instance $\mathcal{I}'$ over a smaller universe $[n]' = \{e \in [n] : h(e) = 1\}$, with subsampled sets $S'_i = S_i \cap [n]'$ for each $S_i \in \mathcal{F}$. An optimal solution to $\mathcal{I}'$ may not correspond to an optimal solution to $\mathcal{I}$, but when the hash function is chosen appropriately, a good solution to $\mathcal{I}'$ corresponds to a nearly-as-good solution to $\mathcal{I}$ with high probability. The hash function h is chosen such that $\mathbb{P}[h(e) = 1] = p$ for all $e \in [n]$. When the subsampling rate p is small, the space consumption of $\mathcal{A}$ is substantially reduced, and the overall algorithm becomes low-space.

The subsampling rate p is set in terms of the optimal coverage, OPT. Of course, we do not know OPT in advance! To get around this, MV-4 runs parallel instantiations of $\mathcal{A}$, each corresponding to a different guess v for OPT. Each instantiation $\mathcal{A}_v$ therefore has its own subsampling rate p_v, hash function h_v, and is run on a separate subsampled instance $\mathcal{I}'_v$. This introduces two new problems. Firstly, at the end of the stream, how do we know which instantiation corresponds to the correct guess? Rather than solve this problem directly, MV-4 maintains an F_0-*sketch*[8] of the set of elements covered (in the *unsubsampled* universe, $[n]$)

[8] Given a stream of items, an F_0-sketch is a small data structure capable of closely estimating the number

by the solution found by each instantiation. At the end of the stream, the solution with the highest estimated coverage is taken. Jaud et al. [12] gave a simpler method, which uses the *subsampled* coverage achieved by each instantiation to decide which solution to return.

The second problem introduced by the use of parallel instantiations concerns the space consumption of instantiations corresponding to bad guesses for OPT. If a guess v is much smaller than OPT, the subsampling rate p_v is set too high, the subsampled universe is too large, and so $\mathcal{A}_v$ uses too much space. To address this, MV-4 keeps track of the space consumption of each instantiation and terminates the instantiation if it uses too much space.

2.2 Generalised subsampling

In this section, we present a generalised version of McGregor and Vu's [15] subsampling technique. This allows the subroutine $\mathcal{A}$ to be replaced by *any* set-streaming maximum coverage algorithm $\mathcal{B}$. In Section 2.3, we apply our generalised subsampling technique to maximum coverage algorithms derived from state-of-the-art algorithms for submodular maximisation. The space consumption of these algorithms is proportional to $d = \max_{S_i \in \mathcal{F}} |S_i|$, the maximum set size of the problem instance. Under this condition, generalised subsampling substantially reduces the space consumption of $\mathcal{B}$. Our main result is as follows.

▶ **Theorem 2** (Generalised subsampling). *Suppose $\mathcal{B}$ achieves an approximation factor of α in-expectation*[9] *for set-streaming maximum coverage using $O(ds)$ space, where d is the maximum set size. There exists an algorithm, called GS($\mathcal{B}$), which, given $\varepsilon > 0$, achieves an approximation factor of $\alpha - \varepsilon$ in-expectation and uses $\tilde{O}(k\varepsilon^{-2}s)$ space.*

An important aspect of Theorem 2 is that no assumptions are made about the design of $\mathcal{B}$. This presents a challenge when we try to replace $\mathcal{A}$ with $\mathcal{B}$ in MV-4, as the operation of MV-4 relies on an important property of $\mathcal{A}$. In particular, $\mathcal{A}$ is *threshold-based*: as each set arrives in the stream, the algorithm decides irrevocably whether to include the set in the solution, and maintains a set C of the elements covered by the chosen sets. The decision to include an incoming set is based on whether the additional coverage provided by the set exceeds some threshold. This property is important to the design of MV-4 for two reasons. Firstly, it allows MV-4 to maintain a sketch of the elements covered by each instantiation of $\mathcal{A}$ in the unsubsampled universe, since after each set arrives, $\mathcal{A}$ can inform MV-4 as to whether it chose the set.[10] Secondly, the space consumption of $\mathcal{A}$ is directly proportional to the size of C, so this value may be used to trigger the termination of instantiations that use too much space.

We do not want to assume that $\mathcal{B}$ is threshold-based – indeed, we want to assume nothing at all about the design $\mathcal{B}$ – so we use an alternative idea. We abandon guessing OPT, and instead guess d, the maximum set size.[11] Our approach requires that some guess w satisfies $d/2 \leq w \leq d$, so we make guesses in powers of 2. The problem of choosing which

of distinct items in the stream. The details are unimportant for our purposes.

[9] There is nothing special about in-expectation guarantees, and the theorem could just as easily be proved for with-high-probability guarantees. We focus on in-expectation results since this is the type of guarantee made about the state-of-the-art submodular maximisation algorithms that we apply our results to in Section 2.3.

[10] The alternative approach used by Jaud et al. [12] works quite differently, but also exploits the threshold-based architecture of $\mathcal{A}$ by using the size of C for each instantiation to decide which solution to return.

[11] This is similar to an idea appearing in Badanidiyuru et al. [5], who studied streaming algorithms for submodular maximisation. Their algorithm, SIEVE-STREAMING, runs parallel instantiations corresponding to guesses for the maximum value of the given submodular function over any one item from the stream.

