Towards Tight Bounds for the Streaming Set Cover Problem

Sariel Har-Peled* Piotr Indyk† Sepideh Mahabadi‡ Ali Vakilian§

Abstract

We consider the classic Set Cover problem in the data stream model. For n elements and m sets we give a $O(1/\delta)$-pass algorithm with a strongly sub-linear $\widetilde{O}(mn^{\delta})$ space and logarithmic approximation factor[1]. This yields a significant improvement over the earlier algorithm of Demaine et al. [DIMV14] that uses exponentially larger number of passes. We complement this result by showing that the tradeoff between the number of passes and space exhibited by our algorithm is tight, at least when the approximation factor is equal to 1. Specifically, we show that any algorithm that computes set cover exactly using $(\frac{1}{2\delta}-1)$ passes must use $\widetilde{\Omega}(mn^{\delta})$ space. Furthermore, we consider the problem in the geometric setting where the elements are points in $\mathbb{R}^2$ and sets are either discs, axis-parallel rectangles, or fat triangles in the plane, and show that our algorithm (with a slight modification) uses the optimal $\widetilde{O}(n)$ space to find a logarithmic approximation in $O(1/\delta)$ passes.

Finally, we show that any randomized one-pass algorithm that distinguishes between covers of size 2 and 3 must use a linear (i.e., $\Omega(mn)$) amount of space. This is the first result showing that a randomized, approximate algorithm cannot achieve a sub-linear space bound. This indicates that using multiple passes might be necessary in order to achieve sub-linear space bounds for this problem while guaranteeing small approximation factors.

1. Introduction

The Set Cover problem is a classic combinatorial optimization task. Given a ground set of n elements $\mathsf{U} = \{e_1, \cdots, e_n\}$, and a family of m sets $\mathcal{F} = \{\mathsf{r}_1, \ldots, \mathsf{r}_m\}$, the goal is to select a subset $\mathcal{I} \subseteq \mathcal{F}$ such that $\mathcal{I}$ *covers* U, i.e., $\mathsf{U} \subseteq \bigcup_{S \in \mathcal{I}} S$, and the number of the sets in $\mathcal{I}$ is as small as possible. Set Cover is a well-studied problem with applications in many areas, including operations research [GW97], information retrieval and data mining [SG09], web host analysis [CKT10], and many others.

Although the problem of finding an optimal solution is NP-complete, a natural greedy algorithm which iteratively picks the "best" remaining set is widely used. The algorithm often finds solutions that are very close to optimal. Unfortunately, due to its sequential nature, this algorithm does not scale very well to massive data sets (e.g., see Cormode et al. [CKW10] for an experimental evaluation). This difficulty has motivated a considerable research effort whose goal was to design algorithms that are capable of handling large data efficiently on modern architectures. Of particular interest are *data stream* algorithms, which compute the solution using only a small number of

*Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; http://sarielhp.org/; sariel@illinois.edu.

†CSAIL; EECS; MIT; 32 Vassar Street; Cambridge, MA, 02139, USA; indyk@mit.edu.

‡CSAIL; EECS; MIT; 32 Vassar Street; Cambridge, MA, 02139, USA; mahabadi@mit.edu.

§CSAIL; EECS; MIT; 32 Vassar Street; Cambridge, MA, 02139, USA; vakilian@mit.edu.

[1]The notation $\widetilde{O}(f(n))$ is a short form for $O\big(f(n,m)\operatorname{polylog} f(n,m)\big)$. Similarly, $\widetilde{\Omega}(f(n,m)) = \Omega\big(f(n,m)/\operatorname{polylog} f(n,m)\big)$.

sequential passes over the data using a limited memory. In the *streaming* Set Cover problem [SG09], the set of elements U is stored in the memory in advance; the sets $\mathsf{r}_1, \cdots, \mathsf{r}_m$ are stored consecutively in a read-only repository and an algorithm can access the sets only by performing sequential scans of the repository. However, the amount of read-write memory available to the algorithm is limited, and is smaller than the input size (which could be as large as mn). The objective is to design an efficient approximation algorithm for the Set Cover problem that performs few passes over the data, and uses as little memory as possible.

The last few years have witnessed a rapid development of new streaming algorithms for the Set Cover problem, in both theory and applied communities, see [SG09, CKW10, KMVV13, ER14, DIMV14, CW16]. Figure 1.1 presents the approximation and space bounds achieved by those algorithms, as well as the lower bounds[2].

Result	Approximation	Passes	Space	R
Greedy	$\ln n$	1	$O(mn)$	
algorithm	$\ln n$	n	$O(n)$	
[SG09]	$O(\log n)$	$O(\log n)$	$O(n \log n)$	
[ER14]	$O(\sqrt{n})$	1	$\widetilde{O}(n)$	
[CW16]	$O(n^{\delta}/\delta)$	$1/\delta - 1$	$\widetilde{O}(n)$	
[Nis02]	$\frac{1}{2}\log n$	$O(\log n)$	$\widetilde{\Omega}(m)$	R
[DIMV14]	$O(4^{1/\delta}\rho)$	$O(4^{1/\delta})$	$\widetilde{O}(mn^{\delta})$	R
[DIMV14]	$O(1)$	$O(\log n)$	$\widetilde{\Omega}(mn)$	
Theorem 2.8	$O(\rho/\delta)$	$2/\delta$	$\widetilde{O}(mn^{\delta})$	R
Theorem 3.7	3/2	1	$\Omega(mn)$	R
Theorem B.4	1	$1/2\delta - 1$	$\widetilde{\Omega}(mn^{\delta})$	R
Geometric Set Cover(Theorem A.6)	$O(\rho_g/\delta)$	$3/\delta$	$\widetilde{O}(n)$	R
s-Sparse Set Cover(Theorem C.6)	1	$1/2\delta - 1$	$\widetilde{\Omega}(ms)$	R

Figure 1.1: Summary of past work and our results. The last column indicates if the scheme is randomized, ρ denotes the approximation factor of an off-line algorithm solving Set Cover, which is $\ln n$ for the greedy, and 1 for exponential algorithm. Similarly, ρ_g denotes the approximation factor of an off-line algorithm for the geometric Set Cover. We allow any $\delta > 0$ except in the geometric Set Cover in which $\delta \leq \frac{1}{4}$. Finally, in the s-Sparse Set Cover problem, $s \leq n^{\delta}$ denotes an upper bound on the sizes of the input sets. Moreover, [ER14] and [CW16] proved that their algorithms are tight. Here, and in the rest of the paper, all log are in base two.

Related work. The semi-streaming Set Cover problem was first studied by Saha and Getoor [SG09]. Their result for Max k-Cover problem implies a $O(\log n)$-pass $O(\log n)$-approximation algorithm for the Set Cover problem that uses $\widetilde{O}(n)$ space. In $\widetilde{O}(n)$ space regime, Emek and Rosen studied designing one-pass streaming algorithms for the Set Cover problem [ER14] and gave a *deterministic* greedy based $O(\sqrt{n})$-approximation for the problem. Moreover they proved that their algorithm is tight, even for randomized algorithms. The lower/upper bound results of [ER14] applied also to a generalization of the Set Cover problem, the ε-Partial Set Cover$(\mathsf{U}, \mathcal{F})$ problem in

[2]Note that the simple greedy algorithm can be implemented by either storing the whole input (in one pass), or by iteratively updating the set of yet-uncovered elements (in at most n passes).

which the goal is to cover $(1-\varepsilon)$ fraction of elements U and the size of the solution is compared to the size of an optimal cover of $\textsf{Set Cover}(\mathsf{U}, \mathcal{F})$. Very recently, Chakrabarti and Wirth extended the result of [ER14] and gave a trade-off streaming algorithm for the Set Cover problem in *multiple passes* [CW16]. They gave a *deterministic* algorithm with p passes over the data stream that returns a $(p+1)n^{1/(p+1)}$-approximate solution of the Set Cover problem in $\widetilde{O}(n)$ space. Moreover they proved that achieving $0.99n^{1/(p+1)}/(p+1)^2$ in p passes using $\widetilde{O}(n)$ space is not possible even for randomized protocols which shows that their algorithm is tight up to a factor of $(p+1)^3$. Their result also works for the ε-Partial Set Cover problem.

In a different regime which was first studied by Demaine et al., the goal is to design a "low" approximation algorithms (depending on the computational model, it could be $O(\log n)$ or $O(1)$) in the smallest possible space [DIMV14]. They proved that any constant pass *deterministic* $(\log n/2)$-approximation algorithm of the Set Cover problem requires $\widetilde{\Omega}(mn)$ space. It shows that unlike the results in $\widetilde{O}(n)$-space regime, to obtain a sublinear "low" approximation streaming algorithm for the Set Cover problem in a constant number of passes, using *randomness* is necessary. Moreover, [DIMV14] presented a $O(4^{1/\delta})$-approximation algorithm that makes $O(4^{1/\delta})$ passes and uses $\widetilde{O}(mn^{\delta})$ memory space.

The Set Cover problem is not polynomially solvable even in the restricted instances with points in $\mathbb{R}^2$ as elements, and geometric objects (either all disks or axis parallel rectangles or fat triangles) in plane as sets [FG88, FPT81, HQ15]. As a result, there has been a large body of work on designing approximation algorithms for the geometric Set Cover problems. See for example [MRR14, AP14, AES10, CV07] and references therein.

1.1. Our results

Despite the progress outlined above, however, some basic questions still remained open. In particular:

(A) Is it possible to design a *single* pass streaming algorithm with a "low" approximation factor[3] that uses sublinear (i.e., $o(mn)$) space?

(B) If such single pass algorithms are not possible, what are the achievable trade-offs between the number of passes and space usage?

(C) Are there special instances of the problem for which more efficient algorithms can be designed?

In this paper, we make a significant progress on each of these questions. Our upper and lower bounds are depicted in Figure 1.1.

On the algorithmic side, we give a $O(1/\delta)$-pass algorithm with a strongly sub-linear $\widetilde{O}(mn^{\delta})$ space and logarithmic approximation factor. This yields a significant improvement over the earlier algorithm of Demaine et al. [DIMV14] which used exponentially larger number of passes. The trade-off offered by our algorithm matches the lower bound of Nisan [Nis02] that holds at the endpoint of the trade-off curve, i.e., for $\delta = \Theta(1/\log n)$, up to poly-logarithmic factors in space[4]. Furthermore, our algorithm is very simple and succinct, and therefore easy to implement and deploy.

Our algorithm exhibits a natural tradeoff between the number of passes and space, which resembles tradeoffs achieved for other problems [GM07, GM08, GO13]. It is thus natural to conjecture that this tradeoff might be tight, at least for "low enough" approximation factors. We present the first step in this direction by showing a lower bound for the case when the approximation factor

[3] Note that the lower bound in [DIMV14] excluded this possibility only for deterministic algorithms, while the upper bound in [ER14, CW16] suffered from a polynomial approximation factor.

[4] Note that to achieve a logarithmic approximation ratio we can use an off-line algorithm with the approximation ratio $\rho = 1$, i.e., one that runs in exponential time (see Theorem 2.8).

is equal to 1, i.e., the goal is to compute the optimal set cover. In particular, by an information theoretic lower bound, we show that any *streaming* algorithm that computes set cover using $(\frac{1}{2\delta}-1)$ passes must use $\widetilde{\Omega}(mn^{\delta})$ space (even assuming exponential computational power). Furthermore, we show that a stronger lower bound holds if all the input sets are sparse, that is if their cardinality is at most s. We prove a lower bound of $\widetilde{\Omega}(ms)$ for $s = O(n^{\delta})$.

We also consider the problem in the geometric setting in which the elements are points in $\mathbb{R}^2$ and sets are either discs, axis-parallel rectangles, or fat triangles in the plane. We show that a slightly modified version of our algorithm achieves the *optimal* $\widetilde{O}(n)$ space to find a $O(\rho/\delta)$-approximation in $O(1/\delta)$ passes.

