Electronic Colloquium on Computational Complexity, Report No. 113 (2015)

Incidence Geometries and the Pass Complexity of Semi-Streaming Set Cover

Amit Chakrabarti[*] Anthony Wirth[†]

Abstract

Set cover, over a universe of size n, may be modelled as a data-streaming problem, where the m sets that comprise the instance are to be read one by one. A semi-streaming algorithm is allowed only $O(n \operatorname{poly}\{\log n, \log m\})$ space to process this stream. For each $p \geqslant 1$, we give a very simple deterministic algorithm that makes p passes over the input stream and returns an appropriately certified $(p+1)n^{1/(p+1)}$-approximation to the optimum set cover. More importantly, we proceed to show that this approximation factor is essentially tight, by showing that a factor better than $0.99\, n^{1/(p+1)}/(p+1)^2$ is unachievable for a p-pass semi-streaming algorithm, even allowing randomisation. In particular, this implies that achieving a $\Theta(\log n)$-approximation requires $\Omega(\log n/ \log\log n)$ passes, which is tight up to the $\log\log n$ factor.

These results extend to a relaxation of the set cover problem where we are allowed to leave an ε fraction of the universe uncovered: the tight bounds on the best approximation factor achievable in p passes turn out to be $\Theta_p(\min\{n^{1/(p+1)}, \varepsilon^{-1/p}\})$.

Our lower bounds are based on a construction of a family of high-rank incidence geometries, which may be thought of as vast generalisations of affine planes. This construction, based on algebraic techniques, appears flexible enough to find other applications and is therefore interesting in its own right.

[*]Department of Computer Science, Dartmouth College. Work supported in part by NSF Award CCF-1217375.

[†]Department of Computing and Information Systems, The University of Melbourne. Supported in part by an ARC Future Fellowship.

ISSN 1433-8092

1 Introduction

The *set cover* problem is one of the most basic and well-studied optimisation problems in computer science. It features either directly or in various guises in a wide array of applications, such as facility location, information retrieval [2], software test selection, and tableau generation [15]. It is also at the heart of a rich theory spanning approximation algorithms [30] and computational complexity theory [3], where efforts to understand the complexity of set cover have led to interesting combinatorial and mathematical interactions. In this work, we consider set cover as a "big data" problem; specifically, we are concerned with space-efficient algorithms for set cover in the well-established *data streaming* model [26, 4]. This setting has been studied in several recent works, including Saha & Getoor [28], Emek & Rosén [12], and Demaine et al. [10].

An instance of set cover is given by a pair $(\mathcal{X}, \mathcal{F})$, where $\mathcal{X}$ is a finite universe with cardinality $|\mathcal{X}| = n$ and $\mathcal{F} \subseteq 2^{\mathcal{X}}$ is a finite collection (multiset) of subsets of $\mathcal{X}$ with cardinality $|\mathcal{F}| = m$. The pair $(\mathcal{X}, \mathcal{F})$ satisfies the guarantee that the sets in $\mathcal{F}$ together cover $\mathcal{X}$, i.e., $\bigcup_{S \in \mathcal{F}} S = \mathcal{X}$. A candidate solution to the instance is a subcollection $Sol \subseteq \mathcal{F}$; it is said to be *feasible* if $\bigcup_{S \in Sol} S = \mathcal{X}$. Its *cost* is defined to be the cardinality $|Sol|$.[1] The desired goal is to find a feasible solution while keeping its cost small. A feasible solution with minimum possible cost is said to be *optimal* and its cost is called the *optimum cost* or *optimum value* of the instance. Henceforth we shall call this problem $\textsc{set-cover}_{n,m}$, or simply $\textsc{set-cover}$.

It is well-known that finding an optimal solution to $\textsc{set-cover}$ is NP-hard [18]; that finding an α-approximate solution—defined as a feasible solution whose cost is at most α times that of the optimum—is possible in polynomial time for $\alpha = \ln n - \ln \ln n + \Theta(1)$ [29]; and that doing so for $\alpha < (1 - \varepsilon) \ln n$ is impossible unless $\mathsf{NP} = \mathsf{P}$ [11]. Thus, for traditional Turing Machine computation, the complexity of $\textsc{set-cover}$ is essentially fully understood. However, for genuinely huge instances of $\textsc{set-cover}$, additional considerations become important: how will the data be accessed and how will it be manipulated in a relatively small amount of working memory?

This motivates a careful study of the complexity of $\textsc{set-cover}$ in a data-streaming setting. The instance $(\mathcal{X}, \mathcal{F})$ is presented as a stream consisting of the sets in $\mathcal{F}$, one at a time; the universe $\mathcal{X}$ is known in advance, so we may assume that $\mathcal{X} = [n] := \{1, 2, \ldots, n\}$. Representing an instance of $\textsc{set-cover}_{n,m}$ requires $\Theta(mn)$ bits in general. Thus, in $\Theta(mn)$ bits of space (working memory), we could simply run our favourite offline algorithm. The challenge is to work with *sublinear*—i.e., $o(mn)$—space. A *p-pass* algorithm may read its input stream up to p times; this parameter p, sometimes called the pass complexity, ought to be a small constant, or perhaps $O(\log n)$. Of course, in addition to space and pass efficiency, we would also want our algorithms to process each set quickly, with very simple operations and logic.

Since $\Omega(n)$ space is required simply to certify that a computed solution is feasible, we shall think of an algorithm as highly space-efficient if it uses $\widetilde{O}(n) := O(n \operatorname{poly}\{\log n, \log m\})$ space. Following a convention started with the study of streaming graph algorithms [14], and continued in this context by Emek & Rosén [12], we shall call such an algorithm a *semi-streaming algorithm*. Emek & Rosén undertook a detailed study of *one-pass* semi-streaming algorithms for $\textsc{set-cover}$, obtaining nearly tight bounds on the best approximation ratio achievable by such algorithms. In this work, we provide tight bounds for the multi-pass case, giving an almost complete understanding of the pass/approximation tradeoff for semi-streaming algorithms. In particular, this answers an open question explicitly raised by Saha & Getoor [28].

[1]In *weighted* set cover, each set $S \in \mathcal{F}$ has a cost or *weight* $w(S) \geqslant 0$ and the cost of Sol is $\sum_{S \in Sol} w(S)$. The major contributions of this work being lower bounds, we focus on the purely combinatorial setting, which is of course a strength for lower bounds.

1.1 Our Results and Techniques

A classic result of Johnson [17], refined by Slavík [29], gives a $(\ln n - \ln\ln n + \Theta(1))$-approximation to SET-COVER by a greedy algorithm. Given an instance $(\mathcal{X}, \mathcal{F})$, at each step, it adds to the current solution the set from $\mathcal{F}$ that *contributes* most, i.e., covers the largest number of as-yet-uncovered elements. Notice that this can be implemented as a semi-streaming algorithm, using one pass for each step, but this leads to $\Omega(n)$ passes, which is ridiculously expensive. Saha & Getoor [28] gave a different algorithm, which guarantees an $O(\log n)$-approximation using only $O(\log n)$ passes. Emek & Rosén [12] asked how good an approximation is possible for a one-pass semi-streaming algorithm. They showed that an approximation ratio of $O(\sqrt{n})$ is achievable and that the ratio must be $\Omega(n^{1/2-\delta})$ for every constant $\delta > 0$; Section 1.2 adds some detail. Our first result generalises their upper bound, trading off additional passes for improved approximation.

Result 1 (Formalised as Theorem 2.5)**.** *In p passes, within semi-streaming space bounds, we can compute a $(p+1)n^{1/(p+1)}$-approximate solution to* SET-COVER *together with an appropriate "certificate of coverage."*

The algorithm behind Result 1 is a variant of the greedy approach wherein each pass picks sets that contribute above some well-chosen threshold for that pass, and the sequence of thresholds is geometrically decreasing. This kind of thresholding is itself a variant of ideas introduced by Cormode, Karloff, and Wirth [9] in a non-streaming context. Our algorithm needs one final "folding" trick that considers the final two thresholds in the sequence in a single pass.