Algorithm 1 The generalised subsampling algorithm GS($\mathcal{B}$).

```
1: $W \leftarrow \{2^i : i \in \mathbb{N},\ 2^i \le n\}$                    ▷ Guesses for $d$
2: $\lambda \leftarrow \lfloor 2k \log(em\varepsilon^{-1}) \rfloor$        ▷ Hash function independence parameter
3: for $w \in W$ do
4:     Initialise $\mathcal{B}_w$, an instantiation of $\mathcal{B}$
5:     $p_w \leftarrow \min\{1, 3k\varepsilon^{-2}\log(em\varepsilon^{-1})w^{-1}\}$    ▷ Subsampling rate
6:     Sample $h_w \in \mathcal{H}_{p_w,\lambda}$ uniformly at random     ▷ Choose hash function
7:     active$_w$ ← true                                          ▷ Kill switch indicator
8: for $i = 1, \ldots, m$ do                          ▷ Iterate over each set in the stream
9:     for $e \in S_i$ do                             ▷ Iterate over each element in the set
10:        for $w \in W$ do
11:            if $d^*_w > 2p_w w(1+\varepsilon)$ then
12:                active$_w$ ← false                                 ▷ Terminate $\mathcal{B}_w$
13:            if active$_w$ and $h_w(e) = 1$ then
14:                Supply $e$ to the stream of $\mathcal{B}_w$
15: Let $I_w \subset [m]$ be the solution returned by $\mathcal{B}_w$
16: $w_c \leftarrow \min\{w \in W : d^*/2 \le w \le d^*\}$               ▷ At this point, $d^* = d$.
17: return $I_{w_c}$
```

solution to return becomes trivial, since we can easily keep track of the maximum observed set size and choose the appropriate instantiation at the end of the stream (Line 16). We also keep track of the maximum observed set size for each *subsampled* problem instance, and use this value to trigger the termination of bad instantiations (Line 11).

Our generalised subsampling algorithm GS($\mathcal{B}$) is formalised by Algorithm 1. The variable d^* keeps track of the running maximum set size observed so far in the stream. For example, $d^* = 0$ at the start of the stream, and $d^* = d$ at the end of the stream. Similarly, d^*_w keeps track of the maximum set size observed for the subsampled problem instance $\mathcal{I}'_w$ corresponding to the guess w for d. We omit from Algorithm 1 the straightforward steps required to keep track of these variables. The set $\mathcal{H}_{p_w,\lambda}$ appearing on Line 6 represents a λ-wise independent family of hash functions with the property that for all $e \in [n]$, $\mathbb{P}[h(e) = 1] = p_w$. A hash function h from this family may be sampled, stored, and evaluated using $O(\lambda)$ space [15]. In the remainder of this section, we show that GS($\mathcal{B}$) achieves the approximation factor and space consumption guaranteed by Theorem 2.

Approximation factor. GS($\mathcal{B}$) returns the solution found by $\mathcal{B}_{w_c}$, where w_c is the smallest guess for d satisfying $d/2 \le w_c \le d$ (see Line 16), so we focus on this instantiation of $\mathcal{B}$. In particular, S'_i is taken to mean the version of S_i subsampled using the hash function h_{w_c}, and the w_c subscript is often suppressed (e.g., $p = p_{w_c}$ and $\mathcal{I}' = \mathcal{I}'_{w_c}$).

Since each element $e \in [n]$ is subsampled with probability p, we expect that for any given choice of sets $S_1, \ldots, S_l$, the subsampled coverage of these sets will be approximately p times their unsubsampled coverage. The following result formalises this intuition.

▶ **Lemma 3.** *With probability at least $1 - \varepsilon$, for all collections of up to k sets $S_1, \ldots, S_l \in \mathcal{F}$, $|S'_1 \cup \cdots \cup S'_l| = p \cdot |S_1 \cup \cdots \cup S_l| \pm p\varepsilon d$.*

This result is similar to Lemma 8 from McGregor and Vu [15], with a few minor changes. Most notably, the probability of success is changed from a term involving m to one involving ε, which is important for the purpose of making an in-expectation guarantee.

Our proof more closely resembles the proof of Corollary 5 from Jaud et al. [12], who showed that the independence factor λ can be decreased substantially from its original definition in McGregor and Vu [15]. As in Jaud et al. [12], we require the following concentration bound for our proof.

▶ **Theorem 4** (Schmidt et al. [19]). *Let $X_1, \ldots, X_n$ be λ-wise independent binary random variables. Let $X = \sum_{i=1}^n X_i$ and $\mu = \mathbb{E}[X]$. If $\lambda \leq \lfloor \min(\gamma, \gamma^2)\mu e^{-1/3} \rfloor$, then*

$$\mathbb{P}\left[|X - \mu| \geq \gamma\mu\right] \leq e^{-\lfloor \lambda/2 \rfloor}.$$