Finally, we show that any randomized one-pass algorithm that distinguishes between covers of size 2 and 3 must use a linear (i.e., $\Omega(mn)$) amount of space. This is the first result showing that a randomized, approximate algorithm cannot achieve a sub-linear space bound. This indicates that using multiple passes might be necessary in order to achieve sub-linear space bounds while guaranteeing small approximation factors.

1.1.1. Our techniques

Basic idea. Our algorithm is based on the idea that whenever a large enough set is encountered, we can immediately add it to the cover. Specifically, we guess (up to factor two) the size of the optimal cover k. Thus, a set is "large" if it covers at least $1/k$ fraction of the remaining elements. A small set, on the other hand, can cover only a "few" elements, and we can store (approximately) what elements it covers by storing (in memory) an appropriate random sample. At the end of the pass, we have (in memory) the projections of "small" sets onto the random sample, and we compute the optimal set cover for this projected instance using an offline solver. By carefully choosing the size of the random sample, this guarantees that only a small fraction of the set system remains uncovered. The algorithm then makes an additional pass to find the residual set system (i.e., the yet uncovered elements), making two passes in each iteration, and continuing to the next iteration.

Thus, one can think about the algorithm as being based on a simple iterative "dimensionality reduction" approach. Specifically, in two passes over the data, the algorithm selects a "small" number of sets that cover all but $n^{-\delta}$ fraction of the uncovered elements, while using only $\widetilde{O}(mn^{\delta})$ space. By performing the reduction step $1/\delta$ times we obtain a complete cover. The dimensionality reduction step is implemented by computing a small cover for a *random subset* of the elements, which also covers the vast majority of the elements in the ground set. This ensures that the remaining sets, when restricted to the random subset of the elements, occupy only $\widetilde{O}(mn^{\delta})$ space. As a result the procedure avoids a complex set of recursive calls as presented in Demaine et al. [DIMV14], which leads to a simpler and more efficient algorithm.

Geometric results. Further using techniques and results from computational geometry we show how to modify our algorithm so that it achieves better bounds for the Set Cover problems on geometric instances. In particular, we show that it gives a $O(1/\delta)$-pass $O(\rho/\delta)$-approximation algorithm using $\widetilde{O}(n)$ space when the elements are points in $\mathbb{R}^2$ and the sets are either discs, axis parallel rectangles, or fat triangles in the plane. In particular, we use the following surprising property of the set systems that arise out of points and disks: the the number of sets is nearly linear as long as one considers only sets that contain "a few" points.

More surprisingly, this property extends, with a twist, to certain geometric range spaces that might have quadratic number of shallow ranges. Indeed, it is easy to show an example of n points in the plane, where there are $\Omega(n^2)$ distinct rectangles, each one containing exactly two points, see Figure 1.2. However, one can "split" such ranges into a small number of canonical sets, such that the number of shallow sets in the canonical set system is near linear. This enables us to store

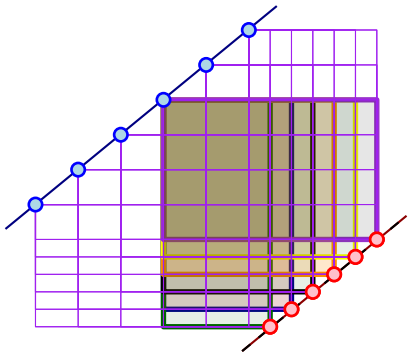

Figure 1.2: Consider two parallel lines in the plane with positive slope. Place $n/2$ points on each line such that all the points on the top line lie above and to the left of all the points on the bottom line. Let the set of rectangles for this instance be all the rectangles which have a point on the top line as their upper left corner and a point on the bottom line as their lower right corner. Clearly, we have $n^2/4$ distinct rectangles (i.e., sets), each containing two points. As such, we cannot afford to store explicitly in memory the set system, since it requires too much space.

explicitly in memory the small canonical sets encountered during the scan, and still use only near linear space.

We note that the idea of splitting ranges into small canonical ranges is an old idea in orthogonal range searching. It was used by Aronov et al. [AES10] for computing small ε-nets for these range spaces. The idea in the form we use, was further formalized by Ene et al. [EHR12].

Lower bound. The lower bounds for multi-pass algorithms of the Set Cover problem are obtained via a careful reduction from Intersection Set Chasing. The latter problem is a communication complexity problem where n players need to solve a certain "set-disjointness-like" problem in p rounds. A recent paper [GO13] showed that this problem requires $\frac{n^{1+\Omega(1/p)}}{p^{O(1)}}$ bits of communication complexity for p rounds. This yields our desired trade-off of $\widetilde{\Omega}(mn^{\delta})$ space in $1/2\delta$ passes for exact protocols of Set Cover in the communication complexity and hence in the streaming model. Furthermore, we show a stronger lower bound on memory space of sparse instances of Set Cover in which all input sets have cardinality at most s. By a reduction from a variant of Equal Pointer Chasing which maps the problem to a *sparse* instance of Set Cover, we show that in order to have an exact streaming algorithm of s-Sparse Set Cover with $o(ms)$ space, $\Omega(\log n)$ passes is necessary. More precisely, any $(\frac{1}{2\delta}-1)$-pass *exact* randomized algorithm of s-Sparse Set Cover requires $\widetilde{\Omega}(ms)$ memory space, if $s \leq n^{\delta}$.

Our single pass lower bound proceeds by showing a lower bound for a one-way communication complexity problem in which one party (Alice) has a collection of sets, and the other party (Bob) needs to determine whether the complement of his set is covered by one of the Alice's sets. We show that if Alice's sets are chosen at random, then Bob can decode Alice's input by employing a small collection of "query" sets. This implies that the amount of communication needed to solve the problem is linear in the description size of Alice's sets, which is $\Omega(mn)$.

```
iterSetCover((U, F), δ):
  // Try in parallel all possible (2-approx) sizes of optimal cover
  for k ∈ {2^i | 0 ≤ i ≤ log n} do in parallel:        // n = |U|
      sol ← ∅
      Repeat for 1/δ times
          Let S be a sample of U of size cρkn^δ log m log n
          L ← S,    F_S ← ∅
          for r ∈ F do                    // By doing one pass
              if |L ∩ r| ≥ |S|/k then     // Size Test
                  sol ← sol ∪ {r}
                  L ← L \ r
              else
                  F_S ← F_S ∪ {r ∩ L} // Store explicitly in memory the set r ∩ L
          D ← algOfflineSC(S, F_S, k),        sol ← sol ⋃ D
          U ← U \ ⋃_{r∈sol} r    // By doing additional pass over data
  return best sol computed in all parallel executions.
```

Figure 1.3: A tight streaming algorithm for the (unweighted) Set Cover problem. Here, **algOfflineSC** is an offline solver for Set Cover that provides ρ-approximation, and c is some appropriate constant.

2. Streaming algorithm for set cover

2.1. Algorithm

In this section, we design an efficient streaming algorithm for the Set Cover problem that matches the lower bound results we already know about the problem. In the Set Cover problem, for a given set system $(\mathsf{U}, \mathcal{F})$, the goal is to find a subset $\mathcal{I} \subseteq \mathcal{F}$, such that $\mathcal{I}$ covers U and its cardinality is minimum. In the following, we sketch the **iterSetCover** algorithm (see also Figure 1.3).

In the **iterSetCover** algorithm, we have access to the **algOfflineSC** subroutine that solves the given Set Cover instance offline (using linear space) and returns a ρ-approximate solution where ρ could be anywhere between 1 and $\Theta(\log n)$ depending on the computational model one assumes. Under exponential computational power, we can achieve the optimal cover of the given instance of the Set Cover ($\rho = 1$); however, under $\mathbf{P} \neq \mathbf{NP}$ assumption, ρ cannot be better than $c \cdot \ln n$ where c is a constant [AMS06, RS97] given polynomial computational power.

Let $n = |\mathsf{U}|$ be the initial number of elements in the given ground set. The **iterSetCover** algorithm, needs to guess (up to a factor of two) the size of the optimal cover of $(\mathsf{U}, \mathcal{F})$. To this end, the algorithm tries, in parallel, all values k in $\{2^i \mid 0 \leq i \leq \log n\}$. This step will only increase the memory space requirement by a factor of $\log n$.

Consider the run of the **iterSetCover** algorithm, in which the guess k is correct (i.e., $|\mathrm{OPT}| \leq k < 2|\mathrm{OPT}|$, where OPT is an optimal solution). The idea is to go through $O(1/\delta)$ iterations such that each iteration only makes two passes and at the end of each iteration the number of uncovered elements reduces by a factor of n^δ. Moreover, the algorithm is allowed to use $\widetilde{O}(mn^\delta)$ space.

In each iteration, the algorithm starts with the current ground set of uncovered elements U, and copies it to a *leftover* set L. Let S be a large enough uniform sample of elements U. In a single pass, using S, we estimate the size of all large sets in $\mathcal{F}$ and add $\mathsf{r} \in \mathcal{F}$ to the solution $\mathtt{sol}$ immediately (thus avoiding the need to store it in memory). Formally, if r covers at least $\Omega(|\mathsf{U}|/k)$

yet-uncovered elements of L then it is a heavy set, and the algorithm immediately adds it to the output cover. Otherwise, if a set is small, i.e., its covers less than $|\mathsf{U}|/k$ uncovered elements of L, the algorithm stores the set r in memory. Fortunately, it is enough to store its projection over the sampled elements explicitly (i.e., $\mathsf{r} \cap \mathsf{L}$) – this requires remembering only the $O(|\mathsf{S}|/k)$ indices of the elements of $\mathsf{r} \cap \mathsf{L}$.

In order to show that a solution of the Set Cover problem over the sampled elements is a good cover of the initial Set Cover instance, we apply the *relative (p, ε)-approximation* sampling result of [HS11] (see Definition 2.4) and it is enough for S to be of size $O(\rho k n^{\delta} \log m \log n)$. Using relative (p, ε)-approximation sampling, we show that after two passes the number of uncovered elements is reduced by a factor of n^{δ}.

Since in each iteration we pick $O(\rho k)$ sets and the number of uncovered elements decreases by a factor of n^{δ}, after $1/\delta$ iterations the algorithm picks $O(\rho k/\delta)$ sets and covers all elements. Moreover, the memory space of the whole algorithm is $O(\rho m n^{\delta} \log m \log n)$ (see Lemma 2.2).

2.2. Analysis

In the rest of this section we prove that the **iterSetCover** algorithm with high probability returns a $O(\rho/\delta)$-approximate solution of Set Cover$(\mathsf{U}, \mathcal{F})$ in $2/\delta$ passes using $\widetilde{O}(mn^{\delta})$ memory space.

Lemma 2.1. *The number of passes the* **iterSetCover** *algorithm makes is* $2/\delta$.

Proof: In each of the $1/\delta$ iterations of the **iterSetCover** algorithm, the algorithm makes two passes. In the first pass, based on the set of sampled elements S, it decides whether to pick a set or keep its projection over S in the memory. Then the algorithm calls **algOfflineSC** which does not require any passes over $\mathcal{F}$. The second pass is for computing the set of uncovered elements at the end of the iteration. We need this pass because we only know the projection of the sets we picked in the current iteration over S and not over the original set of uncovered elements. Thus, in total we make $2/\delta$ passes. Also note that for different guesses for the value of k, we run the algorithm in parallel and hence the total number of passes remains $2/\delta$. ■

Lemma 2.2. *The memory space used by the* **iterSetCover** *algorithm is* $\widetilde{O}(mn^{\delta})$.

Proof: In each iteration of the algorithm, it picks during the first pass at most m sets (more precisely at most k sets) which requires $O(m \log m)$ memory. Moreover, in the first pass we keep the projection of the sets whose projection over the uncovered sampled elements has size at most $|\mathsf{S}|/k$. Since there are at most m such sets, the total required space for storing the projections is bounded by $O\big(\rho m n^{\delta} \log m \log n\big)$.