The Emek–Rosén algorithm solves a more general problem, with set weights and a relaxed feasibility condition (partial coverage, which we describe below). For the basic combinatorial SET-COVER problem, our algorithm nevertheless makes a (small) contribution even in the one-pass case, with the simplicity of its logic as compared to Emek–Rosén: our logic, being a variant of the basic greedy approach, is arguably easier to implement and analyse. But most importantly, this algorithm sets the stage for our main result, which gets at the *pass complexity* of the problem.

Result 2 (Main result, formalised as Theorem 3.8)**.** *In p passes, approximating the optimum of a* SET-COVER *instance to a factor smaller than $0.99\, n^{1/(p+1)}/(p+1)^2$ requires more than semi-streaming space. This applies even to the decision problem of distinguishing a small optimum value from a large one.*

Results 1 and 2 together provide a near-complete understanding of the power of each additional pass in improving the quality of an approximate solution to SET-COVER. Saha & Getoor had posed the problem of obtaining this kind of tradeoff as an open question. Result 2 immediately implies that obtaining an $O(\log n)$-approximation under semi-streaming space bounds requires $\Omega(\log n/\log\log n)$ passes, almost matching the pass complexity of the Saha–Getoor algorithm (or, for that matter, the algorithm behind our Result 1).

In establishing Result 2, we invent a family of novel combinatorial structures that we call *edifices*. To explain these, we first consider $p = 1$. In this case, an $\Omega(\sqrt{n})$ bound follows from a reduction from the INDEX problem in communication complexity, via set systems based on affine planes of finite order.[2] A "hard instance" for one pass consists of a family of sets of two different sizes: one "large" set and many "medium" sets with very small pairwise intersections. The family of lines in $\mathbb{F}^2$, where $\mathbb{F}$ is a finite field, gets us most of the way towards the desired properties. To generalise this to $p > 1$, we reduce from the multi-party communication problem POINTER-JUMPING. For this reduction, we need a more elaborate set system with sets of many different sizes (similar to contribution thresholds in the multi-pass algorithms) and a tree-like incidence

[2]Emek & Rosén also use affine planes, but differently, and obtain an $\Omega(n^{1/2-\delta})$ bound versus our $\sqrt{n}/(4+\delta)$.

structure, plus a small-intersection property as before. Very roughly, for $p = 2$, we start by considering quadric surfaces inside $\mathbb{F}^3$, and then lines in $\mathbb{F}^2$ lifted onto these surfaces; for higher p, we start with the appropriate extensions of these ideas to higher-degree algebraic varieties. These varieties form a certain incidence geometry—we call it an edifice—that is a vast generalisation of affine planes. Bounding the sizes of certain pairwise intersections between these varieties is the most technical part of this work.

Following Emek & Rosén, we also study *partial* set covers. In the PARTIAL-COVER$_{n,m,\varepsilon}$ problem, an instance $\mathcal{I} = (\mathcal{X}, \mathcal{F}, \varepsilon)$ consists of $\mathcal{X}$, $\mathcal{F}$, and a parameter $\varepsilon \in [0, 1]$. We require a $(1 - \varepsilon)$-partial cover of $\mathcal{X}$: a collection $Sol \subseteq \mathcal{F}$ that covers at least $(1 - \varepsilon)|\mathcal{X}|$ elements. A solution Sol is α-approximate if $|Sol| \leqslant \alpha |Opt|$, where Opt is a minimum-cost *total* set cover for $(\mathcal{X}, \mathcal{F})$.

Result 3 (Formalised as Theorems 4.1 and 4.4)**.** *The smallest α for which a semi-streaming algorithm can compute an α-approximate $(1 - \varepsilon)$-partial cover is in $\Theta_p(\min\{n^{1/(p+1)}, \varepsilon^{-1/p}\})$. The lower bound applies to a decision problem of distinguishing a small total cover from a necessarily large partial cover.*

The upper bound in Result 3 builds on the one-pass Emek–Rosén algorithm; thus we lose the extreme simplicity of the algorithm behind Result 1, but gain the ability to handle *weighted* instances. The main contribution is again the lower bound. It requires a reexamination of the edifices constructed for establishing Result 2 and proving that they satisfy additional geometric properties. These properties then allow us to build new edifices with different parameters that are suited to the problem at hand. This construction shows the power of the axiomatic approach we take in defining edifices.

We note in passing the minor result (formalised as Theorem 3.9) that a tweak to Result 2 gives a rounds/approximation tradeoff for a two-player communication version of SET-COVER à la Nisan [27] and Demaine et al. [10].

1.2 Related Work

The quantification of savings afforded by extra streaming passes dates back to Munro & Paterson [25], who studied pass/space tradeoffs for median-finding. This general topic remains current [16, 8, 22, 7].

Efforts to understand the hardness of SET-COVER have led to many deep insights and connections with various kinds of mathematics. Our technical contributions continue this tradition. In the series of hardness-of-approximation results beginning with Lund & Yannakakis [21, 13, 24], recently culminated in Dinur & Steurer [11], each result required new insights into PCPs and parallel repetition; for details, see the latter paper and the references therein. Closer to this work, Nisan [27] initiated the study of SET-COVER as a (two-player) communication problem and showed that, for every constant $\delta > 0$, computing a $(\frac{1}{2} - \delta) \log_2 n$-approximation to SET-COVER$_{n,m}$ requires $\Omega(m)$ randomised communication. His "hard instances" used $m \approx \exp(\sqrt{n})$. Nisan's original motivation was combinatorial auctions, but his result can be interpreted in the data-streaming setting as saying that a semi-streaming $(\frac{1}{2} - \delta) \log_2 n$-approximation is impossible, regardless of the number of passes. Demaine et al. [10] showed that *deterministic* streaming algorithms achieving a $\Theta(1)$-approximation require $\Omega(mn)$ space, thereby ruling out sublinear-space solutions altogether.

All of the above lower bounds have, at their core, some variant of an old combinatorial construction: namely, that of a set system with the so-called r-covering property [21]. Our own combinatorial constructions (of edifices) play an analogous role in our lower bounds, but are quite different at a technical level. In particular, they result in SET-COVER$_{n,m}$ instances where $m = n^{\Theta(1)}$. Their closest relative is the construction in Emek & Rosén [12] based on lines in an affine plane.

Turning to upper bounds, traditional (offline) approximation algorithms for SET-COVER are discussed at length in Vazirani [30]; see also Slavík [29] and the references therein. Alon et al. [1] studied SET-COVER in an online setting, focussing on competitive ratios rather than space considerations, but under a fundamentally different input model: the sets are known in advance and elements of the universe $\mathcal{X}$ arrive in a stream. The setting we study was first considered by Saha & Getoor [28], who called it "set streaming." They gave a 4-approximation algorithm for MAX-k-COVERAGE, the problem of choosing k sets from the stream so as to maximise the cardinality of their union. Iterating this algorithm for $O(\log n)$ passes immediately gives an $O(\log n)$-approximation for SET-COVER. Cormode, Karloff, and Wirth [9], targeting external-memory efficiency, developed a "disk-friendly greedy" (DFG) algorithm for SET-COVER. In short, each step of DFG adds some set whose contribution is at least $1/\beta$ times the maximum. As designed, DFG yields an $O(\log_\beta n)$-pass, $(1 + \beta \ln n)$-approximate, $O(n \log n)$-space streaming algorithm.

The single-pass semi-streaming setting was first, and thoroughly, studied by Emek & Rosén [12]. Indeed, their results extend to PARTIAL-COVER, as well as item- and set-weighted variants. Their algorithm, like ours, computes a *certificate of coverage* that indicates, for each item, which set (if any) covers it: the implied solution *Sol* covers a $1 - \varepsilon$ (weighted) fraction of $\mathcal{X}$ and has $w(Sol) = O(\min\{1/\varepsilon, \sqrt{n}\} w(Opt))$. On the lower bound side, they prove that for every $\varepsilon \geqslant 1/\sqrt{n}$, a randomised semi-streaming algorithm that *certifies* an (unweighted) α-approximate $(1 - \varepsilon)$-cover must have $\alpha = \Omega(1/\varepsilon)$. Outputting only the sets in a solution (without a certificate) still requires $\alpha = \Omega(\varepsilon^{-1} \log\log n / \log n)$. The still-weaker problem of approximating the optimum *value* requires $\alpha = \Omega(n^{1/2-\delta})$ for every constant $\delta > 0$. Emek & Rosén remark [12, footnote 3] that they can show this only for SET-COVER, and not for $(1 - \varepsilon)$-PARTIAL-COVER with $\varepsilon \gg 1/\sqrt{n}$. Compare these lower bounds with our Results 2 and 3, specialised to $p = 1$.