Proof of Lemma 3. Fix any collection of sets $S_1, \ldots, S_l \in \mathcal{F}$ with $1 \leq l \leq k$. Let $D = |S_1 \cup \cdots \cup S_l|$ be the unsubsampled coverage of the collection and let $D' = |S_1' \cup \cdots \cup S_l'|$ be the subsampled coverage. Let $X_e = 1$ if $e \in S_1 \cup \cdots \cup S_l$ and $h(e) = 1$, and let $X_e = 0$ otherwise. Then $D' = \sum_{i=1}^n X_i$ and $\mu \coloneqq \mathbb{E}[D'] = pD$. The X_i are λ-wise independent by the choice of h, where $\lambda = \lfloor 2k \log(em\varepsilon^{-1}) \rfloor$. Define $\gamma = \varepsilon d/D$ so that $\gamma\mu = p\varepsilon d$. Before applying Theorem 4, we verify the necessary condition on the independence factor:

$$\begin{aligned}\left\lfloor \min(\gamma, \gamma^2)\mu e^{-1/3} \right\rfloor &= \left\lfloor \min(1, \gamma)\gamma\mu e^{-1/3} \right\rfloor \geq \left\lfloor \varepsilon \cdot p\varepsilon d \cdot \frac{2}{3} \right\rfloor \\ &= \left\lfloor \varepsilon^2 \cdot 3k\varepsilon^{-2} \log(em\varepsilon^{-1}) w^{-1} \cdot d \cdot \frac{2}{3} \right\rfloor \geq \lfloor 2k \log(em\varepsilon^{-1}) \rfloor = \lambda,\end{aligned}$$

where we make use of the fact that $\gamma \geq \varepsilon$ (since $d \geq D$), $e^{-1/3} \geq 2/3$, and $w \leq d$. Therefore, we have

$$\begin{aligned}\mathbb{P}\left[|D' - \mu| \geq p\varepsilon d\right] = \mathbb{P}\left[|D' - \mu| \geq \gamma\mu\right] &\leq \exp\left(-\left\lfloor \frac{\lfloor 2k \log(em\varepsilon^{-1}) \rfloor}{2} \right\rfloor\right) \\ &\leq \exp\left(-\lfloor k \log(em\varepsilon^{-1}) \rfloor\right) \leq \exp\left(-k \log(em\varepsilon^{-1}) + 1\right) \\ &\leq (em\varepsilon^{-1})^{-k} \cdot e = e^{-k+1} m^{-k} \varepsilon^k \leq m^{-k}\varepsilon.\end{aligned}$$

The total number of collections is $\sum_{i=1}^k \binom{m}{i} \leq \sum_{i=1}^k \frac{m^k}{k} = m^k$, so taking a union bound, the probability that $|D' - \mu| \geq p\varepsilon d$ for some collection is at most $m^k m^{-k}\varepsilon = \varepsilon$. ◀

Using Lemma 3, we can show that a good solution to the subsampled instance $\mathcal{I}'$ corresponds to a nearly-as-good solution to $\mathcal{I}$. This result is analogous to Corollary 9 from McGregor and Vu [15].

▶ **Corollary 5.** *Let OPT' be the optimal coverage for the subsampled problem instance $\mathcal{I}'$. If a choice of k sets $S_1, \ldots, S_k$ satisfies $|S_1' \cup \cdots \cup S_k'| \geq \beta \cdot \mathrm{OPT}'$, then with probability at least $1 - \varepsilon$, $|S_1 \cup \cdots \cup S_k| \geq (\beta - 2\varepsilon) \cdot \mathrm{OPT}$.*

Proof. Let $O_1, \ldots, O_k$ be an optimal solution to the unsubsampled problem instance $\mathcal{I}$. We have

$$\mathrm{OPT}' \geq |O_1' \cup \cdots \cup O_k'| \geq p \cdot |O_1 \cup \cdots \cup O_k| - p\varepsilon d \geq p(1 - \varepsilon) \cdot \mathrm{OPT},$$

where the second inequality follows from Lemma 3. Now let $S_1, \ldots, S_k$ be a collection of sets satisfying $|S_1' \cup \cdots \cup S_k'| \geq \beta \cdot \mathrm{OPT}$. Applying Lemma 3 once again, we have

$$\begin{aligned}|S_1 \cup \ldots \cup S_k| &\geq \frac{1}{p} \cdot |S_1' \cup \ldots \cup S_k'| - \varepsilon d \geq \frac{1}{p} \cdot \beta \cdot \mathrm{OPT}' - \varepsilon \cdot \mathrm{OPT} \\ &\geq \frac{\beta}{p} \cdot p(1 - \varepsilon) \cdot \mathrm{OPT} - \varepsilon \cdot \mathrm{OPT} \geq (\beta - 2\varepsilon) \cdot \mathrm{OPT}.\end{aligned}$$ ◀

Now $\mathcal{B}$ achieves an approximation factor of α in-expectation on the subsampled instance $\mathcal{I}'$. Therefore, by Corollary 5, with probability at least $1-\varepsilon$, $\mathcal{B}$ achieves an approximation factor of $\alpha - 2\varepsilon$ in-expectation. The unconditional expected approximation factor is therefore at least $(1-\varepsilon)(\alpha - 2\varepsilon) \geq (\alpha - 3\varepsilon)$. Dividing ε by 3 at the start (omitted from Algorithm 1 for brevity) corrects this to $\alpha - \varepsilon$ without changing the asymptotic space complexity.