Since in the second pass the algorithm only updates the set of uncovered elements, the amount of space required in the second pass is $O(n)$. Thus, the total required space to perform each iteration of the **iterSetCover** algorithm is $\widetilde{O}(mn^{\delta})$. Moreover, note that the algorithm does not need to keep the memory space used by the earlier iterations; thus, the total space consumed by the algorithm is $\widetilde{O}(mn^{\delta})$. ■

Next we show the sets we picked before calling **algOfflineSC** has large size on U.

Lemma 2.3. *With probability at least* $1 - m^{-c}$ *all sets that pass the "Size Test" in the* **iterSetCover** *algorithm have size at least* $|\mathsf{U}|/ck$.

Proof: Let r be a set of size less than $|\mathsf{U}|/ck$. In expectation, $|\mathsf{r}\cap\mathsf{S}|$ is less than $(|\mathsf{U}|/ck)\cdot(|\mathsf{S}|/|\mathsf{U}|) = \rho n^{\delta}\log m\log n$. Thus using Chernoff bound for large enough c,

$$\Pr(|\mathsf{r}\cap\mathsf{S}| \geq (1+(c-1))(\rho n^{\delta}\log m\log n)) \leq \exp\left(-\frac{(c-1)^2\rho n^{\delta}\log m\log n}{4}\right) \leq (1/m)^{c+1}.$$

Applying the union bound, with probability at least $1-m^{-c}$, all sets that pass "Size Test" have size at least $|\mathsf{U}|/ck$. ∎

In what follows we define the *relative* (p,ε)*-approximation* sample of a set system and mention the result of Har-Peled and Sharir [HS11] on the minimum required number of sampled elements to get a relative (p,ε)-approximation of the given set system.

Definition 2.4. Let $(\mathsf{V},\mathcal{H})$ be a set system, i.e., V is a set of elements and $\mathcal{H}\subseteq 2^{\mathsf{V}}$ is a family of subsets of the ground set V. For given parameters $0<\varepsilon,p<1$, a subset $Z\subseteq\mathsf{V}$ is a *relative* (p,ε)*-approximation* for $(\mathsf{V},\mathcal{H})$, if for each $\mathsf{r}\in\mathcal{H}$, we have that if $|\mathsf{r}|\geq p|\mathsf{V}|$ then $(1-\varepsilon)\dfrac{|\mathsf{r}|}{|\mathsf{V}|} \leq \dfrac{|\mathsf{r}\cap Z|}{|Z|} \leq (1+\varepsilon)\dfrac{|\mathsf{r}|}{|\mathsf{V}|}$. If the range is light (i.e., $|\mathsf{r}|<p|\mathsf{V}|$) then it is required that $\dfrac{|\mathsf{r}|}{|\mathsf{V}|}-\varepsilon p \leq \dfrac{|\mathsf{r}\cap Z|}{|Z|} \leq \dfrac{|\mathsf{r}|}{|\mathsf{V}|}+\varepsilon p$. Namely, Z is $(1\pm\varepsilon)$-multiplicative good estimator for the size of ranges that are at least p-fraction of the ground set.

The following lemma is a simplified variant of a result in Har-Peled and Sharir [HS11] – indeed, a set system with M sets, can have VC dimension at most $\log M$. This simplified form also follows by a somewhat careful but straightforward application of Chernoff's inequality.

Lemma 2.5. *Let $(\mathsf{U},\mathcal{F})$ be a finite set system, and p,ε,q be parameters. Then, a random sample of U of size $\frac{c'}{\varepsilon^2 p}\left(\log|\mathcal{F}|\log\frac{1}{p}+\log\frac{1}{q}\right)$, for an absolute constant c' is a relative (p,ε)-approximation, for all ranges in $\mathcal{F}$, with probability at least $(1-q)$.*

Lemma 2.6. *Assuming $|\mathrm{OPT}|\leq k\leq 2|\mathrm{OPT}|$, after any iteration, with probability at least $1-m^{1-c/4}$ the number of uncovered elements decreases by a factor of n^{δ}, and this iteration adds $O(\rho|\mathrm{OPT}|)$ sets to the output cover.*

Proof: Let $\mathsf{V}\subseteq\mathsf{U}$ be the set of uncovered elements at the beginning of the iteration and note that the total number of sets that is picked during the iteration with high probability is at most $(1+\rho)k$ (see Lemma 2.3). Consider all possible such covers, that is $\mathcal{G}=\left\{\mathcal{F}'\subseteq\mathcal{F}\;\middle|\;|\mathcal{F}'|\leq(1+\rho)k\right\}$, and observe that $|\mathcal{G}|\leq m^{(1+\rho)k}$. Let $\mathcal{H}$ be the collection that contains all possible sets of uncovered elements at the end of the iteration, defined as $\mathcal{H}=\left\{\mathsf{V}\setminus\bigcup_{\mathsf{r}\in\mathcal{C}}\mathsf{r}\;\middle|\;\mathcal{C}\in\mathcal{G}\right\}$. Moreover, set $p=2/n^{\delta}$, $\varepsilon=1/2$ and $q=m^{-c}$ and note that $|\mathcal{H}|\leq|\mathcal{G}|\leq m^{(1+\rho)k}$. Since $\frac{c'}{\varepsilon^2 p}(\log|\mathcal{H}|\log\frac{1}{p}+\log\frac{1}{q})\leq c\rho k n^{\delta}\log m\log n=|\mathsf{S}|$ for large enough c, by Lemma 2.5, S is a relative (p,ε)-approximation with $(1-q)$ probability. Let $\mathcal{D}\subseteq\mathcal{F}$ be the collection of sets picked during the iteration which covers all elements in S. Since S is a relative (p,ε)-approximation sample of $(\mathsf{V},\mathcal{H})$ with probability at least $1-m^{-c}$, the number of uncovered elements of V (or U) by $\mathcal{D}$ is at most $\varepsilon p|\mathsf{V}|=|\mathsf{U}|/n^{\delta}$.

Hence, in each iteration we pick $O(\rho k)$ sets and at the end of iteration the number of uncovered elements reduces by n^{δ}. ∎

Lemma 2.7. *The* **iterSetCover** *algorithm computes a set cover of $(\mathsf{U},\mathcal{F})$, whose size is within a $O(\rho/\delta)$ factor of the size of an optimal cover with probability at least $1-m^{1-c/4}$.*

Proof: Consider the run of **iterSetCover** for which the value of k is between $|\mathrm{OPT}|$ and $2|\mathrm{OPT}|$. In each of the $(1/\delta)$ iterations made by the algorithm, by Lemma 2.6, the number of uncovered elements decreases by a factor of n^{δ} where n is the number of initial elements to be covered by the sets. Moreover, the number of sets picked in each iteration is $O(\rho k)$. Thus after $(1/\delta)$ iterations, all elements would be covered and the total number of sets in the solution is $O(\rho|\mathrm{OPT}|/\delta)$. Moreover by Lemma 2.6, the success probability of all the iterations, is at least $1 - \frac{1}{\delta m^{c/4}} \geq 1 - (1/m)^{c/4-1}$. ■

Theorem 2.8. *The* Iterative-Set-Cover$(\mathsf{U}, \mathcal{F}, \delta)$ *algorithm makes* $2/\delta$ *passes, uses* $\widetilde{O}(mn^{\delta})$ *memory space, and finds a* $O(\rho/\delta)$*-approximate solution of the* Set Cover *problem with high probability.*

Furthermore, the **iterSetCover** *algorithm matches the known lower bound on the memory space of the streaming* Set Cover *problem up to a* $\mathrm{polylog}(m)$ *factor where* m *is the number of sets in the input.*

Proof: The first part of the proof follows from Lemma 2.1, Lemma 2.2, and Lemma 2.7.

As for the lower bound, note that by a result of Nisan [Nis02], any randomized $(\frac{\log n}{2})$-approximation protocol of set cover for $(\mathsf{U}, \mathcal{F})$ in the one-way communication model requires $\Omega(m)$ bits of communication, no matter how many number of rounds it makes. This implies that any randomized $O(\log n)$-pass, $(\frac{\log n}{2})$-approximation algorithm of Set Cover$(\mathsf{U}, \mathcal{F})$ requires $\widetilde{\Omega}(m)$ space, even under the exponential computational power assumption.

By the above, the **iterSetCover** algorithm makes $O(1/\delta)$ passes and uses $\widetilde{O}(mn^{\delta})$ space to return a $O(\frac{1}{\delta})$-approximate solution under the exponential computational power assumption ($\rho = 1$). Thus by letting $\delta = c/\log n$, we will have a $(\frac{\log n}{2})$-approximation streaming algorithm using $\widetilde{O}(m)$ space which is optimal up to a factor of $\mathrm{polylog}(m)$. ■

Theorem 2.8 provides a strong indication that our trade-off algorithm is optimal.

3. Lower Bound for Single Pass Algorithms

In this section, we study the Set Cover problem in the two-party communication model and give a tight lower bound on the communication complexity of the randomized protocols solving the problem in a single round. In the two-party Set Cover, we are given a set of elements U and there are two players Alice and Bob where each of them has a collection of subsets of U, $\mathcal{F}_A$ and $\mathcal{F}_B$. The goal for them is to find a minimum size cover $\mathcal{C} \subseteq \mathcal{F}_A \cup \mathcal{F}_B$ covering U with communicating the fewest number of bits.

Our main lower bound result for the single pass protocols of Set Cover is the following theorem which implies that the naive approach in which one party sends all of its sets to the the other one is optimal.

Theorem 3.1. *Any single round randomized protocol that approximates* Set Cover$(\mathsf{U}, \mathcal{F})$ *within a factor better than* $3/2$ *and error probability* $O(m^{-c})$ *requires* $\Omega(mn)$ *bits of communication where* $n = |\mathsf{U}|$ *and* $m = |\mathcal{F}|$ *and* c *is a sufficiently large constant.*

We consider the case in which the parties want to decide whether there exists a cover of size 2 for U in $\mathcal{F}_A \cup \mathcal{F}_B$ or not. If any of the parties has a cover of size at most 2 for U, then it becomes trivial. Thus the question is whether there exist $\mathsf{r}_a \in \mathcal{F}_A$ and $\mathsf{r}_b \in \mathcal{F}_B$ such that $\mathsf{U} \subseteq \mathsf{r}_a \cup \mathsf{r}_b$.

A key observation is that to decide whether there exist $\mathsf{r}_a \in \mathcal{F}_A$ and $\mathsf{r}_b \in \mathcal{F}_B$ such that $\mathsf{U} \subseteq \mathsf{r}_a \cup \mathsf{r}_a$, one can instead check whether there exists $\mathsf{r}_a \in \mathcal{F}_A$ and $\mathsf{r}_b \in \mathcal{F}_B$ such that $\overline{\mathsf{r}_a} \cap \overline{\mathsf{r}_b} = \emptyset$. In other words we need to solve OR of a series of two-party Set Disjointness problems. In two-party Set

Disjointness problem, Alice and Bob are given subsets of U, r_a and r_b and the goal is to decide whether $\mathsf{r}_a \cap \mathsf{r}_b$ is empty or not with the fewest possible bits of communication. Set Disjointness is a well-studied problem in the communication complexity and it has been shown that any randomized protocol of Set Disjointness with $O(1)$ error probability requires $\Omega(n)$ bits of communication where $n = |\mathsf{U}|$ [BJKS04, KS92, Raz92].

We can think of the following extensions of the Set Disjointness problem.

- *Many vs One:*
  In this variant, Alice has m subsets of U, $\mathcal{F}_A$ and Bob is given a single set r_b. The goal is to determine whether there exists a set $\mathsf{r}_a \in \mathcal{F}_A$ such that $\mathsf{r}_a \cap \mathsf{r}_b = \emptyset$.

- *Many vs Many:*
  In this variant, each of Alice and Bob are given a collection of subsets of U and the goal for them is to determine whether there exist $\mathsf{r}_a \in \mathcal{F}_A$ and $\mathsf{r}_b \in \mathcal{F}_B$ such that $\mathsf{r}_a \cap \mathsf{r}_b = \emptyset$.