The main result of Demaine et al. [10], whose deterministic lower bound we have discussed, is a *randomised* sublinear-space, though not semi-streaming, algorithm for SET-COVER. It achieves an $O(4^{1/\delta})$-pass, $O(4^{1/\delta}\rho)$-approximation in $\tilde{O}(mn^\delta)$ space, where ρ is the approximation ratio of whatever offline SET-COVER algorithm we are prepared to run.

2 A Simple Deterministic Multi-Pass Algorithm

Model of computation. An instance of SET-COVER$_{n,m}$ consists of sets $S_1, \ldots, S_m \subseteq [n]$, specified as a stream of tokens (i, S_i), where S_i is described in some reasonable way (either as a list of its elements or as a characteristic vector) and i is the ID of S_i. The IDs need not appear in the order $1, 2, \ldots, m$. The desired output is a set $Sol \subseteq [m]$ consisting of the IDs of sets that together cover $[n]$, plus a certificate: an array $Coverer[1 \ldots n]$ in which, for each x, $Coverer[x]$ is the ID of a set that covers j. Strictly speaking, *Sol* is redundant because it can be computed from *Coverer*, but keeping track of it explicitly aids exposition.

Recall that a semi-streaming algorithm is allowed $\widetilde{O}(n) := O(n \operatorname{poly}\{\log n, \log m\})$ bits of space. This clearly suffices to represent each of *Sol* and *Coverer*, which need only $\Theta(n \log m)$ bits, under the sensible assumption that $|Sol| \leqslant n$. An ideal semi-streaming algorithm for SET-COVER would use no more space than this, asymptotically, and our Algorithms 1 and 2 achieve this space bound.

2.1 Algorithm and Analysis

As promised, we begin by giving a very simple deterministic p-pass, semi-streaming, "progressive greedy" algorithm that returns a $(p + 1)n^{1/(p+1)}$-approximation. The basic idea is that the first pass is very conservatively greedy, picking a set into the solution iff its *contribution* is at least some large number τ_1 (i.e., it covers at least τ_1 as-yet-uncovered elements); the second pass repeats this logic with a threshold $\tau_2 < \tau_1$, making it slightly less conservative; and so on. Choosing suitable thresholds gets us to a p-pass $pn^{1/p}$-approximation. This is the naïve version of progressive greedy. Our final algorithm "folds" the last two passes of this naïve version into a single pass, achieving the desired bound.

Algorithm 1 Naïve version of "progressive greedy" algorithm for SET-COVER, in p passes

```
 1: procedure GreedyPass(stream $\sigma$, threshold $\tau$, set $Sol$, array $Coverer$)
 2:     foreach $(i, S)$ in $\sigma$ do
 3:         $C \leftarrow \{x : Coverer[x] \neq 0\}$            ▷ $C$ is the set of already covered elements
 4:         if $|S \setminus C| \geqslant \tau$ then
 5:             $Sol \leftarrow Sol \cup \{i\}$
 6:             foreach $x \in S \setminus C$ do $Coverer[x] \leftarrow i$

 7: procedure ProgGreedyNaive(stream $\sigma$, integer $n$, integer $p \geqslant 1$)
 8:     $Coverer[1 \ldots n] \leftarrow 0^n$;  $Sol \leftarrow \varnothing$
 9:     for $j = 1$ to $p$ do GreedyPass($\sigma, n^{1-j/p}, Sol, Coverer$)
10:     output $Sol, Coverer$
```

For ease of reading we have not optimised the per-token processing time in our pseudocode. Clearly, in each pass, each set S can be processed in $O(|S|)$ time in a RAM-style machine with word size $\Omega(\log m)$.

To analyse Algorithm 1, fix an arbitrary instance of SET-COVER$_{n,m}$. Each call to GreedyPass makes a single pass through σ, considering every set S. Note that the contribution of S in such a pass is the quantity $|S \setminus C|$, computed in line 4, which is the number of *new* elements that S covers. Let $Opt \subseteq [m]$ be an optimum solution. For ease of exposition we will pretend that Opt and Sol are collections of sets from the input instance (they are in fact collections of IDs of such sets).

Definition 2.1. A (τ, ρ)-bounded pass is a run of GreedyPass with threshold τ where, if C_0 is the set of covered elements at the start of the pass, then for all S in σ we have $|S \setminus C_0| \leqslant \rho\tau$.

Lemma 2.2. *A (τ, ρ)-bounded pass adds at most $\rho|Opt|$ sets to Sol.*

Proof. Put $D = [n] \setminus C_0$. Each set in Opt includes at most $\rho\tau$ of the elements in D, yet the sets in Opt together cover D. Therefore $|Opt| \geqslant |D|/(\rho\tau)$. Meanwhile, in this pass, each set added to Sol includes at least τ elements of D, so the pass adds at most $|D|/\tau \leqslant \rho|Opt|$ sets to Sol. □

Lemma 2.3. *Algorithm 1 is a p-pass semi-streaming $pn^{1/p}$-approximation algorithm for* SET-COVER$_{n,m}$.

Proof. The algorithm's correctness and $\widetilde{O}(n)$ space bound are obvious, so we focus on the approximation ratio. We claim that for each $j \in [p]$, the jth pass of Algorithm 1 is $(n^{1-j/p}, n^{1/p})$-bounded.

Let us prove this claim. Put $\tau_j = n^{1-j/p}$. For $j = 1$, the precondition required by Definition 2.1 is trivially satisfied. For larger j, consider an arbitrary set S in σ and let C_0 be as in Definition 2.1, for the jth pass. If S were added to Sol in an earlier pass, then $|S \setminus C_0| = 0$. If not, then by the logic of GreedyPass, set S's contribution was less than τ_{j-1} during the $(j-1)$th pass. Since C_0 is a superset of the set of elements that had been covered when S was processed in the $(j-1)$th pass, we have $|S \setminus C_0| < \tau_{j-1} = n^{1/p}\tau_j$.

Having proved the claim, it follows from Lemma 2.2 that each pass adds at most $n^{1/p}|Opt|$ sets to Sol. Therefore, in the end we have $|Sol| \leqslant pn^{1/p}|Opt|$, as required. □

In fact, since the first pass adds at most $n^{1/p}$ sets, we have $|Sol| \leqslant n^{1/p}(1 + (p-1)|Opt|)$.

Folding the last two passes. The final pass of Algorithm 1 picks a set merely for making a *nonzero* contribution. When there are at least two passes, this final-pass logic can be "folded into" the penultimate pass as follows. During the pth pass of a $p+1$-pass scheme, we run GREEDYPASS as usual and additionally, in parallel, run a second instance of GREEDYPASS with threshold 1 that builds an alternate solution *Alt* (certified by a new array *Backup*, analogous to *Coverer*), starting from $\varnothing$. Thus, *Alt* is the solution that a 1-pass version of Algorithm 1 would have built. At the end of the penultimate (pth) pass, *Sol* might have left some elements of $\mathcal{X}$ uncovered. We fix this by post-processing: for each such element x, we add to *Sol* the set in *Alt* that covered x; this information can be read from *Backup*. Algorithm 2 implements this very idea.