Of course, the above reasoning is only valid if, in the at-least-$(1-\varepsilon)$-probability event that all the approximations in Lemma 3 hold, the instantiation $\mathcal{B}_{w_c}$ is not terminated.[12] Let $i'_{\max} = \arg\max_i |S'_i|$ be the index of the largest set in the subsampled instance. We have

$$d^*_{w_c} \leq d_{w_c} = \left|S'_{i'_{\max}}\right| \overset{*}{\leq} p_{w_c}\left|S_{i'_{\max}}\right| + p_{w_c}\varepsilon d \leq p_{w_c} d(1+\varepsilon) \leq 2p_{w_c} w_c (1+\varepsilon),$$

where the starred inequality follows from an application of Lemma 3 on the singleton collection containing only $S_{i'_{\max}}$. Therefore, the instantiation $\mathcal{B}_{w_c}$ is not terminated.

Space consumption. We now consider an arbitrary instantiation $\mathcal{B}_w$, where w is not necessarily the correct guess for d. Unlike n, m, and k, we do not assume that a maximum coverage streaming algorithm is provided with the maximum set size d at the start of the stream. Furthermore, at any point during the stream, it could be that the largest set has already been observed (i.e., $d^* = d$). Therefore, since $\mathcal{B}_w$ uses $O(d_w s)$ space, it must also use $O(d^*_w s)$ space. Now as long as $\mathcal{B}_w$ is not terminated, by Line 11, we have $d^*_w \leq 2p_w w(1+\varepsilon)$, so the space consumption of $\mathcal{B}_w$ is

$$O(d^*_w s) = O(p_w w(1+\varepsilon)s) = O(k\varepsilon^{-2}\log(em\varepsilon^{-1})w^{-1}ws) = \tilde{O}(k\varepsilon^{-2}s).$$

Each instantiation also requires the storage of the hash function h_w, but this takes just $O(\lambda) = \tilde{O}(k)$ space. There are $|W| = O(\log n)$ instantiations, so the total space complexity of GS($\mathcal{B}$) is $\tilde{O}(k\varepsilon^{-2}s)$. This completes the proof of Theorem 2.

2.3 Beating 1/2 via submodular maximisation

Submodular maximisation is a heavily studied problem in combinatorial optimisation that may be viewed as a generalisation of maximum coverage. A function $f : 2^V \to \mathbb{R}$ over a ground set V is said to be *submodular* if we have

$$f(e \mid X) \geq f(e \mid Y) \quad \text{for all } X \subseteq Y \subseteq V \text{ and } e \in V \setminus Y,$$

where $f(e \mid X) = f(X \cup \{e\}) - f(X)$ is the marginal gain of e given X. Submodular maximisation is the problem of choosing $A \subseteq V$ such that $f(A)$ is maximised. We consider a popular variant of the problem in which a cardinality constraint $|A| \leq k$ is applied and f is assumed to be non-negative and monotone.

Submodular maximisation has received a lot of attention in the streaming model [5, 17, 1]. In this model, the items of V arrive one at a time in the stream, and we assume access to an oracle for f, which may be queried in $O(1)$ time. In the arbitrary-arrival model, it is known that $\Omega(mk^{-3})$ space is required to achieve an approximation factor above 1/2, where $m = |V|$ is the length of the stream [8].[13] In the random-arrival model, however, the 1/2 barrier has been broken. This was first achieved by Norouzi-Fard et al. [17], who gave a $\tilde{O}(k)$-space

[12] Note that the inequality in Corollary 5 always holds in this event.

[13] This is the same result that states that $\Omega(mk^{-3})$ space is required to achieve approximation factors better than 1/2 for arbitrary-arrival set-streaming maximum coverage (cf. Table 1).

algorithm, called SALSA, which achieves an approximation factor of $1/2 + c_0$ in-expectation for a very small constant $c_0 > 0$. More recently, Agrawal et al. [1] gave an algorithm which achieves an approximation factor of $1 - 1/e - \varepsilon - o(1)$ in-expectation, where the $o(1)$ term is a function of k. The algorithm, which we call SMC after its authors, uses $\tilde{O}_\varepsilon(k)$ space, where a complicated exponential dependence on ε is suppressed. Liu et al. [14] improved the space complexity to $\tilde{O}(k\varepsilon^{-1})$ while maintaining the approximation factor; we call this improved algorithm SMC$^+$.

Submodular maximisation generalises maximum coverage: given a maximum coverage instance $\mathcal{I} = (\mathcal{F}, k)$, we simply set $V = \mathcal{F}$ and define f to be the coverage function $f(X) = |\cup_{S_i \in X} S_i|$. It is not hard to see that f is non-negative, monotone, and submodular. In the study of maximum coverage, however, we do *not* assume oracle access for evaluating coverage, so it must be evaluated explicitly. Therefore, if we use a submodular maximisation streaming algorithm for set-streaming maximum coverage, entire sets must be retained in memory, leading to an increase in space consumption proportional to d, the size of the largest set. Applying Theorem 2, however, we have the following key result.