Note that deciding whether two-party Set Cover has a cover of size 2 is equivalent to solving the (Many vs Many)-Set Disjointness problem. Moreover, any lower bound for (Many vs One)-Set Disjointness clearly implies the same lower bound for the (Many vs Many)-Set Disjointness problem. In the following theorem we show that any single-round randomized protocol that solves (Many vs One)-Set Disjointness(m, n) with $O(m^{-c})$ error probability requires $\Omega(mn)$ bits of communication.

Theorem 3.2. *Any randomized protocol for* (Many vs One)-Set Disjointness(m, n) *with error probability* $O(m^{-c})$ *requires* $\Omega(mn)$ *bits of communication.*

The idea is to show that if there exists a single-round randomized protocol for (Many vs One)-Set Disjointness(n, m) with $o(mn)$ bits of communication and error probability $O(m^{-c})$, then with constant probability one can recover mn "random bits" from $o(mn)$ bits which is a contradiction.

Suppose that Alice has a collection of m uniformly and independently random subsets of U (in each of her subsets the probability that $e \in \mathsf{U}$ is in the subset is $1/2$). Lets assume that there exists a *single round* protocol $\mathbf{I}$ for (Many vs One)-Set Disjointness(n, m) with error probability $O(m^{-c})$ using $o(mn)$ bits of communication. Let **algExistsDisj** be Bob's algorithm in protocol $\mathbf{I}$. Then we show that one can recover mn random bits with constant probability using **algExistsDisj** subroutine and the message s sent by the first party in protocol $\mathbf{I}$. The **algRecoverBit** which is shown in Figure 3.1, is the algorithm to recover random bits using protocol $\mathbf{I}$ and **algExistsDisj**.

To this end, Bob gets the message s communicated by protocol $\mathbf{I}$ from Alice and considers all subsets of size $c_1 \log m$ and $c_1 \log m + 1$ of U. Note that s is communicated only once and thus the same s is used for all queries that Bob makes. Then at each step Bob picks a random subset r_b of size $c_1 \log m$ of U and solve the (Many vs One)-Set Disjointness problem with input $(\mathcal{F}_A, \mathsf{r}_b)$ by running **algExistsDisj**(s, r_b). Next we show that if r_b is disjoint from a set in $\mathcal{F}_A$, then with high probability there is *exactly* one set in $\mathcal{F}_A$ which is disjoint from r_b (see Lemma 3.3). Thus once Bob finds out that his query, r_b, is disjoint from a set in $\mathcal{F}_A$, he can query all sets $\mathsf{r}_b^+ \in \{\mathsf{r}_b \cup e | e \in \mathsf{U} \setminus \mathsf{r}_b\}$ and recover the set (or union of sets) in $\mathcal{F}_A$ that is disjoint from r_b. By a simple *pruning step* we can detect the ones that are union of more than one set in $\mathcal{F}_A$ and only keep the sets in $\mathcal{F}_A$.

In Lemma 3.5, we show that the number of queries that Bob is required to make to recover $\mathcal{F}_A$ is $O(m^c)$ where c is a constant.

Lemma 3.3. *Let* r_b *be a random subset of* U *of size* $c \log m$ *and let* $\mathcal{F}_A$ *be a collection of* m *random subsets of* U*. The probability that there exists exactly one set in* $\mathcal{F}_A$ *that is disjoint from* r_b *is at least* $\frac{1}{m^{c+1}}$*.*

algRecoverBit(U, s):

$\mathcal{F}_a \leftarrow \emptyset$
for $i = 1$ to $m^c \log m$ **do**
  Let r_b be a random subset of U of size $c_1 \log m$
  if **algExistsDisj**$(s, \mathsf{r}_b) =$ `true`
  `// Discovering the set (or union of sets) in` $\mathcal{F}_A$ `disjoint from` r_b
    $\mathsf{r} \leftarrow \emptyset$
    for $e \in \mathsf{U} \setminus \mathsf{r}_b$
      if **algExistsDisj**$(\mathsf{r}_b \cup e, s) =$ `false`
        $\mathsf{r} \leftarrow \mathsf{r} \cup e$
    if $\exists \mathsf{r}' \in \mathcal{F}_a$ s.t. $\mathsf{r} \subset \mathsf{r}'$ `// Pruning step`
      $\mathcal{F}_a \leftarrow \mathcal{F}_a \setminus \{\mathsf{r}'\}$, $\mathcal{F}_a \leftarrow \mathcal{F}_a \cup \{\mathsf{r}\}$
    else if $\nexists \mathsf{r}' \in \mathcal{F}_a$ s.t. $\mathsf{r}' \subset \mathsf{r}$
      $\mathcal{F}_a \leftarrow \mathcal{F}_a \cup \{\mathsf{r}\}$
return $\mathcal{F}_a$

Figure 3.1: **algRecoverBit** uses a protocol of (Many vs One)-Set Disjointness(m, n) to recover Alice's sets, $\mathcal{F}_A$ in Bob's side.

Proof: The probability that r_b is disjoint from exactly one set in $\mathcal{F}_A$ is

$$\begin{aligned}&\mathtt{Pr}(\mathsf{r}_b \text{ is disjoint from at least } \textbf{one} \text{ set in } \mathcal{F}_A)\\ -&\mathtt{Pr}(\mathsf{r}_b \text{ is disjoint from at least } \textbf{two} \text{ sets in } \mathcal{F}_A) \geq (\frac{1}{2})^{c\log m} - \binom{m}{2}(\frac{1}{2})^{2c\log m} \geq \frac{1}{m^{c+1}}.\end{aligned}$$

The above inequality comes from the following inequalities.

$$\mathtt{Pr}(\mathsf{r}_b \text{ is disjoint from at least } \textbf{one} \text{ set in } \mathcal{F}_A) \geq \mathtt{Pr}(\mathsf{r}_b \text{ is disjoint from } \mathsf{r}) = (\frac{1}{2})^{c\log m},$$

where r is an arbitrary set in $\mathcal{F}_A$. For each set $\mathsf{r} \in \mathcal{F}_A$, since any element is contained in r with probability $\frac{1}{2}$, the probability that r is disjoint from r_b is $(1/2)^{c\log m}$. Moreover since there exist $\binom{m}{2}$ pairs of sets in $\mathcal{F}_A$, and for each pair of sets $\mathsf{r}_1, \mathsf{r}_2 \in \mathcal{F}_A$, the probability that r_1 and r_2 are disjoint from r_b is m^{-2c},

$$\mathtt{Pr}(\mathsf{r}_b \text{ is disjoint from at least } \textbf{two} \text{ sets in } \mathcal{F}_A) \leq m^2 \times \frac{1}{m^{2c}} \leq \frac{1}{m^{2c-2}}.$$ ■

A family of sets $\mathcal{M}$ is called *intersecting* iff for any sets $A, B \in \mathcal{M}$ either both $A \setminus B$ and $B \setminus A$ are non-empty or both $A \setminus B$ and $B \setminus A$ are empty; in other words, there exists no $A, B \in \mathcal{M}$ such that $A \subseteq B$. Let $\mathcal{F}_A$ be a collection of subsets of U. We show that with high probability after testing $O(m^c)$ queries for sufficiently large constant c, the **algRecoverBit** algorithm recovers $\mathcal{F}_A$ completely if $\mathcal{F}_A$ is intersecting. First we show that with high probability the collection $\mathcal{F}_A$ is intersecting.

Observation 3.4. *Let $\mathcal{F}_A$ be a collection of m uniformly random subsets of U where $|\mathsf{U}| \geq c \log m$. With probability at least $1 - m^{-c/4+2}$, $\mathcal{F}_A$ is an intersecting family.*

Proof: The probability that $\mathsf{r}_1 \subseteq \mathsf{r}_2$ is $(\frac{3}{4})^n$ and there are at most $m(m-1)$ pairs of sets in $\mathcal{F}_A$. Thus with probability at least $1 - m^2(\frac{3}{4})^n \geq 1 - 1/m^{\frac{c}{4}-2}$, $\mathcal{F}_A$ is intersecting. ■

By Observation 3.4 and only considering the case that $\mathcal{F}_A$ is intersecting, we have the following lemma.

Lemma 3.5. *Let $\mathcal{F}_A$ be a collection of m uniformly random subsets of U and suppose that $|\mathsf{U}| \geq c \log m$. After testing at most m^c queries, with probability at least $(1-\frac{1}{m})p^{m^c}$, $\mathcal{F}_A$ is fully recovered, where p is the success rate of protocol $\mathbf{I}$ for the* (Many vs One)-Set Disjointness *problem.*

Proof: By Lemma 3.3, for each $\mathsf{r}_b \subset \mathsf{U}$ of size $c_1 \log m$ the probability that r_b is disjoint from exactly one set in a random collection of sets $\mathcal{F}_A$ is at least $1/m^{c_1+1}$. Given r_b is disjoint from exactly one set in $\mathcal{F}_A$, due to symmetry of the problem, the chance that r_b is disjoint from a specific set $\mathsf{r} \in \mathcal{F}_A$ is at least $\frac{1}{m^{c_1+2}}$. After $\alpha m^{c_1+2} \log m$ queries where α is a large enough constant, for any $\mathsf{r} \in \mathcal{F}_A$,

$$\Pr(\nexists \text{ query } \mathsf{r}_b \text{ that is only disjoint from } \mathsf{r}) \leq (1 - \frac{1}{m^{c_1+2}})^{\alpha m^{c_1+2} \log m} \leq e^{-\alpha \log m} = \frac{1}{m^\alpha}$$

Thus after trying $\alpha m^{c_1+2} \log m$ queries, with probability at least $(1 - \frac{1}{2m^{\alpha-1}}) \geq (1 - \frac{1}{m})$, for each $\mathsf{r} \in \mathcal{F}_A$ we have at least one query that is only disjoint from r (and not any other sets in $\mathcal{F}_A \setminus \mathsf{r}$).

Once we have a query subset r_b which is only disjoint from a single set $\mathsf{r} \in \mathcal{F}_A$, we can ask $n - c \log m$ queries of size $c_1 \log m + 1$ and recover r. Note that if r_b is disjoint from more than one sets in $\mathcal{F}_A$ simultaneously, the process (asking $n - c \log m$ queries of size $c_1 \log m + 1$) will end up in recovering the union of those sets. Since $\mathcal{F}_A$ is an intersecting family with high probability (Observation 3.4), by *pruning step* in the **algRecoverBit** algorithm we are guaranteed that at the end of the algorithm, what we returned is exactly $\mathcal{F}_A$. Moreover the total number of queries the algorithm makes is at most $n \times (\alpha m^{c_1+2} \log m) \leq \alpha m^{c_1+3} \log m \leq m^c$ for $c \geq c_1 + 4$.

Thus after testing m^c queries, $\mathcal{F}_A$ will be recovered with probability at least $(1 - \frac{1}{m})p^{m^c}$ where p is the success probability of the protocol $\mathbf{I}$ for (Many vs One)-Set Disjointness(m, n). ∎

Corollary 3.6. *Let $\boldsymbol{I}$ be a protocol for* (Many vs One)-Set Disjointness(m, n) *with error probability $O(m^{-c})$ and s bits of communication such that $n \geq c \log m$ for large enough c. Then* **algRecoverBit** *recovers $\mathcal{F}_A$ with constant probability success rate using s bits of communication.*

Since **algRecoverBit** recovers mn random bits with constant probability of success (by Corollary 3.6), the size of message sent by Alice, should be $\Omega(mn)$. This proves Theorem 3.2.

Proof of Theorem 3.1: As we showed earlier, the communication complexity of (Many vs One)-Set Disjointness(m, n) is a lower bound for the communication complexity of Set Cover$(\mathsf{U}, \mathcal{F})$ where $|\mathsf{U}| = n$ and $|\mathcal{F}| = m$. Theorem 3.2 showed that any protocol for (Many vs One)-Set Disjointness$(n, \mathcal{F}_A)$ with error probability less than $O(m^{-c})$ requires $\Omega(mn)$ bits of communication. Thus any single-round randomized protocol for Set Cover with error probability $O(m^{-c})$ requires $\Omega(mn)$ bits of communication. ∎

Since any p-pass streaming α-approximation algorithm for problem P that uses $O(s)$ memory space, is a p-round two-party α-approximation protocol for problem P using $O(sp)$ bits of communication [GM08], and by Theorem 3.1, we have the following lower bound for Set Cover problem in the streaming model.