Algorithm 2 Progressive greedy algorithm for SET-COVER in p passes

```
1: procedure ProgGreedy(stream $\sigma$, integer $n$, integer $p \geqslant 1$)
2:     $Coverer[1 \ldots n], Backup[1 \ldots n] \leftarrow 0^n$;  $Sol, Alt \leftarrow \varnothing$
3:     for $j = 1$ to $p-1$ do GreedyPass($\sigma, n^{1-j/(p+1)}, Sol, Coverer$)
4:     in parallel, do GreedyPass($\sigma, n^{1-p/(p+1)}, Sol, Coverer$) and GreedyPass($\sigma, 1, Alt, Backup$)

5:     for $x = 1$ to $n$ do          ▷ Post-processing: elements not covered by Sol will get covered by sets from Alt
6:         if $Coverer[x] = 0$ then
7:             $Sol \leftarrow Sol \cup Backup[x]$
8:             $Coverer[x] \leftarrow Backup[x]$
9:     output $Sol, Coverer$
```

Lemma 2.4. *For every stream σ, the output of* PROGGREEDY(σ, p) *in Algorithm 2 is identical to that of* PROGGREEDYNAIVE$(\sigma, p+1)$ *in Algorithm 1.*

Proof. Fix an input stream σ. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ denote, respectively, the invocation of Algorithm 1 as PROGGREEDYNAIVE$(\sigma, p+1)$ and the invocation of Algorithm 2 as PROGGREEDY(σ, p). Let Sol_1 be the value of *Sol* after p passes of $\mathcal{A}_1$. It is immediate that Sol_1 is also the value of *Sol* in $\mathcal{A}_2$ just before the post-processing loop in lines 5 to 8.

Let Cov_1 and Cov_2 denote, respectively, the final *output* values of the array *Coverer* in $\mathcal{A}_1$ and $\mathcal{A}_2$. Let $C = \bigcup_{S \in Sol_1} S$. Our above observation says that $Cov_1[x] = Cov_2[x]$ for all $x \in C$. It remains to prove that the same equality also holds for all $x \in [n] \setminus C$. But this, too, is immediate from the observation that for each $x \in [n] \setminus C$, each of $Cov_1[x]$ and $Backup[x]$, and thus $Cov_2[x]$ as well, is set to the earliest set in σ that contains x. □

Theorem 2.5. *There is a p-pass, $O(n \log m)$-space algorithm that, for every instance of* SET-COVER$_{n,m}$, *outputs a feasible solution Sol with* $|Sol| \leqslant n^{1/(p+1)}(1 + p|Opt|) \leqslant (p+1)n^{1/(p+1)}|Opt|$.

Proof. This follows immediately by combining Lemma 2.3 with Lemma 2.4. □

Folding three passes? It is natural to wonder whether the above "folding" idea can be taken further, achieving an even better pass/approximation tradeoff. As it turns out, we cannot fold (the last) three passes into one. The most convincing proof is the lower bound that we shall establish in Section 3.

As designed, the algorithm cannot be sure what the contribution of a set will be in a particular pass until it actually sees this set in that pass. In the last pass, however, we need only know that the contribution is non-zero: after the penultimate pass, if $Coverer[x] = 0$ and $Backup[x] = i$, we know "in advance" that set S_i has non-zero contribution.

2.2 Tightness of Analysis

The lower bound in Section 3 shows that the approximation ratio guaranteed by Theorem 2.5 is asymptotically optimal for $p = \Theta(1)$ passes. But if p is allowed to grow with n, then there remains a small $\Theta(p^3)$ discrepancy between that upper bound and the lower bound we shall eventually prove in Theorem 3.8. We can, however, prove that our analysis of the approximation guarantee of Algorithm 1 is tight.

Theorem 2.6. *For each integer $p \geqslant 2$ and q large enough, there is an instance $\mathcal{I}_{n,m}$ of* SET-COVER$_{n,m}$, *with $n = q^p - 1$ and $m \leqslant pq$, such that $\mathcal{I}_{n,m}$ admits a set cover of size 1, whereas Algorithm 1, using p passes and running on $\mathcal{I}_{n,m}$, returns a solution with $p(q-1) \approx pn^{1/p}$ sets.*

Proof. Put $\mathcal{X} = [q]^p \setminus \{(q, q, \ldots, q)\}$. For each $j \in [p]$ and $y \in [q-1]$, define the sets

$$\begin{aligned} \mathcal{X}_j &= \{(x_1, \ldots, x_p) \in \mathcal{X} :\ x_1 = \cdots = x_{j-1} = q\}\,, \\ S_j^y &= \{(x_1, \ldots, x_p) \in \mathcal{X} :\ x_1 = \cdots = x_{j-1} = q \wedge x_j = y\}\,. \end{aligned}$$

Then $\mathcal{X} = \mathcal{X}_1 \supseteq \cdots \supseteq \mathcal{X}_p$ and $S_j^y \subseteq \mathcal{X}_j$. Observe that these sets S_j^y are pairwise disjoint and partition $\mathcal{X}$. Further, $|S_j^y| = q^{p-j}$ and $|\mathcal{X}_j| = q^{p-j+1} - 1$ for all j, y.

Let σ_j be the stream consisting of the sets $\{S_j^y : y \in [q-1]\}$ in some arbitrary order. Let σ be the stream consisting of σ_p followed by σ_{p-1} and so on, down to σ_1, and finally the set $\mathcal{X}$. Consider the SET-COVER$_{n,m}$ instance $\mathcal{I}_{n,m}$ defined by σ: it satisfies $n = |\mathcal{X}| = q^p - 1$ and $m = p(q-1) + 1 \leqslant pq$, as claimed. Since the entire universe $\mathcal{X}$ occurs as a set in $\mathcal{I}_{n,m}$, the optimum set cover consists of just that one set.

Now consider the behaviour of Algorithm 1 on σ. For each $j \in [p]$, let $\tau_j = n^{1-j/p}$ be the threshold in the jth pass. We claim that

$$q^{p-j} - 1 < \tau_j \leqslant q^{p-j}\,. \tag{1}$$

The second inequality in (1) is easy to see: $\tau_j = \left((q^p - 1)^{1/p}\right)^{p-j} \leqslant q^{p-j}$. The first inequality is obvious when $j = p$, so suppose that $1 \leqslant j < p$. Consider the function

$$G_{j,p}(x) = (x^p - 1)^{1-j/p} - (x^{p-j} - 1)\,.$$

A routine calculation shows that the derivative $G'_{j,p}(x) = (p-j)x^{p-1}((x^p-1)^{-j/p} - x^{-j})$. For $x \geqslant 1$, we have $(x^p - 1)^{1/p} < x$, so $(x^p-1)^{-j/p} > x^{-j}$; therefore $G'_{j,p}(x) > 0$. Since $G_{j,p}(1) = 0$, we now conclude that $G_{j,p}(q) > 0$, which gives us the first inequality in (1) and proves the claim. We can now see that the j^{th} pass satisfies the following properties.

1. At the start of the pass, the set of uncovered elements is precisely $\mathcal{X}_j$.
2. Each set in $\sigma_p, \ldots, \sigma_{j+1}$ makes a contribution equal to its cardinality. Therefore the largest such contribution is $q^{p-j-1} < q^{p-j} - 1 < \tau_j$, by (1).
3. Each set in σ_j makes a contribution equal to its cardinality, which is $q^{p-j} \geqslant \tau_j$, by (1).
4. Each set in $\sigma_{j-1}, \ldots, \sigma_1$ makes a contribution of zero.
5. The set $\mathcal{X}$, which arrives at the end of σ, makes a contribution of $q^{p-j} - 1 < \tau_j$, by (1).
6. Therefore the sets added to *Sol* during the pass are exactly the sets in σ_j.

The validity of these properties can be formally proved by backward induction on j. The details are routine and tedious, so we omit them.

Based on the above properties, we see that Algorithm 1 produces a solution consisting of all sets in all substreams σ_j. The number of sets in this solution is $p(q-1)$, as claimed. $\square$

3 The Basic Lower Bound

In this section we establish our main result, which gives a strong lower bound on the best approximation ratio achievable by a semi-streaming algorithm for SET-COVER. Our lower bound gives the optimal dependence of this ratio on n. Moreover, for p passes, our lower bound is only about p^3 times smaller than our upper bound in Theorem 2.5. In particular, when $p = \Theta(1)$, the lower bound is asymptotically optimal.