► **Corollary 6.** *Suppose $\mathcal{B}$ achieves an approximation factor of α in-expectation for streaming submodular maximisation using $O(s)$ space. There exists an algorithm for set-streaming maximum coverage which, given $\varepsilon > 0$, achieves an approximation factor of $\alpha - \varepsilon$ in-expectation and uses $\tilde{O}(k\varepsilon^{-2}s)$ space.*

Applying this result to SALSA with $\varepsilon = c_1 := c_0/2$ yields GS-SALSA, which achieves an approximation factor of $1/2 + c_0 - \varepsilon = 1/2 + c_1$ in-expectation using $\tilde{O}(k\varepsilon^{-2} \cdot k\varepsilon^{-1}) = \tilde{O}(k^2)$ space (note that ε in this case is constant). Applying the result to SMC$^+$ yields GS-SMC$^+$, which uses $\tilde{O}(k^2\varepsilon^{-3})$ space and achieves an approximation factor of $1 - 1/e - \varepsilon - o(1) - \varepsilon$ in-expectation. Dividing ε by 2 returns the approximation factor to $1 - 1/e - \varepsilon - o(1)$ with no change to the space complexity. Recall that achieving an approximation factor above $1 - 1/e$ requires high space [15], so for large k, this is essentially the optimal approximation. The $o(1)$ term decreases quite slowly, however, so for small k, GS-SALSA may be preferable.

3 Hardness Result for Edge-Streaming

In this section, we prove Theorem 1, which essentially says that no low-space algorithm can achieve a nontrivial approximation in the random-arrival edge-streaming model.

Our proof is a reduction from the heavily studied *t-party set disjointness problem*, DISJ$_t$. In this problem, t players are each given a set $S_i \subseteq [N]$, where the i^{th} player knows only S_i. Either all sets are pairwise disjoint (a YES instance), or are pairwise disjoint except for an element that is common to all t sets (a NO instance). Gronemeier [9] showed that any protocol that solves this problem with probability $2/3$ requires $\Omega(N/t)$ bits of communication, even if the players may use public randomness. The hardness instance for this problem is such that we may assume that $|S_1| = \ldots = |S_t| = cN/t$ for an arbitrarily small constant $c > 0$.

Our proof combines ideas from two existing approaches, both of which are also reductions from DISJ$_t$. The first is a result due to Andoni et al. [3], who proved a near-tight space lower bound for the problem of *frequency moment estimation* on random-arrival streams. Given a stream $\langle a_1, \ldots, a_l \rangle$, where each $a_i \in [n]$, the k^{th} frequency moment is defined as $F_k := \sum_{i \in [n]} f_i^k$, where $f_i = |\{j : a_j = i\}|$ is the number of times i appears in the stream. Andoni et al. [3] studied the problem of 0.5-estimating this quantity, which is known to require $\tilde{\Theta}(n^{1-2/k})$ space in the arbitrary-arrival model. They showed that $\Omega(n^{1-2.5/k}/\log n)$ space is necessary in the random-arrival model, almost matching the upper bound. In their proof, a stream of integers is constructed such that we have $F_k = l$ for a YES instance and $F_k \geq 2l$ for a NO instance. Our proof is very similar, but we include all the details for completeness. The main difference in our proof is that we need to construct a stream of *edges*, rather than a stream of integers. We use an idea from Indyk and Vakilian [11], who proved that $\Omega(\alpha^2 m)$ space is required to α-approximate maximum coverage in the arbitrary-arrival edge-streaming model. Their proof involves the construction of a stream of edges such that the maximum coverage is 1 for a YES instance and $1/\alpha$ for a NO instance. We make a small change to this construction to ensure that the stream is in random order.

We start with two slightly reparameterised preliminary results from Andoni et al. [3]. Throughout this section, we assume that α and δ have been fixed.

▶ **Lemma 7.** *Let $\mathcal{W} = \{I_1, \ldots, I_t\}$ be $t = l^{2\delta/5}$ random intervals from*

$$\mathrm{Cycle}_{l,w} \coloneqq \{[i-1 \pmod{l}+1, \ldots, w+i-2 \pmod{l}+1] : i \in [l]\},$$

where $w = c_1 l^{1-3\delta/5}$ and $c_1 > 0$ is a sufficiently small absolute constant. With probability at least 0.99,

1. *$I_{i_1} \cap I_{i_2} \cap I_{i_3} = \emptyset$ for any $i_1 < i_2 < i_3$.*
2. *$|\{(i_1, i_2) : i_1 < i_2,\ I_{i_1} \cap I_{i_2} \neq \emptyset\}| \leq \sqrt{t}$.*

The set $\mathrm{Cycle}_{l,w}$ is simply the set of size-w intervals from $[l]$, including those that "wrap around" from l to 1. For example, $\mathrm{Cycle}_{4,3} = \{[1,2,3],[2,3,4],[3,4,1],[4,1,2]\}$.

▶ **Lemma 8.** *Consider a uniformly random subset $S \subseteq [l]$ of size $t = l^{2\delta/5}$. For some sufficiently small absolute constant $c_2 > 0$, with probability at least 0.99, for each $j_1, j_2 \in S$, $|j_2 - j_1| \geq c_2 l^{1-4\delta/5}$.*

We also require the following result.