Theorem 3.7. *Any single-pass randomized streaming algorithm of* Set Cover$(\mathsf{U}, \mathcal{F})$ *that computes a $(3/2)$-approximate solution with probability $\Omega(1 - m^{-c})$ requires $\Omega(mn)$ memory space.*

Acknowledgment. The work was supported by NSF, Simons Foundation and MADALGO — Center for Massive Data Algorithmics — a Center of the Danish National Research Foundation.

References

[AES10] B. Aronov, E. Ezra, and M. Sharir. Small-size ε-nets for axis-parallel rectangles and boxes. *SIAM Journal on Computing*, 39(7):3248–3282, 2010.

[AMS06] N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algo.*, 2(2):153–177, 2006.

[AP14] P. K. Agarwal and J. Pan. Near-linear algorithms for geometric hitting sets and set covers. In *Proc. 30th Annu. Sympos. Comput. Geom.* (SoCG), pages 271–279, 2014.

[BJKS04] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Sys. Sci.*, 68(4):702–732, 2004.

[CKT10] F. Chierichetti, R. Kumar, and A. Tomkins. Max-cover in map-reduce. In *Proc. 19th Int. Conf. World Wide Web* (WWW), pages 231–240, 2010.

[CKW10] G. Cormode, H. J. Karloff, and A. Wirth. Set cover algorithms for very large datasets. In *Proc. 19th ACM Conf. Info. Know. Manag.* (CIKM), pages 479–488, 2010.

[CS89] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[CV07] K. L. Clarkson and K. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete Comput. Geom.*, 37(1):43–58, 2007.

[CW16] A. Chakrabarti and T. Wirth. Incidence geometries and the pass complexity of semi-streaming set cover. In *Proc. 27th ACM-SIAM Sympos. Discrete Algs.* (SODA), 2016. To appear. Preprint arXiv:1507.04645.

[DIMV14] E. D. Demaine, P. Indyk, S. Mahabadi, and A. Vakilian. On streaming and communication complexity of the set cover problem. In *Proc. 28th Int. Symp. Dist. Comp.* (DISC), volume 8784 of *Lect. Notes in Comp. Sci.*, pages 484–498, 2014.

[EHR12] A. Ene, S. Har-Peled, and B. Raichel. Geometric packing under non-uniform constraints. In *Proc. 28th Annu. Sympos. Comput. Geom.* (SoCG), pages 11–20, 2012.

[ER14] Y. Emek and A. Rosén. Semi-streaming set cover. In *Proc. 41st Int. Colloq. Automata Lang. Prog.* (ICALP), volume 8572 of *Lect. Notes in Comp. Sci.*, pages 453–464, 2014.

[FG88] T. Feder and D. H. Greene. Optimal algorithms for approximate clustering. In *Proc. 20th Annu. ACM Sympos. Theory Comput.* (STOC), pages 434–444, 1988.

[FPT81] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Inform. Process. Lett.*, 12(3):133–137, 1981.

[GM07] S. Guha and A. McGregor. Lower bounds for quantile estimation in random-order and multi-pass streaming. In *Proc. 34th Int. Colloq. Automata Lang. Prog.* (ICALP), volume 4596 of *Lect. Notes in Comp. Sci.*, pages 704–715, 2007.

[GM08] S. Guha and A. McGregor. Tight lower bounds for multi-pass stream computation via pass elimination. In *Proc. 35th Int. Colloq. Automata Lang. Prog.* (ICALP), volume 5125 of *Lect. Notes in Comp. Sci.*, pages 760–772, 2008.

[GO13] V. Guruswami and K. Onak. Superlinear lower bounds for multipass graph processing. In *Proc. 28th Conf. Comput. Complexity* (CCC), pages 287–298, 2013.

[GW97] T. Grossman and A. Wool. Computational experience with approximation algorithms for the set covering problem. *Euro. J. Oper. Res.*, 101(1):81–92, 1997.

[HQ15] S. Har-Peled and K. Quanrud. Approximation algorithms for polynomial-expansion and low-density graphs. In *Proc. 23nd Annu. European Sympos. Algorithms* (ESA), volume 9294 of *Lect. Notes in Comp. Sci.*, pages 717–728, 2015.

[HS11] S. Har-Peled and M. Sharir. Relative (p, ε)-approximations in geometry. *Discrete Comput. Geom.*, 45(3):462–496, 2011.

[KMVV13] R. Kumar, B. Moseley, S. Vassilvitskii, and A. Vattani. Fast greedy algorithms in MapReduce and streaming. In *Proc. 25th ACM Sympos. Parallel Alg. Arch.* (SPAA), pages 1–10, 2013.

[KS92] B. Kalyanasundaram and G. Schintger. The probabilistic communication complexity of set intersection. *SIAM J. Discrete Math.*, 5(4):545–557, 1992.

[MRR14] N. H. Mustafa, R. Raman, and S. Ray. Settling the apx-hardness status for geometric set cover. In *Proc. 55th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 541–550, 2014.

[Nis02] N. Nisan. The communication complexity of approximate set packing and covering. In *Proc. 29th Int. Colloq. Automata Lang. Prog.* (ICALP), pages 868–875, 2002.

[Raz92] A. A. Razborov. On the distributional complexity of disjointness. *Theo. Comp. Sci.*, 106(2):385–390, 1992.

[RS97] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proc. 29th Annu. ACM Sympos. Theory Comput.* (STOC), pages 475–484, 1997.

[SG09] B. Saha and L. Getoor. On maximum coverage in the streaming model & application to multi-topic blog-watch. In *Proc. SIAM Int. Conf. Data Mining* (SDM), pages 697–708, 2009.

A. Geometric Set Cover

In this section, we consider the streaming Set Cover problem in the geometric settings. We present an algorithm for the case where the elements are a set of n points in the plane $\mathbb{R}^2$ and the m sets are either all disks, all axis-parallel rectangles, or all α-fat triangles (which for simplicity we call shapes) given in a data stream. As before, the goal is to find the minimum size cover of points from the given sets. We call this problem the Points-Shapes Set Cover problem.

Note that, the description of each shape requires $O(1)$ space and thus the Points-Shapes Set Cover problem is trivial to be solved in $O(m+n)$ space. In this setting the goal is to design an algorithm whose space is sub-linear in $O(m+n)$. Here we show that almost the same algorithm as **iterSetCover** (with slight modifications) uses $\widetilde{O}(n)$ space to find a $O(\rho/\delta)$-approximate solution of the Points-Shapes Set Cover problem in $O(1/\delta)$ passes.

A.1. Preliminaries

A triangle $\triangle$ is called α-*fat* (or simply fat) if the ratio between its longest edge and its height on this edge is bounded by a constant $\alpha > 1$ (there are several equivalent definitions of α-fat triangles).

Definition A.1. Let $(\mathsf{U}, \mathcal{F})$ be a set system such that U is a set of points and $\mathcal{F}$ is a collection of shapes, in the plane $\mathbb{R}^2$. The *canonical representation* of $(\mathsf{U}, \mathcal{F})$ is a collection $\mathcal{F}'$ of regions such that the following conditions hold. First, each $\mathsf{r}' \in \mathcal{F}'$ has $O(1)$ description. Second, for each $\mathsf{r}' \in \mathcal{F}'$, there exists $\mathsf{r} \in \mathcal{F}$ such that $\mathsf{r}' \cap \mathsf{U} \subseteq \mathsf{r} \cap \mathsf{U}$. Finally, for each $\mathsf{r} \in \mathcal{F}$, there exists c_1 sets $\mathsf{r}'_1, \cdots, \mathsf{r}'_{c_1} \in \mathcal{F}'$ such that $\mathsf{r} \cap \mathsf{U} = (\mathsf{r}'_1 \cup \cdots \cup \mathsf{r}'_{c_1}) \cap \mathsf{U}$ for some constant c_1.

The following two results are from [EHR12] which are the formalization of the ideas in [AES10].

Lemma A.2. (Lemma 4.18 in [EHR12]) *Given a set of points U in the plane $\mathbb{R}^2$ and a parameter w, one can compute a set $\mathcal{F}'_{\mathtt{total}}$ of $O(|\mathsf{U}|w^2 \log |\mathsf{U}|)$ axis-parallel rectangles with the following property. For an arbitrary axis-parallel rectangle r that contains at most w points of U, there exist two axis-parallel rectangles $\mathsf{r}'_1, \mathsf{r}'_2 \in \mathcal{F}'_{\mathtt{total}}$ whose union has the same intersection with U as r, i.e., $\mathsf{r} \cap \mathsf{U} = (\mathsf{r}'_1 \cup \mathsf{r}'_2) \cap \mathsf{U}$.*

Lemma A.3. (Theorem 5.6 in [EHR12]) *Given a set of points U in the plane $\mathbb{R}^2$, a parameter w and a constant α, one can compute a set $\mathcal{F}'_{\mathtt{total}}$ of $O(|\mathsf{U}|w^3 \log^2 |\mathsf{U}|)$ regions each having $O(1)$ description with the following property. For an arbitrary α-fat triangle r that contains at most w points of U, there exist nine regions from $\mathcal{F}'_{\mathtt{total}}$ whose union has the same intersection with U as r.*

Using the above lemmas we get the following lemma.

Lemma A.4. *Let U be a set of points in $\mathbb{R}^2$ and let $\mathcal{F}$ be a set of shapes (discs, axis-parallel rectangles or fat triangles), such that each set in $\mathcal{F}$ contains at most w points of U. Then, in a single pass over the stream of sets $\mathcal{F}$, one can compute the canonical representation $\mathcal{F}'$ of $(\mathsf{U}, \mathcal{F})$. Moreover, the size of the canonical representation is $O(|\mathsf{U}|w^3 \log^2 |\mathsf{U}|)$ and the space requirement of the algorithm is $\widetilde{O}(|\mathcal{F}'|) = \widetilde{O}(|\mathsf{U}|w^3 \log^2 |\mathsf{U}|)$.*

Proof: For the case of axis-parallel rectangles and fat triangles, first we can uses Lemma A.2 and Lemma A.3 to get the set $\mathcal{F}'_{\mathtt{total}}$ offline. By cheching the proofs of the lemmas, it is easy to check that this would require $\widetilde{O}(\mathcal{F}'_{\mathtt{total}}) = \widetilde{O}(|\mathsf{U}|w^3 \log^2 |\mathsf{U}|)$ memory space. Then going one pass over the stream of sets $\mathcal{F}$, we can find the canonical representation $\mathcal{F}'$ by picking all the sets $S' \in \mathcal{F}'_{\mathtt{total}}$ such that $S' \cap \mathsf{U} \subseteq S \cap \mathsf{U}$ for some $S \in \mathcal{F}$. For discs however, we just make one pass over the sets $\mathcal{F}$ and keep a maximal subset $\mathcal{F}' \subseteq \mathcal{F}$ such that for each pair of sets $S'_1, S'_2 \in \mathcal{F}'$ their projection on U are different, i.e., $S'_1 \cap \mathsf{U} \neq S'_2 \cap \mathsf{U}$. By a standard technique of Clarkson and Shor [CS89], it can be proved that the size of the canonical representation, i.e., $|S'|$, is bounded by $O(|\mathsf{U}|w^2)$. Note that this is just counting the number of discs that contain at most w points, namely the at most w-level discs. ■

A.2. Algorithm

The outline of the Points-Shapes-Set-Cover algorithm (shown in Figure A.1) is very similar to the **iterSetCover** algorithm presented earlier in Section 2.