3.1 Warm Up: One-Pass Algorithms

Our proof is based on a fairly technical combinatorial construction. To motivate it, let us first outline a simple proof of a *one-pass* lower bound. We start with the well-known INDEX (or IDX) problem in communication complexity, where Alice must send Bob a (possibly random) message about her n-bit string x, so that Bob, who holds an index $h \in [n]$, can output x_h (the hth bit of x) with high probability. A textbook result [20] is that this requires Alice to send $\Omega(n)$ bits. To reduce IDX to SET-COVER, we construct a universe $\mathcal{X}$ and a family of n distinct sets $S_1, \ldots, S_n \subset \mathcal{X}$. Alice encodes x as the stream of sets $\{S_i : x_i = 1\}$, and Bob encodes h as a "stream" of just one set: $\mathcal{X} \setminus S_h$. Alice's stream followed by Bob's is an instance of SET-COVER.

When $x_h = 1$, this instance clearly has $|Opt| = 2$. We can force $|Opt|$ to be much larger when $x_h = 0$ if we make each $|S_i|$ large and each $|S_i \cap S_j|$ small (for $i \neq j$): since Alice's stream is missing S_h, it will take "many" sets S_i, $i \neq h$, to cover the elements of S_h.

Incidence geometry gives us an elegant construction of a collection $\{S_1, \ldots, S_n\}$ with these properties. Consider the lines of an affine plane of order q, with q a prime power. More explicitly, let $\mathbb{F}_q$ denote the finite field with q elements, $\mathcal{X} = \mathbb{F}_q^2$, $n = |\mathcal{X}| = q^2$, and $\{S_1, \ldots, S_n\}$ be some collection of n distinct lines out of the $q^2 + q$ such lines in $\mathbb{F}_q^2$. Then each $|S_i| = q$ and each $|S_i \cap S_j| \leqslant 1$, for $i \neq j$. In particular, $x_h = 0$ now implies that $|Opt| \geqslant q = \sqrt{n}$. Therefore approximating such a SET-COVER instance to a factor smaller than $\sqrt{n}/2$ is enough to solve IDX, whence an algorithm achieving such approximation must use $\Omega(n)$ space.

To rule out a semi-streaming algorithm we must prove a stronger, $n^{1+\Omega(1)}$ space, lower bound. A simple tweak achieves this: sticking with the universe $\mathbb{F}_q^2$, replace the lines in the above construction with degree-2 algebraic curves, say. This preserves the essential dichotomy between large $|S_i|$ and small $|S_i \cap S_j|$ while allowing us to reduce from an IDX instance on $n^{1+\Omega(1)}$ bits.

The one-pass lower bound proof we have just outlined is arguably more straightforward than the Emek–Rosén proof [12]. Though both proofs begin with the affine plane, our builds an explicit set system, rather than relying on a probabilistic argument, and reduces directly from IDX, rather than a employing bespoke entropy calculations, leading to a more modular proof. But there is a far more important takeaway from our proof: the observation that employing higher-degree curves adds great flexibility to the construction. Exploiting this observation to its fullest allows us to handle multi-pass algorithms by greatly generalising the construction, moving from affine planes to more abstract incidence geometries that we call *edifices* (Definition 3.3 below). Edifices, like affine planes, are examples of Buekenhout geometries [6].

3.2 Multi-Player Tree Pointer Jumping

A popular source problem for multi-pass streaming lower bounds is the communication problem *multi-player tree pointer jumping*, which generalises IDX. Let T be a rooted tree with $k \geqslant 2$ layers of vertices, where a vertex is in layer ℓ if it is at distance exactly $k - \ell$ from the root (thus, the root is in layer k) and every leaf is in layer 1. The pointer jumping problem on T, denoted MPJ_T, is a k-player number-in-hand communication game involving players named $\text{PLR}_1, \ldots, \text{PLR}_k$. For $1 < j \leqslant k$, PLR_j's input specifies one *pointer* (i.e., out-edge) at each vertex in layer j; by definition each such pointer leads to a vertex in layer $j - 1$. Furthermore, PLR_1's input specifies a bit at each layer-1 vertex; these bits are called *leaf bits*. Given such an input, π, let $T|_\pi$ denote the subgraph of T defined by retaining only those edges of T that correspond to pointers in π. Then $T|_\pi$ contains a unique root-to-leaf path, ending at a leaf v_π, say. The desired output corresponding to π, denoted $\text{MPJ}_T(\pi)$, is defined to be the leaf bit at v_π.

The communication game involves players announcing messages on a shared broadcast channel, according to a public-coin randomised protocol. The protocol proceeds in *rounds*, where a round is defined as one message each from $\text{PLR}_1, \ldots, \text{PLR}_k$, speaking in that order. The last message of the protocol must be a single bit, which is defined to be the protocol's output. An $[r, C, \varepsilon]$-protocol for MPJ_T is defined to be one in which

- there are at most r rounds of communication;
- within each round, the total number of bits communicated is at most C; and
- the protocol's output equals $\text{MPJ}_T(\pi)$ with probability at least $1 - \varepsilon$.

Definition 3.1. The r-round randomised communication complexity of MPJ_T is defined to be $\mathrm{R}^r(\text{MPJ}_T) := \min\{C : \text{there exists an } [r, C, \tfrac{1}{3}]\text{-protocol for } \text{MPJ}_T\}$.

Intuitively, if players trying to solve MPJ_T are restricted to a "small" amount of communication per round, then because they are forced to speak in the "wrong" order, in the first round the only player who is able to convey "useful" information is PLR_k, in the second round the only such player is PLR_{k-1}, and so on. Therefore, if the protocol is further restricted to $k - 1$ rounds, PLR_1 rarely gets a chance to convey useful information and so the protocol's error probability should be high. This intuition was formalised in the *round elimination* ideas of Miltersen et al. [23]. Using these ideas and a *direct sum* argument, Chakrabarti, Cormode, and McGregor [8] proved a distributional communication complexity lower bound for MPJ. We only need the consequent randomised communication complexity bound, stated below.

Theorem 3.2 ([8, Theorem 4.5])**.** *Let T be a complete t-ary tree with $k \geqslant 2$ layers of vertices. Then* $\mathrm{R}^{k-1}(\text{MPJ}_T) = \Omega(t/k^2)$. □

3.3 Reduction to Set Cover via Edifices

Definition 3.3. A (k, d, q, t)-*edifice* $\mathcal{T}$ over a universe $\mathcal{X}$ is a rooted tree, together with an associated collection of sets called the *varieties* of the edifice, satisfying the following properties.

(E1) $\mathcal{T}$ is a complete t-ary tree, i.e., every non-leaf vertex has exactly t children.

(E2) $\mathcal{T}$ has k levels (equivalently, depth $k - 1$), numbered 1 through k from leaves to root.

(E3) Each vertex v of $\mathcal{T}$ has an associated set $X_v \subseteq \mathcal{X}$, called the *variety* at v.

(E4) If u is the parent of v, then $X_u \supseteq X_v$. If r is the root of $\mathcal{T}$, then $X_r = \mathcal{X}$.

(E5) If z is a leaf of $\mathcal{T}$, then $|X_z| \geqslant q$.

(E6) For each leaf z of $\mathcal{T}$ and each vertex v not an ancestor of z, we have $|X_z \cap X_v| \leqslant d + k - 1$.

The (k, d, q, t)-edifices that interest us will have $k \approx d \ll q \ll t$. In particular, if $d + k \leqslant q$ and $t \geqslant 2$, it is easy to prove from (E4), (E5), and (E6) that varieties at distinct vertices are distinct (as sets). For readers familiar with incidence geometry, we note that these varieties then form a Buekenhout geometry [6] of rank k, where the type map sends each variety to the level of its corresponding vertex and the incidence relation is symmetrised set inclusion. Thus our notion of an edifice generalises affine planes, which we used in our warm-up proof: an affine plane over $\mathbb{F}_q$ is a $(2, 0, q, q^2 + q)$-edifice over the universe $\mathbb{F}_q^2$.

Theorem 3.4. *Suppose there exists a $(p + 1, d, q, t)$-edifice $\mathcal{T}$ with $q > (p + d)(p + 1)$. Then every randomised p-pass streaming algorithm that, with probability at least $2/3$, approximates* SET-COVER *to a factor smaller than $q/((p + d)(p + 1))$ must use at least $\mathrm{R}^p(\text{MPJ}_{\mathcal{T}})/p = \Omega(t/p^3)$ bits of space.*

Proof. Let the edifice $\mathcal{T}$ be over a universe $\mathcal{X}$. We shall transform an input π to $\text{MPJ}_{\mathcal{T}}$ into an instance $\mathcal{I}(\pi)$ of SET-COVER on the universe $\mathcal{X}$, with each set in $\mathcal{I}(\pi)$ being assigned to one of $\text{PLR}_1, \ldots, \text{PLR}_{p+1}$.