▶ **Lemma 9.** *Let X be the number of unique values produced by $\lceil t/4 \rceil$ independent, uniformly random draws from $[t]$ (with replacement). For sufficiently large t, $\mathbb{P}(X \geq t/6) \geq 0.99$.*

Proof. Let $X_i = \mathbf{1}(i \text{ is drawn at least once})$, then $X = \sum_{i=1}^{t} X_i$. We have

$$\mu \coloneqq \mathbb{E}[X] = t\mathbb{E}[X_1] = t\left(1 - (1-1/t)^{\lceil t/4 \rceil}\right) \geq t\left(1 - e^{-1/4}\right) \geq t/5.$$

By a Chernoff bound for negatively correlated Boolean random variables where we set $\beta = 6/5$, we have

$$\mathbb{P}\left(X < \frac{t}{6}\right) = \mathbb{P}\left(X < \frac{t/5}{\beta}\right) \leq \mathbb{P}\left(X < \frac{\mu}{\beta}\right) \leq \left(\frac{e^{1/\beta - 1}}{\beta^\beta}\right)^\mu = c_3^\mu \leq c_3^{t/5},$$

where c_3 is an absolute constant less than 1. By setting $t \geq 5\log_{c_3}(0.01) \approx 11.9$, we get $\mathbb{P}(X < t/6) \leq 0.01$ and the result follows. ◀

We are now ready to prove Theorem 1. Let $\mathcal{S} = \{S_1, \ldots, S_t\}$ be an instance of DISJ_t where $t = l^{2\delta/5}$, $N = l^{1-\delta/5}$, and for all $i \in [t]$, we have $|S_i| = c_1 N/t = c_1 l^{1-3\delta/5} =: w$, where c_1 is as in Lemma 7. Any protocol that solves $\mathcal{S}$ with probability at least 2/3 requires $\Omega(N/t) = \Omega(l^{1-3\delta/5})$ bits of communication.

Let $\mathcal{A}$ be an s-space algorithm for random-arrival edge-streaming maximum coverage that achieves an approximation factor of α with probability at least 0.9 on streams containing l edges. We describe a protocol that uses $\mathcal{A}$ to solve $\mathcal{S}$ using $O(ts)$ bits of communication, and therefore deduce that $s = \Omega(l^{1-3\delta/5}/t) = \Omega(l^{1-\delta}) = \Omega(m^{1-\delta})$ (note that $m \leq l$, since l edges cannot possibly specify the contents of more than m non-empty sets).

Using public randomness, the players pick:

1. Intervals $\mathcal{W} = \{I_1 = [a_1, b_1], \ldots, I_t = [a_t, b_t]\}$ from $\text{Cycle}_{l,w}$, ordered such that $b_i \leq b_j$ if $i \leq j$. The intervals are chosen independently with replacement.
2. A permutation σ of $[2l]$.
3. A binary string r of length t^2/c_2, where c_2 is as in Lemma 8.

Each interval in $\mathcal{W}$ has size $|I_i| = w$. If $b_1 < w$, then at least one set "wraps around" from l to 1. In this case, terminate with failure. Each interval has probability $(w-1)/l$ of wrapping around, so by a simple union bound, $\mathbb{P}[b_1 < w] \leq t(w-1)/l \leq c_1 l^{-\delta/5}$. For large enough l, this value is at most 0.01. Also, if any $j \in [n]$ appears in three or more distinct intervals in $\mathcal{W}$, or if more than $\sqrt{t}$ pairs of intervals intersect, the protocol terminates with failure. By Lemma 7, this too occurs with probability at most 0.01.

The players construct, as described below, a length-l stream $U = \langle u_1, \ldots, u_l \rangle$ representing a maximum coverage problem instance $\mathcal{I} = (\mathcal{F} = \{H_1, \ldots, H_m\}, k = 1)$ in the edge-streaming model, where each $H_i \subseteq [t]$. Each edge is a set-element pair $u_j = (y_j, e_j)$ indicating that $e_j \in H_{y_j}$. The players take turns generating edges and feeding them to the stream of the algorithm $\mathcal{A}$. When it is no longer a player's turn to generate the next edge, the player sends the entire state of $\mathcal{A}$ to the appropriate player, which requires $O(s)$ bits of communication.

Constructing the stream. The construction is very similar to the one presented by Andoni et al. [3]. First, each player i randomly orders their set S_i to produce a string s^i of length w. Now consider u_j, the j^{th} element in the stream U. The player chosen to construct u_j depends on the number of intervals in $\mathcal{W}$ that contain j.

0. If j appears in none of the intervals, u_j is determined by player i, where $I_i = [a_i, b_i]$ is the interval with the smallest a_i such that $a_i > j$. If no such interval exists (which may occur towards the end of the stream) then u_j is determined by player t.
1. If j appears in just one interval I_i, u_j is determined by player i.
2. If j appears in two intervals I_i and I_{i+1}, we have a "clash" between players i and $i+1$. We partition $[l]$ into t^2/c_2 equally sized intervals, which we call *zones*. We say that zone u is a *0-zone* if $r_u = 0$, and a *1-zone* otherwise. If j is in a 0-zone, u_j is determined by player i. Otherwise, u_j is determined by player $i+1$. Each zone has length $w_2 := c_2 l^{1-4\delta/5}$.

Note that j never appears in more than two intervals, since this results in termination of the protocol. Figure 1 shows an example of how players might be chosen to construct the stream.

At the end of their interval, each player sends the state of $\mathcal{A}$ to the next player. Also, when two players' intervals overlap, they may need to send the state of $\mathcal{A}$ back and forth several times (e.g., in Figure 1, players 2 and 3 need to pass the state to each other three times). Assuming the protocol does not terminate, there are at most $\sqrt{t}$ clashes, and each clash results in at most $w/w_2 = O(\sqrt{t})$ messages being sent. The total number of messages is $O(t + \sqrt{t} \cdot \sqrt{t}) = O(t)$, so the communication complexity of the protocol is $O(ts)$.