algGeomSC$(\mathsf{U}, \mathcal{F}, \delta)$:

for $k \in \{2^i \mid 0 \le i \le \log n\}$ **do in parallel**: // $n = |\mathsf{U}|$
  Let $\mathsf{L} \leftarrow \mathsf{U}$ and $\mathtt{sol} \leftarrow \emptyset$
  Repeat $1/\delta$ times:
    for $\mathsf{r} \in \mathcal{F}$ **do** // Pass
      if $|\mathsf{r} \cap \mathsf{L}| \ge |\mathsf{U}|/k$ **then**
        $\mathtt{sol} \leftarrow \mathtt{sol} \cup \{\mathsf{r}\}$
        $\mathsf{L} \leftarrow \mathsf{L} \setminus \mathsf{r}$
    $\mathsf{S} \leftarrow$ sample of L of size $c\rho k(n/k)^\delta \log m \log n$
    $\mathcal{F}_\mathsf{S} \leftarrow$**compCanonicalRep**$(\mathsf{S}, \mathcal{F}, |\mathsf{S}|/k)$ // Pass
    $\mathtt{sol}_\mathsf{S} \leftarrow$ **algOfflineSC**$(\mathsf{S}, \mathcal{F}_\mathsf{S})$
    for $\mathsf{r} \in \mathcal{F}$ **do** // Pass
      if $\exists \mathsf{r}' \in \mathtt{sol}_\mathsf{S}$ s.t. $\mathsf{r}' \cap \mathsf{S} \subseteq \mathsf{r} \cap \mathsf{S}$ **then**
        $\mathtt{sol} \leftarrow \mathtt{sol} \cup \{\mathsf{r}\}$ and $\mathtt{sol}_\mathsf{S} \leftarrow \mathtt{sol}_\mathsf{S} \setminus \{\mathsf{r}'\}$
        $\mathsf{L} \leftarrow \mathsf{L} \setminus \mathsf{r}$
  for $\mathsf{r} \in \mathcal{F}$ **do** // Final Pass
    if $\mathsf{r} \cap \mathsf{L} \neq \emptyset$ **then**
      $\mathtt{sol} \leftarrow \mathtt{sol} \cup \{\mathsf{r}\}$
      $\mathsf{L} \leftarrow \mathsf{L} \setminus \mathsf{r}$
return smallest $\mathtt{sol}$ computed in parallel

Figure A.1: A streaming algorithm for Points-Shapes Set Cover problem.

In the first pass, the algorithm picks all the sets that cover a large number of yet-uncovered elements. Next, we sample S. Since we have removed all the ranges that have large size, in the first pass, the size of the remaining ranges *restricted to* the sample S is small. Therefore by Lemma A.4, the canonical representation of $(\mathsf{S}, \mathcal{F}_\mathsf{S})$ has small size and we can afford to store it in the memory. We use Lemma A.4 to compute the canonical representation $\mathcal{F}_\mathsf{S}$ in one pass. The algorithm then uses the sets in $\mathcal{F}_\mathsf{S}$ to find a cover $\mathtt{sol}_\mathsf{S}$ for the points of S. Next, in one additional pass, the algorithm replaces each set in $\mathtt{sol}_\mathsf{S}$ by one of its supersets from $\mathcal{F}$.

Finally, note that in the algorithm of Section 2, we are assuming that the size of the optimal solution is $O(k)$. Thus it is enough to stop the iterations once the number of uncovered elements is less than k. Then we can pick an arbitrary set for each of the uncovered elements. This would add only k more sets to the solution. Using this idea, we can reduce the size of the sampled elements down to $c\rho k(n/k)^\delta \log m \log n$ which would help us in getting near-linear space in the geometric setting. Note that the final pass of the algorithm can be embedded into the previous passes but for the sake of clarity we write it separately.

A.3. Analysis

By a similar approach to what we used in Section 2 to analyze the pass count and approximation guarantee of **iterSetCover** algorithm, we can show that the number of passes of the **algGeomSC** algorithm is $3/\delta + 1$ (which can be reduced to $3/\delta$ with minor changes), and the algorithm returns an $O(\rho/\delta)$-approximate solution. Next, we analyze the space usage and the correctness of the algorithm.

Lemma A.5. *The algorithm uses $\widetilde{O}(n)$ space.*

Proof: Consider an iteration of the algorithm. The memory space used in the first pass of each iteration is $\widetilde{O}(n)$. The size of S is $c\rho k(n/k)^{\delta}\log m\log n$ and after the first pass the size of each set is at most $|\mathsf{U}|/k$. Thus using Chernoff bound for each set $\mathsf{r}\in\mathcal{F}\setminus\mathtt{sol}$,

$$\mathbf{Pr}\left[|\mathsf{r}\cap\mathsf{S}|>(1+2)\frac{|\mathsf{U}|}{k}\times\frac{|\mathsf{S}|}{|\mathsf{U}|}\right]\leq\exp\left(-\frac{4|\mathsf{S}|}{3k}\right)\leq(1/m)^{c+1}.$$

Thus, with probability at least $1-m^{-c}$ (by the union bound), all the sets that are not picked in the first pass, cover at most $3|\mathsf{S}|/k=c\rho(n/k)^{\delta}\log m\log n$ elements of S. Therefore, we can use Lemma A.4 to show that the number of sets in the canonical representation of $(\mathsf{S},\mathcal{F}_{\mathsf{S}})$ is at most

$$|\mathcal{F}_{\mathsf{S}}|=O\left(|\mathsf{S}|\left(\frac{3|\mathsf{S}|}{k}\right)^{3}\log^{2}|\mathsf{S}|\right)=O\Big(\rho^{4}k(n/k)^{4\delta}\log^{4}m\log^{6}n\Big)=O(\rho^{4}n\log^{4}m\log^{6}n),$$

as long as $\delta\leq 1/4$. To store each set in a canonical representation of $(\mathsf{S},\mathcal{F})$ only constant space is required. Moreover, by Lemma A.4, the space requirement of the second pass is $\widetilde{O}(|\mathcal{F}_{\mathsf{S}}|)=\widetilde{O}(n)$. Therefore, the total required space is $\widetilde{O}(n)$ and the lemma follows. ∎

Theorem A.6. *Given a set system defined over a set U of n points in the plane, and a set of m ranges $\mathcal{F}$ (which are either all disks, axis-parallel rectangles, or fat triangles). Let ρ be the quality of approximation to the offline set-cover solver we have, and let $0<\delta<1/4$ be an arbitrary parameter.*

The algorithm **algGeomSC**, *depicted in Figure A.1, with high probability, returns an $O(\rho/\delta)$-approximate solution of the optimal set cover solution for the instance $(\mathsf{U},\mathcal{F})$. This algorithm uses $\widetilde{O}(n)$ space, and performs $3/\delta+1$ passes over the data.*

Proof: As before consider the run of the algorithm in which $|\mathrm{OPT}|\leq k<2|\mathrm{OPT}|$. Let V be the set of uncovered elements L at the beginning of the iteration and note that the total number of sets that is picked during the iteration is at most $(1+c_1\rho)k$ where c_1 is the constant defined in Definition A.1. Let $\mathcal{G}$ denote all possible such covers, that is $\mathcal{G}=\left\{\mathcal{F}'\subseteq\mathcal{F}\;\middle|\;|\mathcal{F}'|\leq(1+c_1\rho)k\right\}$. Let $\mathcal{H}$ be the collection that contains all possible set of uncovered elements at the end of the iteration, defined as $\mathcal{H}=\left\{\mathsf{V}\setminus\bigcup_{\mathsf{r}\in\mathcal{C}}\mathsf{r}\;\middle|\;\mathcal{C}\in\mathcal{G}\right\}$. Set $p=(k/n)^{\delta}$, $\varepsilon=1/2$ and $q=m^{-c}$. Since for large enough c, $\frac{c'}{\varepsilon^2 p}(\log|\mathcal{H}|\log\frac{1}{p}+\log\frac{1}{q})\leq c\rho k(n/k)^{\delta}\log m\log n=|\mathsf{S}|$ with probability at least $1-m^{-c}$, by Lemma 2.5, the set of sampled elements S is a relative (p,ε)-approximation sample of $(\mathsf{V},\mathcal{H})$.

Let $\mathcal{C}\subseteq\mathcal{F}$ be the collection of sets picked in the third pass of the algorithm that covers all elements in S. By Lemma A.4, $|\mathcal{C}|\leq c_1\rho k$ for some constant c_1. Since with high probability S is a relative (p,ε)-approximation sample of $(\mathsf{V},\mathcal{H})$, the number of uncovered elements of V (or L) after adding $\mathcal{C}$ to $\mathtt{sol}$ is at most $\varepsilon p|\mathsf{V}|\leq|\mathsf{U}|(k/n)^{\delta}$. Thus with probability at least $(1-m^{-c})$, in each iteration and by adding $O(\rho k)$ sets, the number of uncovered elements reduces by a factor of $(n/k)^{\delta}$. Therefore, after $1/\delta$ iterations the algorithm picks $O(\rho k/\delta)$ sets and with high probability the number of uncovered elements is at most $n(k/n)^{\delta/\delta}=k$. Thus, in the final pass the algorithm only adds k sets to the solution $\mathtt{sol}$, and hence the approximation factor of the algorithm is $O(\rho/\delta)$. ∎

Remark A.7. The result of Theorem A.6 is similar to the result of Agarwal and Pan [AP14] – except that their algorithm performs $O(\log n)$ iterations over the data, while the algorithm of Theorem A.6 performs only a constant number of iterations. In particular, one can use the algorithm of Agarwal and Pan [AP14] as the offline solver.

B. Lower bound for multipass algorithms

In this section we give lower bound on the memory space of multipass streaming algorithms of the Set Cover problem. Our main result is $\Omega(mn^{\delta})$ space for streaming algorithms that return an optimal solution of the Set Cover problem in $O(1/\delta)$ passes. Our approach is to reduce the communication Intersection Set Chasing(n,p) problem introduced by Guruswami and Onak [GO13] to the communication Set Cover problem.

Consider a communication problem P with n players $P_1,\cdots,P_n$. The problem P is a (n,r)-communication problem if players communicate in r rounds and in each round they speak in order $P_1,\cdots,P_n$. At the end of the rth round P_n should return the solution. Moreover we assume private randomness and public messages. In what follows we define the communication Set Chasing and Intersection Set Chasing problems.

Definition B.1 (Communication Set Chasing(n,p) Problem). Set Chasing(n,p) is a $(p,p-1)$ communication problem in which the player i has a function $f_i:[n]\to 2^{[n]}$ and the goal is to compute $\vec{f}_1(\vec{f}_2(\cdots\vec{f}_p(\{1\})\cdots))$ where $\vec{f}_i(S)=\bigcup_{s\in S}f_i(s)$. Figure B.1(a) shows an instance of the communication Set Chasing $(4,3)$.

Definition B.2 (Communication Intersection Set Chasing(n,p) Problem). The communication Intersection Set Chasing(n,p) is a $(2p,p-1)$ communication problem in which the first p players have an instance of the Set Chasing(n,p) problem and the other p players have another instance of the Set Chasing(n,p) problem. The output of the Intersection Set Chasing(n,p) is 1 if the solutions of the two instances of the Set Chasing(n,p) intersect and 0 otherwise. Figure B.1(b) shows an instance of the Intersection Set Chasing $(4,3)$. The function f_i of each player P_i is specified by a set of directed edges form a copy of vertices labeled $\{1,\cdots,n\}$ to another copy of vertices labeled $\{1,\cdots,n\}$.

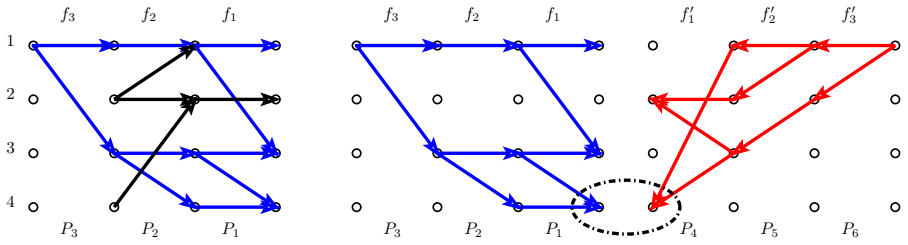

Figure B.1: (a) shows an example of the communication Set Chasing$(4,3)$ and (b) is an instance of the communication Intersection Set Chasing$(4,3)$.