The transformation is as follows. Let u be a vertex of $\mathcal{T}$ in layer $j \geqslant 2$. Then π specifies a pointer from u to some vertex, say v. We encode this pointer as the set $X_u \setminus X_v$ and assign this set to PLR_j. We perform this encoding for each vertex in layers 2 and higher. Furthermore, we encode the leaf bits of π as the collection of sets $\{X_z : \pi \text{ specifies a '1' at leaf } z\}$ and assign all sets in this collection to PLR_1. Finally, we assign every singleton subset of $\mathcal{X}$ to PLR_1. This completes the specification of our SET-COVER instance, which is valid thanks to the inclusion of the singletons.

Let $v_{p+1}, \ldots, v_1$ be the unique root-to-leaf path in $\mathcal{T}|_\pi$, with v_j being in layer j, for each j. Put $X_j = X_{v_j}$, for each j. By (E4), $\mathcal{X} = X_{p+1} \supseteq \cdots \supseteq X_1$, so the encodings of the pointers at $v_{p+1}, \ldots, v_2$ together cover $\bigcup_{j=2}^{p+1}(X_j \setminus X_{j-1}) = \mathcal{X} \setminus X_1$. Now suppose that $\text{MPJ}_{\mathcal{T}}(\pi) = 1$. Then the encoding of the leaf bits includes X_1, so $\mathcal{I}(\pi)$ has a set cover of size $Q_1 := p + 1$.

Next, suppose that $\text{MPJ}_{\mathcal{T}}(\pi) = 0$. A set cover must, in particular, cover X_1. However, the encodings of the pointers at $v_{p+1}, \ldots, v_2$ are all disjoint from X_1 and the encoding of the leaf bits does not include X_1. Therefore, X_1 must be covered using only singletons and sets corresponding to non-ancestors of v_1. For each such non-ancestor, y, the corresponding set in $\mathcal{I}(\pi)$ is a subset of the variety X_y. By (E6), such a set covers at most $d + p$ elements of X_1 whereas, by (E5), $|X_1| \geqslant q$. Therefore every set cover in $\mathcal{I}(\pi)$ uses least $Q_0 := q/(d + p)$ sets.

It follows that approximating even the optimum *value* of $\mathcal{I}(\pi)$ to a factor smaller than $Q_0/Q_1 = q/((p + d)(p + 1))$ is sufficient to determine $\text{MPJ}_{\mathcal{T}}(\pi)$.

Let $\mathcal{A}$ be a p-pass $\frac{1}{3}$-error randomised streaming algorithm that approximates SET-COVER this well, using at most s bits of space. The players can solve $\text{MPJ}_{\mathcal{T}}$ as follows. On input π, each player follows the above encoding scheme so that players jointly arrive at the SET-COVER instance $\mathcal{I}(\pi)$, with sets assigned amongst the players. They simulate the execution of $\mathcal{A}$ on the stream σ obtained by taking PLR_1's sets, followed by PLR_2's sets, and so on. Each time the execution of $\mathcal{A}$ moves off one player's portion of σ, that player broadcasts the memory contents of σ. This simulation uses one communication round per streaming pass, and spends sp bits of communication per round. Therefore it yields a $[p, sp, \frac{1}{3}]$-protocol for $\text{MPJ}_{\mathcal{T}}(\pi)$, whence $sp \geqslant \mathrm{R}^p(\text{MPJ}_{\mathcal{T}})$. ☐

3.4 Construction of an Edifice

Theorem 3.5. *Let k, d, and q be integers with $k \geqslant 1$, $d \geqslant 0$, and $q \geqslant d + k$, with q being a prime power. Then there exists a $(k, d, q, q^{d+k}(1 - 1/q))$-edifice.*

Proof. We shall construct an explicit edifice over the universe $\mathcal{X} = \mathbb{F}_q^k$. The varieties of our edifice will be certain well-structured varieties in the sense of algebraic geometry, i.e., solution sets of polynomial equations. Write the coordinates of a generic point in $\mathbb{F}_q^k$ as $(x, y_1, \ldots, y_{k-1})$. An *edificial equation* of *rank* i is defined to be an equation of the form

$$y_i = \ell_i(y_1, \ldots, y_{i-1}, f_{k-i}(x)), \qquad 1 \leqslant i \leqslant k-1, \tag{2}$$

where $\ell_i(z_1, \ldots, z_i)$ is a homogeneous linear form over $\mathbb{F}_q$ whose z_i-coefficient is nonzero and $f_j(x)$ is a monic polynomial in $\mathbb{F}_q[x]$ of degree exactly $d + j$. Equation (2) is abbreviated as $[\![\ell_i : f_{k-i}]\!]$.

Notice that irrespective of the value of i there are exactly $d + k$ coefficients appearing on the right-hand side of eq. (2), one of which must be nonzero. There are exactly $t := q^{d+k}(1 - 1/q)$ ways to choose these coefficients, leading to exactly t distinct edificial equations of each rank.

Let $\mathcal{T}$ be a rooted complete t-ary tree with k levels, the root r being at level k. For $1 \leqslant i \leqslant k-1$, for each level-$(i+1)$ vertex v of $\mathcal{T}$, label each of the t edges leaving v with one of the t distinct rank-i edificial equations. Associate a variety X_v with vertex v as follows. Let $X_r = \mathcal{X}$. If $v \neq r$, let X_v be the variety defined by the set of edificial equations labelling the edges on the path from r to v. We shall show that $\mathcal{T}$, with these associated varieties, forms a (k, d, q, t)-edifice. Certainly, properties (E1), (E2), (E3), and (E4) are immediate. The following observation will be helpful in establishing the remaining properties.

Observation 3.6. *Suppose* $\mathbf{x} = (x, y_1, \ldots, y_{k-1})$ *satisfies the edificial equations* $[\![\ell_1 : f_{k-1}]\!], \ldots, [\![\ell_j : f_{k-j}]\!]$ *for some* j *with* $1 \leqslant j \leqslant k-1$. *Then there exist linear forms* $\lambda_i(z_1, \ldots, z_i)$ *over* $\mathbb{F}_q$ *such that*

$$y_i = \lambda_i(f_{k-1}(x), \ldots, f_{k-i}(x)), \qquad 1 \leqslant i \leqslant j. \tag{3}$$

Therefore each of $y_1, \ldots, y_j$ *is determined by* x.

For the rest of this proof let z be a leaf; let $\mathbf{x} = (x, y_1, \ldots, y_{k-1}) \in X_z$ be an arbitrary point in the variety at z and let $[\![\ell_1 : f_{k-1}]\!], \ldots, [\![\ell_{k-1} : f_1]\!]$ be the edificial equations defining X_z. We record the following corollary of Observation 3.6.

Observation 3.7. *The point* $\mathbf{x}$ *is completely determined by its first coordinate* x.

It follows that for each $a \in \mathbb{F}_q$, X_z contains exactly one such point $\mathbf{x}$ with $x = a$, whence $|X_z| = |\mathbb{F}_q| = q$. This establishes property (E5).

Property (E6) requires a more careful examination of the form of the edificial equations. Consider a vertex v that is not an ancestor of the leaf z. Let u be the highest (by level) ancestor of v that is still not an ancestor of z. Since $X_u \supseteq X_v$, it suffices to prove that $|X_z \cap X_u| \leqslant d + k - 1$. Suppose u is at level $j < k$. Then X_u is defined by the $k - j - 1$ highest-ranked edificial equations that define X_z (which are of ranks $k - 1$ through $j + 1$) plus an additional rank-j equation $[\![\ell_j^+ : f_{k-j}^+]\!]$, where either $\ell_j \neq \ell_j^+$ or $f_{k-j} \neq f_{k-j}^+$, or both.