Now suppose player i has been chosen to construct the edge $u_j = (y_j, e_j)$. The set ID y_j is chosen in exactly the same way as the integer a_j in Andoni et al. [3]: set $y_j = \sigma(x_j)$, where the definition of x_j differs for each of the three cases.

0. Set $x_j = l + j$ to be a *padding value.*
1. Set $x_j = s^i_q$ to be a *standard value* with probability 1/2, where j is the q^{th} element of I_i. Otherwise, set $x_j = l + j$ to be a padding value.
2. Set $x_j = s^i_q$ to be a standard value, where j is the q^{th} element of I_i.

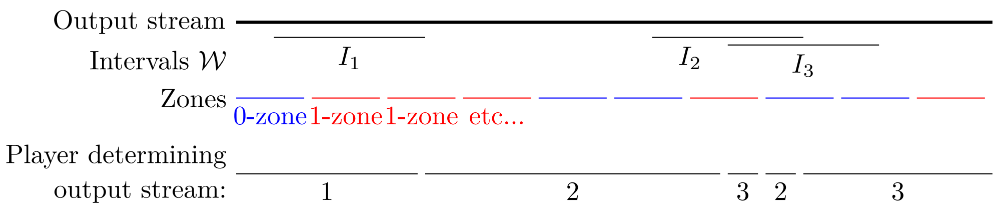

Figure 1 An example allocation of stream construction responsibility for a simple DISJ$_3$ instance in which there is a single overlap between two intervals (I_2 and I_3). Note that this is a non-terminating instance.

Table 2 A portion of an example reduction for a DISJ$_t$ NO instance with multiply occurring element 3. Column j denotes the index of the stream, while the strings s^2 and s^3 (which contain the contents of the sets S_2 and S_3) have been positioned in place of the intervals $I_2 = \{11, \ldots, 14\}$ and $I_3 = \{13, \ldots, 16\}$ for clarity. The "Player" column denotes the player who determines the edge $u_j = (\sigma(x_j), p_j)$. In Case 1 (e.g., $j = 11$), there are two different equally likely values for x_j; both values are shown. We assume that $l = 20$.

j	s^2	s^3	Zone	Player	Case	x_j
$\vdots$			$\vdots$	$\vdots$	$\vdots$	$\vdots$
10			0	2	0	$l + j = 30$
11	4		0	2	1	$s^2_1 = 4$ or $l + j = 31$
12	5		1	2	1	$s^2_2 = 5$ or $l + j = 32$
13	3	8	1	3	2	$s^3_1 = 8$
14	1	2	0	2	2	$s^2_4 = 1$
15		3	0	3	1	$s^3_3 = 3$ or $l + j = 35$
16		6	0	3	1	$s^3_4 = 6$ or $l + j = 36$
17			0	4	0	$l + j = 37$
$\vdots$			$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 2 shows an example of how the x_j are defined for a portion of the stream that includes two overlapping intervals. Observe that each occurrence of an element e in a set S_i has probability $1/2$ of appearing as a standard value $x_j = e$ for some j.[14]

It remains to choose the element e_j. In their proof for the arbitrary-arrival case, Indyk and Vakilian [11] set $e_j = i$ and insert the edge (x_j, e_j) into the stream, where x_j is always a standard value (the intervals are contiguous sections of the stream). The result is that each $H_e \in \mathcal{F}$ contains the IDs of the players holding the element e, so the maximum coverage is either 1 (in a YES instance) or t (in a NO instance). This idea does not work for our purposes, since if we set $e_j = i$, consecutive edges inserted by the same player will always have the same e_j. Instead, we set $e_j = p_j$, where $p_j \in [t]$ is a *randomly sampled* player ID.

By the same argument given by Andoni et al. [3], the stream is in (nearly[15]) random order. The only difference is the presence of p_j in each of our edges. However, as the p_j are randomly and independently sampled from $[t]$, they have no effect on whether the stream is randomly ordered.

[14] In Case 1, this probability comes from the random choice between a standard value and a padding value. In Case 2, this probability comes from the random choice of zone due to the binary string r.

[15] Certain orderings are impossible due to the termination with failure conditions. Since termination happens with probability at most 0.02, however, the biggest detrimental effect that this can possibly have on the performance of $\mathcal{A}$ is a decrease of 0.02 in the probability of successfully achieving the approximation factor.

Using the output of $\mathcal{A}$. What does the output of $\mathcal{A}$ tell us about the DISJ_t instance $\mathcal{S}$? The following claims are made assuming that the protocol does not terminate with failure.

▷ Claim 10. If $\mathcal{S}$ is a YES instance, the optimal coverage of $\mathcal{I}$ is 1.

Proof. First, observe that the x_j are all distinct, since all padding values are distinct from each other, all standard values are distinct from each other (as $\mathcal{S}$ is a YES instance), and no standard value can ever equal a padding value, as $s_q^i \le N < l < l + j$. Therefore, since σ is a permutation, the y_j are also all distinct, so each edge in the stream specifies a different set ID. Every set is therefore singleton, so the optimal coverage is 1. ◁

▷ Claim 11. If $\mathcal{S}$ is a NO instance, then with probability at least 0.97, the optimal coverage of $\mathcal{I}$ is at least $t/6$.