The communication Set Chasing problem is a generalization of the well-known communication Pointer Chasing problem in which player i has a function $f_i:[n]\to[n]$ and the goal is to compute $f_1(f_2(\cdots f_p(1)\cdots))$.

[GO13] showed that any randomized protocol that solves Intersection Set Chasing(n,p) with error probability less than $1/10$, requires $\Omega(\frac{n^{1+1/(2p)}}{p^{16}\log^{3/2}n})$ bits of communication where n is sufficiently large and $p\le\frac{\log n}{\log\log n}$. In Theorem B.4, we reduce the communication Intersection Set Chasing problem to the communication Set Cover problem and then give the first superlinear memory lower bound for the streaming Set Cover problem.

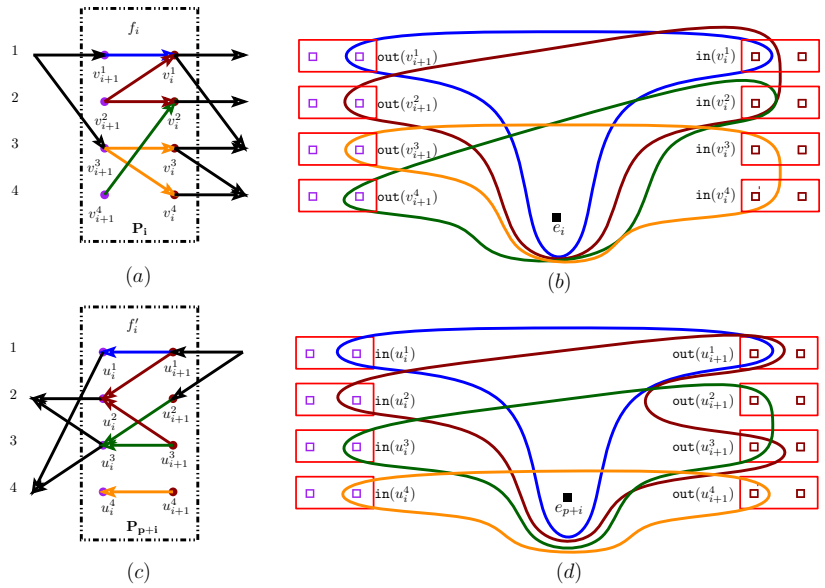

Figure B.2: The gadgets used in the reduction of the communication Intersection Set Chasing problem to the communication Set Cover problem. (a)-(b) shows the construction of the gadget for players 1 to p and (c)-(d) shows the construction of the gadget for players $p+1$ to $2p$.

Definition B.3 (Communication Set Cover($\mathsf{U}, \mathcal{F}, p$) Problem). The communication Set Cover(n, p) is a $(p, p-1)$ communication problem in which a collection of elements U is given to all players and each player i has a collection of subsets of U, $\mathcal{F}_i$. The goal is to solve Set Cover($\mathsf{U}, \mathcal{F}_1 \cup \cdots \cup \mathcal{F}_p$) using the minimum number of communication bits.

Theorem B.4. *Any streaming algorithm that solves* Set Cover($\mathsf{U}, \mathcal{F}$) *optimally with constant probability of error in* $(1/2\delta - 1)$ *passes requires* $\widetilde{\Omega}(mn^{\delta})$ *memory space where* $\delta \geq \frac{\log\log n}{\log n}$.

Consider an instance ISC of the communication Intersection Set Chasing(n, p). We construct an instance of the communication Set Cover($\mathsf{U}, \mathcal{F}, 2p$) problem such that solving Set Cover($\mathsf{U}, \mathcal{F}$) optimally determines whether the output of ISC is 1 or not.

The instance ISC consists of $2p$ players. Each player $1, \cdots, p$ has a function $f_i : [n] \to 2^{[n]}$ and each player $p+1, \cdots, 2p$ has a function $f'_i : [n] \to 2^{[n]}$ (see Figure B.1). In ISC, each function f_i is shown by a set of vertices $v_i^1, \cdots, v_i^n$ and $v_{i+1}^1, \cdots, v_{i+1}^n$ such that there is a directed edge from v_{i+1}^j to v_i^ℓ iff $\ell \in f_i(j)$. Similarly, each function f'_i is denoted by a set of vertices $u_i^1, \cdots, u_i^n$ and $u_{i+1}^1, \cdots, u_{i+1}^n$ such that there is a directed edge from u_{i+1}^j to u_i^ℓ iff $\ell \in f'_i(j)$ (see Figure B.2(a) and Figure B.2(c)).

In the corresponding communication Set Cover instance of ISC, we add two elements $\texttt{in}(v_i^j)$ and $\texttt{out}(v_i^j)$ per each vertex v_i^j where $i \leq p+1, j \leq n$. We also add two elements $\texttt{in}(u_i^j)$ and $\texttt{out}(u_i^j)$ per each vertex u_i^j where $i \leq p+1, j \leq n$. In addition to these elements, for each player i, we add an element e_i (see Figure B.2(b) and Figure B.2(d)).

Next, we define a collection of sets in the corresponding Set Cover instance of ISC. For each player P_i, where $1 \leq i \leq p$, we add a single set S_i^j containing $\texttt{out}(v_{i+1}^j)$ and $\texttt{in}(v_i^\ell)$ for all out-going edges (v_{i+1}^j, v_i^ℓ). Moreover, all S_i^j sets contain the element e_i. Next, for each vertex v_i^j we add a set R_i^j that contains the two corresponding elements of v_i^j, $\texttt{in}(v_i^j)$ and $\texttt{out}(v_i^j)$. In Figure B.2(b), the red rectangles denote R-type sets and the curves denote S-type sets for the first half of the players.

Similarly to the sets corresponding to players 1 to p, for each player P_{p+i} where $1 \le i \le p$, we add a set S^j_{p+i} containing $\texttt{in}(u^j_i)$ and $\texttt{out}(u^\ell_{i+1})$ for all in-coming edges (u^ℓ_{i+1}, u^j_i) of u^j_i (denoting $f'^{-1}_i(j)$). The set S^j_{p+i} contains the element e_{p+i} too. Next, for each vertex u^j_i we add a set T^j_{p+i} that contains the two corresponding elements of u^j_i, $\texttt{in}(u^j_i)$ and $\texttt{out}(u^j_i)$. In Figure B.2(d), the red rectangles denote T-type sets and the curves denote S-type sets for the second half of the players.

At the end, we merge v^i_1s and u^i_1s as shown in Figure B.3. As we merge the corresponding sets of v^j_1s $(R^1_1, \cdots, R^n_1)$ and the corresponding sets of u^j_1s $(T^1_1, \cdots, T^n_1)$, we call the merged sets $T^1_1, \cdots, T^n_1$.

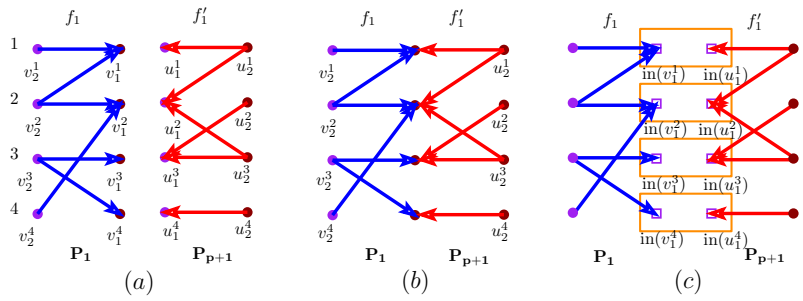

Figure B.3: In (b) two Set Chasing instances merge in their first set of vertices and (c) shows the corresponding gadgets of these merged vertices in the communication Set Cover.

The main claim is that if the solution of ISC is 1 then the size of an optimal solution of its corresponding Set Cover instance SC is $(2p+1)n+1$; otherwise, it is $(2p+1)n+2$.

Lemma B.5. *The size of any feasible solution of* SC *is at least* $(2p+1)n+1$.

Proof: For each player i $(1 \le i \le p)$, since $\texttt{out}(v^j_{i+1})$s are only covered by R^j_{i+1} and S^j_i, at least n sets are required to cover $\texttt{out}(v^1_{i+1}), \cdots, \texttt{out}(v^n_{i+1})$. Moreover for player P_p, since $\texttt{in}(v^j_{p+1})$s are only covered by R^j_{p+1} and e_p is only covered by S^1_p, all $n+1$ sets $R^1_{p+1}, \cdots, R^n_{p+1}, S^1_p$ must be selected in any feasible solution of SC.

Similarly for each player $p+i$ $(1 \le i \le p)$, since $\text{in}(u^j_i)$s are only covered by T^j_i and S^j_{p+i}, at least n sets are required to cover $\texttt{in}(u^1_i), \cdots, \texttt{in}(u^n_i)$. Moreover, considering $u^1_{p+1}, \cdots, u^n_{p+1}$, since $\texttt{in}(u^j_{p+1})$ is only covered by T^j_{p+1}, all n sets $T^1_{p+1}, \cdots, T^n_{p+1}$ must be selected in any feasible solution of SC.

All together, at least $(2p+1)n+1$ sets should be selected in any feasible solution of SC. ∎

Next we show that if the solution of ISC is 1 then the size of an optimal solution of SC is exactly $(2p+1)n+1$.

Lemma B.6. *Suppose that the solution of* ISC *is* 1. *Then the size of an optimal solution of its corresponding* Set Cover *instance is exactly* $(2p+1)n+1$.

Proof: By Lemma B.5, the size of an optimal solution of S is at least $(2p+1)n+1$. Here we prove that $(2p+1)n+1$ sets suffice when the solution of ISC is 1. Let $Q = v^1_{p+1}, v^{j_p}_p, \ldots, v^{j_2}_2, v^{j_1}_1, u^{\ell_1}_1, u^{\ell_2}_2, \ldots, u^{\ell_p}_p, u^1_{p+1}$ be a path in ISC such that $j_1 = \ell_1$ (since the solution of ISC is 1 such a path exists). The corresponding solution to Q can be constructed as follows (See Figure B.4):

- Pick S^1_p and all R^j_{p+1}s $(n+1$ sets$)$.

- For each $v_i^{j_i}$ in Q where $1 < i \leq p$, pick the set $S_{i-1}^{j_i}$ in the solution. Moreover, for each such i pick all sets R_i^j where $j \neq j_i$ ($n(p-1)$ sets).
- For $v_1^{j_1}$ (or $u_1^{\ell_1}$), pick the set $S_{p+1}^{j_1}$. Moreover, pick all sets T_1^j where $j \neq j_1$ (n sets).
- For each $u_i^{\ell_i}$ in Q where $1 < i \leq p$, pick the set $S_{p+i}^{\ell_i}$ in the solution. Moreover, for each such i pick all sets T_i^ℓ where $\ell \neq \ell_i$ ($n(p-1)$ sets).
- Pick all T_{p+1}^js (n sets).

It is straightforward to see that the solution constructed above is a feasible solution. ■

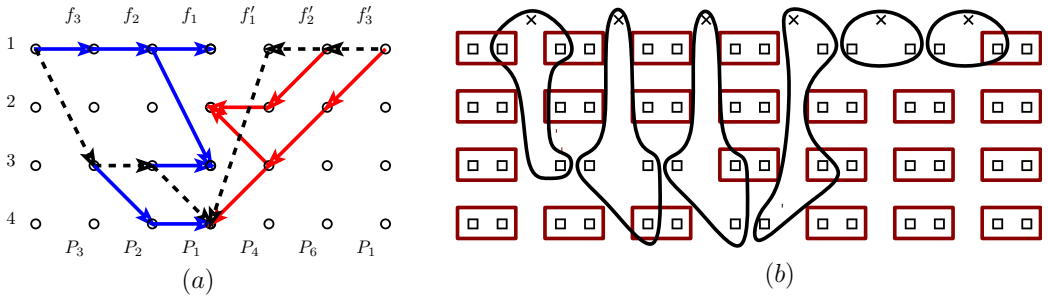

Figure B.4: In (a), path Q is shown with black dashed arcs and (b) shows the corresponding cover of path Q.