Suppose that $\ell_j = \ell_j^+$, so that $f_{k-j} \neq f_{k-j}^+$. Each point $\mathbf{x} = (x, y_1, \ldots, y_{k-1}) \in X_z \cap X_u$ must, in particular satisfy $[\![\ell_j : f_{k-j}]\!]$ and $[\![\ell_j : f_{k-j}^+]\!]$. Comparing these two equations gives

$$\begin{aligned} & \ell_j(y_1, \ldots, y_{j-1}, f_{k-j}(x)) = y_j = \ell_j(y_1, \ldots, y_{j-1}, f_{k-j}^+(x)) && (4) \\ \Rightarrow \quad & \ell_j(0, \ldots, 0, f_{k-j}(x) - f_{k-j}^+(x)) = 0 \\ \Rightarrow \quad & f_{k-j}(x) - f_{k-j}^+(x) = 0, && (5) \end{aligned}$$

because the linear form $\ell_j(z_1, \ldots, z_j)$ is required to have a nonzero z_j-coefficient. The left-hand side of eq. (5) is a nonzero univariate polynomial of degree at most $d + k - j$, whence it has at most $d + k - j$ roots in $\mathbb{F}_q$. By Observation 3.7, it follows that $|X_z \cap X_u| \leqslant d + k - j \leqslant d + k - 1$.

Finally, suppose $\ell_j \neq \ell_j^+$. We now make the crucial observation that

$$f_1(x), \ldots, f_{k-1}(x) \text{ are linearly independent over } \mathbb{F}_q\,, \tag{6}$$

which holds because these polynomials have distinct degrees. With this in mind, examining eqs. (2) and (3) and recalling that $\ell_i(z_1, \ldots, z_i)$ has a nonzero z_i-coefficient, we see that $\lambda_i(z_1, \ldots, z_i)$ also has a nonzero z_i-coefficient. Therefore, for each $i \in \{1, \ldots, k-1\}$, the collection of polynomials $\{\lambda_1(f_{k-1}(x)), \ldots, \lambda_i(f_{k-1}(x), \ldots, f_{k-i}(x))\}$ is a basis for the linear subspace of $\mathbb{F}_q[x]$ spanned by $\{f_{k-1}(x), \ldots, f_{k-i}(x)\}$.

Suppose $\mathbf{x} = (x, y_1, \ldots, y_{k-1}) \in X_z \cap X_u$. Proceeding as in eq. (4), we find that

$$\ell_j(y_1, \ldots, y_{j-1}, f_{k-j}(x)) = y_j = \ell_j^+(y_1, \ldots, y_{j-1}, f_{k-j}^+(x))\,.$$

Therefore there exists a linear form $h(z_1, \ldots, z_{j-1})$ and scalars $a, a^+ \in \mathbb{F}_q$, where either $h \neq 0$ or $a \neq a^+$ or both, such that $h(y_1, \ldots, y_{j-1}) + af_{k-j}(x) - a^+ f_{k-j}^+(x) = 0$. By Observation 3.6,

$$h(\lambda_1(f_{k-1}(x)), \ldots, \lambda_{j-1}(f_{k-1}(x), \ldots, f_{k-j+1}(x))) + af_{k-j}(x) - a^+ f_{k-j}^+(x) = 0\,. \tag{7}$$

We claim that the left-hand side of eq. (7) is a nonzero polynomial. If $h = 0$, this is immediate because $a \neq a^+$, whereas $f_{k-j}(x)$ and $f_{k-j}^+(x)$ are both monic of degree $d + k - j$. If $h \neq 0$, then by our observations about the polynomials $\{\lambda_i(f_{k-1}(x), \ldots, f_{k-i}(x))\}$, the first term on the left-hand side is a nonzero polynomial in the span of $\{f_{k-1}(x), \ldots, f_{k-j+1}(x)\}$. In particular, its degree is at least $d + k - j + 1$. The other two terms have degree at most $d + k - j$, which proves the claim.

Thus, eq. (7) states that x is a root of a nonzero polynomial of degree at most $d + k - 1$, a fact we derived from the condition that $\mathbf{x} \in X_z \cap X_u$. By Observation 3.7, $|X_z \cap X_u| \leqslant d + k - 1$. $\square$

Justifications for observations. For the sake of completeness, we formally justify the observations made in the course of the just-concluded proof. Observation 3.6 can be proved by induction on i. When $i = 1$, eq. (2) specialises to $y_1 = \ell_1(f_{k-1}(x))$, so we reach eq. (3) by taking $\lambda_1 = \ell_1$. For general i, by the induction hypothesis, we have

$$y_i = \ell_i(\lambda_1(f_{k-1}(x)), \ldots, \lambda_{i-1}(f_{k-1}(x), \ldots, f_{k-i+1}(x)), f_{k-i}(x))\,. \tag{8}$$

Each argument to ℓ_i in the above equation is a linear form in $\{f_{k-1}(x), \ldots, f_{k-i}(x)\}$, and ℓ_i is itself a linear form. Taking λ_i to be the "composition" of these linear forms gives us eq. (3).

Observation 3.7 is, as noted, a simple corollary to Observation 3.6.

We turn to the observation, made just after (6), that $\lambda_i(z_1, \ldots, z_i)$ has a nonzero z_i-coefficient. Of the i arguments to ℓ_i, only the last involves $f_{k-i}(x)$, and that last argument is given a nonzero coefficient by the defining property of ℓ_i. The other arguments are polynomials in the span of $\{f_{k-1}(x), \ldots, f_{k-i+1}(x)\}$. The linear independence observed in (6) completes the justification.

3.5 Pass/Approximation Tradeoff for Set Cover

We now bring together our technical results to obtain a pass/approximation tradeoff for SET-COVER in the semi-streaming setting.

Theorem 3.8 (Main result)**.** *Let $c > 1$ be a constant. Let $\mathcal{A}$ be a p-pass streaming algorithm that, for all large enough n and m, approximates the optimum value of* SET-COVER$_{n,m}$ *instances to a factor smaller than $n^{1/(p+1)}/(c(p+1)^2)$ with probability at least $2/3$. Then $\mathcal{A}$ must use $\Omega(n^c/p^3)$ bits of space. This space lower bound applies to instances with $m = \Theta(n^{cp})$.*

Proof. Let q be a sufficiently large prime power. Put $d = (c-1)(p+1)$, $t = q^{d+p+1}(1-1/q)$, and $n = q^{p+1}$. By Theorem 3.5, there exists a $(p+1, d, q, t)$-edifice over a universe $\mathcal{X}$ with $|\mathcal{X}| = n$. By Theorem 3.4, the space usage of $\mathcal{A}$, which approximates SET-COVER to a factor better than $n^{1/(p+1)}/((p+d)(p+1))$, is at least $\mathrm{R}^p(\mathrm{MPJ}_T)/p$, where T is a complete $(p+1)$-level t-ary tree. By Theorem 3.2, this space bound is $\Omega(t/p^3) = \Omega(n^{d/(p+1)+1}(1-1/q)/p^3) = \Omega(n^c/p^3)$.

Examining the reduction in Theorem 3.4 shows that instances of SET-COVER demonstrating the above lower bound have roughly as many sets as the edifice has leaves, i.e., $m = \Theta(n^{cp})$. □

It is instructive to note the following corollaries of Theorem 3.8.

1. Let p be a constant. Then there exist positive constants $\alpha < 1$ and $\beta > 1$ such that $(\alpha n^{1/(p+1)})$-approximating SET-COVER in p streaming passes requires $\Omega(n^\beta)$ space. In particular, such an approximation is not possible for a semi-streaming algorithm.
2. Every multi-pass semi-streaming $O(\log n)$-approximation algorithm for SET-COVER requires $p = \Omega(\log n/\log\log n)$ passes.

3.6 Two-Player Communication Complexity of Set Cover

Nisan [27] and Demaine et al. [10] have studied SET-COVER as a communication game. Our proof of Theorem 3.8 directly implies a lower bound for a certain *multi*-player SET-COVER game. But one may wonder about implications for the more fundamental setting of *two*-player communication complexity. Our next theorem shows that our technology does indeed yield a new two-player result.