Proof. There is some $e \in [N]$ such that $e \in S_i$ for all $i \in [t]$. For each occurrence of e in a set S_i, there is some position in the stream where e appears as an edge $(\sigma(e), p_j)$ with probability $1/2$. Let V be this set of positions (e.g., in Table 2, the multiply occurring element is $e = 3$, and the set V includes 13 and 15). Since the intervals $\mathcal{W}$ are chosen randomly and the strings s^i are randomly ordered, V is a uniformly random subset of $[l]$, so by Lemma 8, with probability at least 0.99, no two elements of V fall within $c_2 l^{1-4\delta/5}$ of each other. This is precisely w_2, the size of each zone, so no two elements fall within the same zone. Each occurrence of e therefore appears in the stream with probability $1/2$ *independently*, so the number of occurrences that appear has a Binomial distribution $q \sim B(t, 1/2)$. When t is sufficiently large,[16] $q \ge t/4$ with probability at least 0.99.

Now consider these q appearances, $(\sigma(e), p_1), \ldots, (\sigma(e), p_q)$. Assuming that we indeed have $q \ge t/4$, by Lemma 9, the number of distinct values among $p_1, \ldots, p_q$ is at least $t/6$, so choosing $H_{\sigma(e)}$ yields a coverage of at least $t/6$. By a union bound on the three sources of error (Lemma 8, variation in the Binomial distribution, and Lemma 9), this occurs with probability at least 0.97. ◁

By taking l large enough, we can ensure that $t/6 \ge 1/\alpha$, in which case $\mathcal{A}$ is powerful enough to distinguish between YES and NO instances. Tallying up the various sources of error, $\mathcal{A}$ succeeds with probability at least 0.9, we avoid early termination with probability at least 0.98, and (in the case of a NO instance) Claim 11 holds with probability at least 0.97. Thus, by a union bound, the protocol succeeds with probability at least $1 - 0.1 - 0.02 - 0.03 = 0.85 \ge 2/3$. This completes the proof of Theorem 1.

References

1 Shipra Agrawal, Mohammad Shadravan, and Cliff Stein. Submodular Secretary Problem with Shortlists. In *10th ITCS*, pages 1:1–1:19, 2019.

2 Aris Anagnostopoulos, Luca Becchetti, Ilaria Bordino, Stefano Leonardi, Ida Mele, and Piotr Sankowski. Stochastic query covering for fast approximate document retrieval. *ACM Transactions on Information Systems*, 33(3):1–35, 2015.

3 Alexandr Andoni, Andrew McGregor, Krzysztof Onak, and Rina Panigrahy. Better bounds for frequency moments in random-order streams, 2008. `arXiv:0808.2222`.

4 Sepehr Assadi and Soheil Behnezhad. Beating two-thirds for random-order streaming matching. In *48th ICALP*, page 19, 2021.

[16] Note that we can make t arbitrarily large by taking l to be sufficiently large.

5 Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *20th ACM SIGKDD*, pages 671–680, 2014.

6 MohammadHossein Bateni, Hossein Esfandiari, and Vahab Mirrokni. Almost optimal streaming algorithms for coverage problems. In *29th ACM SPAA*, pages 13–23, 2017.

7 Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

8 Moran Feldman, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. The one-way communication complexity of submodular maximization with applications to streaming and robustness. In *52nd ACM STOC*, pages 1363–1374, 2020.

9 André Gronemeier. Asymptotically optimal lower bounds on the NIH-multi-party information complexity of the AND-function and disjointness. In *26th STACS*, pages 505–516, 2009.

10 Sudipto Guha and Andrew McGregor. Stream order and order statistics: Quantile estimation in random-order streams. *SIAM Journal on Computing*, 38(5):2044–2059, 2009.

11 Piotr Indyk and Ali Vakilian. Tight trade-offs for the maximum k-coverage problem in the general streaming model. In *38th ACM PODS*, pages 200–217, 2019.

12 Stephen Jaud, Anthony Wirth, and Farhana Choudhury. Maximum coverage in sublinear space, faster, 2023. `arXiv:2302.06137`.

13 Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *15th APPROX*, pages 231–242, 2012.

14 Paul Liu, Aviad Rubinstein, Jan Vondrák, and Junyao Zhao. Cardinality constrained submodular maximization for random streams. In *34th NeurIPS*, pages 6491–6502, 2021.

15 Andrew McGregor and Hoa T Vu. Better streaming algorithms for the maximum coverage problem. *Theory of Computing Systems*, 63:1595–1619, 2019.

16 Nimrod Megiddo, Eitan Zemel, and S Louis Hakimi. The maximum coverage location problem. *SIAM Journal on Algebraic Discrete Methods*, 4(2):253–261, 1983.

17 Ashkan Norouzi-Fard, Jakub Tarnawski, Slobodan Mitrovic, Amir Zandieh, Aidasadat Mousavifar, and Ola Svensson. Beyond 1/2-approximation for submodular maximization on massive data streams. In *35th ICML*, pages 3829–3838, 2018.

18 Barna Saha and Lise Getoor. On maximum coverage in the streaming model & application to multi-topic blog-watch. In *9th SDM*, pages 697–708, 2009.

19 Jeanette P Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff–Hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8(2):223–250, 1995.

20 Huiwen Yu and Dayu Yuan. Set coverage problems in a one-pass data stream. In *13th SDM*, pages 758–766, 2013.