Next we show that if the size of an optimal cover of S is $(2p+1)n+1$ then the solution of ISC is 1.

Lemma B.7. *Suppose that the size of an optimal solution of the corresponding* Set Cover *instance of* ISC, SC, *is* $(2p+1)n+1$. *Then the solution of* ISC *is* 1.

Proof: As we proved earlier in Lemma B.5, any feasible solution of SC picks $R_{p+1}^1, \cdots, R_{p+1}^n, S_p^1$ and $T_{p+1}^1, \cdots, T_{p+1}^n$. Moreover, we proved that for each $1 \leq i < p$, at least n sets should be selected from $R_{i+1}^1, \cdots, R_{i+1}^n, S_i^1, \cdots, S_i^n$. Similarly, for each $1 \leq i \leq p$, at least n sets should be selected from $T_i^1, \cdots, T_i^n, S_{p+i}^1, \cdots, S_{p+1}^n$. Thus if a feasible solution of SC, OPT, is of size $(2p+1)n+1$, it has exactly n sets from each specified group.

Next we consider the first half of the players and second half of the players separately. Consider i such that $1 \leq i < p$. Let $S_i^{j_1}, \cdots, S_i^{j_k}$ be the sets picked in the optimal solution (because of e_i there should be at least one set of form S_i^j in OPT). Since each $\texttt{out}(v_{i+1}^j)$ is only covered by S_i^j and R_{i+1}^j, for all $j \notin \{j_1, \ldots, j_k\}$, R_{i+1}^j should be selected in OPT. Moreover, for all $j \in \{j_1, \cdots, j_k\}$, R_{i+1}^j should not be contained in OPT (otherwise the size of OPT would be larger than $(2p+1)n+1$). Consider $j \in \{j_1, \ldots, j_k\}$. Since R_{i+1}^j is not in OPT, there should be a set S_{i+1}^ℓ selected in OPT such that $\texttt{in}(v_{i+1}^j)$ is contained in S_{i+1}^ℓ. Thus by considering S_is in a decreasing order and using induction, if S_i^j is in OPT then v_{i+1}^j is reachable form v_{p+1}^1.

Next consider a set S_{p+i}^j that is selected in OPT ($1 \leq i \leq p$). By similar argument, T_i^j is not in OPT and there exists a set S_{p+i-1}^ℓ (or S_1^ℓ if $i = 1$) in OPT such that $\texttt{out}(u_i^j)$ is contained in S_{p+i-1}^ℓ. Let $u_{i+1}^{\ell_1}, \cdots, u_{i+1}^{\ell_k}$ be the set of vertices whose corresponding $\texttt{out}$ elements are in S_{p+i}^j. Then by induction, there exists an index r such that v_1^r is reachable from v_{p+1}^1 and u_1^r is also reachable from all $u_{i+1}^{\ell_1}, \cdots, u_{i+1}^{\ell_k}$. Moreover, the way we constructed the instance SC guarantees that all sets $S_{2p}^1, \cdots, S_{2p}^n$ contains $\texttt{out}(u_{p+1}^1)$. Hence if the size of an optimal solution of SC is $(2p+1)n+1$ then the solution of ISC is 1. ■

Corollary B.8. *The solution of* Intersection Set Chasing(n,p) *is* 1 *iff the size of optimal solution of its corresponding* Set Cover *instance (as described in this section) is* $(2p+1)n+1$.

Next we prove the main theorem of this section, Theorem B.4.

Observation B.9. *Any ℓ-pass streaming algorithm of* Set Cover*, $\mathcal{I}$, that solves the problem optimally with a probability of error* `err` *and consumes $O(s)$ memory space, solves the corresponding communication* Set Cover *problem in ℓ rounds using $O(s\ell^2)$ bits of communication with probability error* `err`.

Proof: Starting from player P_1, each player runs $\mathcal{I}$ over its input sets and once P_i is done with its input, she sends the working memory of $\mathcal{I}$ publicly to other players. Then next player starts the same routine using the state of the working memory received from the previous player. Since $\mathcal{I}$ solves the Set Cover instance optimally after ℓ passes using $O(s)$ space with probability error `err`, applying $\mathcal{I}$ as a black box we can solve P in ℓ rounds using $O(s\ell^2)$ bits of communication with probability error `err`. ■

Proof of Theorem B.4: By Observation B.9, any ℓ-round $O(s)$-space algorithm that solves the streaming Set Cover$(\mathsf{U},\mathcal{F})$ optimally can be used to solve the communication Set Cover$(\mathsf{U},\mathcal{F},p)$ problem in ℓ rounds using $O(s\ell^2)$ bits of communication. Moreover, by Corollary B.8, we can decide the solution of the communication Intersection Set Chasing(n,p) by solving its corresponding communication Set Cover problem. Note that while working with the corresponding Set Cover instance of Intersection Set Chasing(n,p), all players know the collection of elements U and each player can construct its collection of sets $\mathcal{F}_i$ using f_i (or f'_i).

However, by a result of [GO13], we know that any protocol that solves the communication Intersection Set Chasing(n,p) problem with probability of error less than $1/10$, requires $\Omega(\frac{n^{1+1/(2p)}}{p^{16}\log^{3/2}n})$ bits of communication. Since in the corresponding Set Cover instance of the communication Intersection Set Chasing(n,p), $|\mathsf{U}|=(2p+1)\times 2n+2p=O(np)$ and $|\mathcal{F}|\leq(2p+1)n+2pn=O(np)$, any $(p-1)$-pass streaming algorithm that solves the Set Cover problem optimally with a probability of error at most $1/10$, requires $\Omega(\frac{n^{1+1/(2p)}}{p^{18}\log^{3/2}n})$ bits of communication. Then using Observation B.9, since $\delta\geq\frac{\log\log n}{\log n}$, any $(\frac{1}{2\delta}-1)$-pass streaming algorithmof Set Cover that finds an optimal solution with error probability less than $1/10$, requires $\widetilde{\Omega}(|\mathcal{F}||\mathsf{U}|^{\delta})$ space. ■

C. Tight Lower Bound for Sparse Set Cover Problem in Multiple Passes

In this part we give a stronger lower bound for the instances of the streaming Set Cover problem with sparse input sets. An instance of the Set Cover problem is s-Sparse Set Cover, if for each set $\mathsf{r}\in\mathcal{F}$ we have $|\mathsf{r}|\leq s$. We can us the same reduction approach described earlier in Section B to show that any $(1/2\delta-1)$-pass streaming algorithm of s-Sparse Set Cover requires $\Omega(|\mathcal{F}|s)$ memory space if $s<|\mathsf{U}|^{\delta}$. To prove this, we need to explain more details of the approach of [GO13] on the lower bound of the communication Intersection Set Chasing problem. They first obtained a lower bound for Equal Pointer Chasing(n,p) problem in which two instances of the communication Pointer Chasing(n,p) are given and the goal is to decide whether these two instances point to a same value or not; $f_p(\cdots f_1(1)\cdots)=f'_p(\cdots f'_1(1)\cdots)$.

Definition C.1 (r-non-injective functions). A function $f:[n]\to[n]$ is called *r-non-injective* if there exists $A\subseteq[n]$ of size at least r and $b\in[n]$ such that for all $a\in A$, $f(a)=b$.

Definition C.2 (Communication Pointer Chasing(n,p) Problem). Pointer Chasing(n,p) is a $(p, p-1)$ communication problem in which the player i has a function $f_i : [n] \to [n]$ and the goal is to compute $f_1(f_2(\cdots f_p(1)\cdots))$.

Definition C.3 (Communication Equal Limited Pointer Chasing(n,p) Problem). The communication Equal Pointer Chasing(n,p) is a $(2p, p-1)$ communication problem in which the first p players have an instance of the Pointer Chasing(n,p) problem and the other p players have another instance of the Pointer Chasing(n,p) problem. The output of the Equal Pointer Chasing(n,p) is 1 if the solutions of the two instances of Pointer Chasing(n,p) have the same value and 0 otherwise. Furthermore in Equal Limited Pointer Chasing(n,p,r), if there exists r-non-injective function f_i, then the output is 1. Otherwise, the output is the same as the value in Equal Pointer Chasing(n,p).

For a boolean communication problem P, OR_t(P) is defined to be OR of t instances of P and the output of OR_t(P) is `true` iff the output of any of the t instances is `true`. Using a direct sum argument, [GO13] showed that the communication complexity of OR_t(Equal Limited Pointer Chasing(n,p,r)) is t times the communication complexity of Equal Limited Pointer Chasing(n,p,r).

Lemma C.4 ([GO13]). *Let n, p, t and r be positive integers such that $n \geq 5p$, $t \leq \frac{n}{4}$ and $r = O(\log n)$. Then the amount of bits of communication to solve OR_t(Equal Limited Pointer Chasing(n,p,r)) with error probability less than $1/3$ is $\Omega(\frac{tn}{p^{16}\log n}) - O(pt^2)$.*

Lemma C.5 ([GO13]). *Let n, p, t and r be positive integers such that $t^{2p}r^{p-1} < \frac{n}{10}$. Then if there is a protocol that solves Intersection Set Chasing(n,p) with probability of error less than $1/10$ using C bits of communication, there is a protocol that solves OR_t(Equal Limited Pointer Chasing(n,p,r)) with probability of error at most $2/10$ using $C + 2p$ bits of communication.*

Consider an instance of OR_t(Equal Limited Pointer Chasing(n,p,r)) in which $t \leq n^{\delta}, r = \log(n), p = (1/2\delta - 1)$ where $1/\delta = o(\log n)$. By Lemma C.4, the required amount of bits of communication to solve the instance with constant probability of success is $\widetilde{\Omega}(tn)$. Then by applying Lemma C.5, to solve the corresponding Intersection Set Chasing, $\widetilde{\Omega}(tn)$ bits of communication is required.

In the reduction of OR_t(Equal Limited Pointer Chasing(n,p,r)) to Intersection Set Chasing(n,p) (proof of Lemma C.5), the r-non-injective property is preserved. In other words, in the corresponding Intersection Set Chasing instance each player's functions $f_i : [n] \to 2^{[n]}$ is union of t r-non-injective functions $f_i(a) := f_{i,1}(a) \cup \cdots \cup f_{i,t}(a)$[5]. Given that none of the $f_{i,j}$ functions is r-non-injective, the corresponding Set Cover instance will have sets of size at most rt (S-type sets are of size at most t for $1 \leq i \leq p$ and of size at most rt for $p+1 \leq i \leq 2p$). Since $r = O(\log n)$, the corresponding Set Cover instance is $\widetilde{O}(t)$-sparse. As we showed earlier in the reduction from Intersection Set Chasing to Set Cover, the number of sets in the corresponding Set Cover instance is $O(np)$. Thus we have the following result for s-Sparse Set Cover problem.

Theorem C.6. *For $s \leq |\mathsf{U}|^{\delta}$, any streaming algorithm that solves s-Sparse Set Cover$(\mathsf{U}, \mathcal{F})$ optimally with probability of error less than $1/10$ in $(\frac{1}{2\delta} - 1)$ passes requires $\widetilde{\Omega}(|\mathcal{F}|s)$ memory space.*

[5]The Intersection Set Chasing instance is obtained by overlaying the t instances of Equal Pointer Chasing(n,p,r). To be more precise, the function of player i in instance j is $\pi_{i,j} \circ f_{i,j} \circ \pi^{-1}_{i+1,j}$ (π are randomly chosen permutation functions) and then stack the functions on top of each other.