In the two-player SET-COVER game, there is a fixed finite universe $\mathcal{X} = [n]$, Alice receives as input a collection $\mathcal{F} \subseteq 2^{\mathcal{X}}$, and Bob receives a collection $\mathcal{G} \subseteq 2^{\mathcal{X}}$. The players wish to solve the SET-COVER instance $(\mathcal{X}, \mathcal{F} \cup \mathcal{G})$ as cheaply as possible. Specifically, they must output a cover certificate (analogous to the array *Coverer* in Algorithm 1) that specifies, for each $x \in \mathcal{X}$, the set in $\mathcal{F} \cup \mathcal{G}$ that covers x. A communication protocol that gives such an output is said to be α-approximate if the implied set cover, *Sol*, satisfies $|Sol| \leqslant \alpha|Opt|$, where *Opt* is an optimum solution to the instance.

By mimicking the standard offline greedy algorithm for SET-COVER, one readily obtains a $(\ln n - \ln\ln n + \Theta(1))$-approximate protocol that communicates at most n messages, each message being n bits long; in particular, the total communication cost is $n^{O(1)}$. Nisan proved [27, Theorem 4] that for every constant $\delta > 0$, a $(\frac{1}{2} - \delta)\log_2 n$-approximate protocol requires an amount of communication that is exponentially larger, roughly $\exp(\sqrt{n})$ for small δ. Nisan's theorem uses a reduction from SET-DISJOINTNESS and is therefore agnostic about the number of messages in the protocol. Our theorem complements this by giving a "bounded-round" lower bound.

Theorem 3.9. *Let $c > 1$ be a constant. Suppose there exists a (randomised) α-approximate protocol for the two-player* SET-COVER *game that communicates a total of C bits in at most r messages. Then either $\alpha \geqslant n^{1/(r+1)}/(c(r+1)^2)$ or $C = \Omega(n^c/r^2)$.*

Proof sketch. We encode an instance of POINTER-JUMPING on a tree as a SET-COVER instance, using our edifices, exactly as in the proof of Theorem 3.4. We then treat POINTER-JUMPING as a *two*-player communication game, with Alice holding the information at vertices of the tree whose level is odd, and Bob holding the rest. For this two-player game, we invoke the bounded-round communication lower bound due to Klauck et al. [19] to finish the proof. □

While we could have used the above two-player version of POINTER-JUMPING as the basis for a data-streaming lower bound, it is important to note that doing so would have considerably weakened the streaming result, because p streaming passes translate into $2p-1$ messages in a two-player protocol.

4 Extension to Partial Cover

Thus far we have focused on the SET-COVER problem as traditionally defined, in which a feasible solution must cover the entire universe. However, as is the case with many optimisation problems, SET-COVER admits a relaxation in the form of a *bicriterial approximation*, wherein the feasibility constraint can be violated by some amount ε, and we seek a solution with cost at most $\alpha(\varepsilon, n)$ times the optimum fully feasible solution, for some function α.

To be precise, we consider the problem PARTIAL-COVER$_{n,m,\varepsilon}$, where an instance $\mathcal{I} = (\mathcal{X}, \mathcal{F}, \varepsilon)$ consists of a universe $\mathcal{X}$, with $|\mathcal{X}| = n$, a collection of sets $\mathcal{F} \subseteq 2^{\mathcal{X}}$ with $|\mathcal{F}| = m$, and a parameter $\varepsilon \in [0, 1]$. The goal is to compute a $(1 - \varepsilon)$-partial cover of $\mathcal{X}$, defined as a collection $Sol \subseteq \mathcal{F}$ that covers at least $(1 - \varepsilon)|\mathcal{X}|$ elements. Such a solution Sol is said to be α-approximate if $|Sol| \leqslant \alpha|Opt|$—or, in the weighted version, $w(Sol) \leqslant \alpha w(Opt)$—where Opt is a minimum-cost set cover for $(\mathcal{X}, \mathcal{F})$. Notice that we are comparing the cost of our partial cover with that of the best *total* cover.

4.1 Upper Bound

We begin with our most general upper bound, which includes partial covers and weighted sets. For convenience in stating the space bound, we assume that all weights are $O(\log m)$-bit integers.

Theorem 4.1. *For every integer $p \geqslant 1$, there is a p-pass, $O(n \log m)$-space algorithm for the* weighted *version of* PARTIAL-COVER$_{n,m,\varepsilon}$ *that produces an $\alpha(n, \varepsilon)$-approximate cost $(1 - \varepsilon)$-partial cover, where $\alpha(n, \varepsilon) = \min\{8p\varepsilon^{-1/p}, (8p + 1)n^{1/(p+1)}\}$.*

Proof. We run the following two schemes in parallel, returning the lower-cost solution. First, we run the Emek–Rosén algorithm for p passes, each time obtaining a $(1 - \varepsilon^{1/p})$-partial cover of the remaining (uncovered) portion of $\mathcal{X}$, and each time adding at most $8\varepsilon^{-1/p}w(Opt)$ cost to our solution Sol. By definition of a partial cover, for each $j \in [p]$, the collection of sets constituting Sol after j passes leaves at most $\varepsilon^{j/p}|\mathcal{X}|$ elements uncovered. Therefore, in the end, Sol is a $(1 - \varepsilon)$-partial cover.

Second, we run the Emek–Rosén algorithm for p passes (again) but here, in each pass, obtaining a $(1 - 1/n^{1/(p+1)})$-partial cover of the remaining (uncovered) portion of $\mathcal{X}$, and each time adding at most $8n^{1/(p+1)}w(Opt)$ cost to our solution Sol. The collection of sets constituting Sol after j passes leaves at most $n^{(p+1-j)/(p+1)}$ elements uncovered. After p passes, Sol covers all but at most $n^{1/(p+1)}$ elements. Covering each of these with its cheapest-covering set—which the Emek–Rosén algorithm records—of cost at most $w(Opt)$, leads to a total cost of at most $(8p + 1)n^{1/(p+1)}w(Opt)$. Since this is a full cover of $\mathcal{X}$, it is also a $(1 - \varepsilon)$-partial cover. □

The above upper bound generalises Theorem 2.5, except for the constant "8" that arises in the Emek–Rosén analysis. One can tweak their algorithm so as to replace the 8 with $(1 + \delta)^3$, where $\delta > 0$ is a constant of our choice, at the cost of increasing the space usage by a factor of $\Theta(1/\log(1 + \delta))$, which is about $\Theta(\delta^{-1})$ for small δ.

4.2 Lower Bound

We shall now show that Theorem 4.1 is asymptotically tight for every constant p by proving an appropriate lower bound on the approximation factor that a semi-streaming algorithm for PARTIAL-COVER can achieve. Our lower bound will hold even for unweighted PARTIAL-COVER and will match the upper bound of Theorem 4.1 up to a $\Theta(p^3)$ factor.

Our proof is based on edifices—as in the proof of Theorem 3.8—except that we need a different, more complicated, setting of parameters that is not directly achieved by Theorem 3.5. Instead, we revisit the edifices constructed in the proof of Theorem 3.5 and observe that they have an additional geometric property that we call *wideness*: roughly speaking, each level contains many groups of mutually parallel varieties. Clustering these parallel classes into "supervarieties" gives us new edifices with the desired parameters.

Let $\mathcal{C}_{\mathcal{T}}(u)$ denote the set of children of a vertex u in a tree $\mathcal{T}$. A (k,d,q,t)-edifice $\mathcal{T}$ is said to be (b,t')-*wide* if, for each non-leaf vertex u of $\mathcal{T}$, there exist subsets $\mathcal{V}_1,\ldots,\mathcal{V}_{t'} \subseteq \mathcal{C}_{\mathcal{T}}(u)$ such that

(W1) $\mathcal{V}_1,\ldots,\mathcal{V}_{t'}$ are pairwise disjoint;

(W2) for all $i \in [t']$, $|\mathcal{V}_i| = b$; and

(W3) for all $i \in [t']$, for all $v \neq v' \in \mathcal{V}_i$, we have $X_v \cap X_{v'} = \varnothing$.

Lemma 4.2. *If there exists a (k,d,q,t)-edifice $\mathcal{T}$ on universe $\mathcal{X}$ that is (b,t')-wide, then there exists a $(k, b^k(d+k-1), b^{k-1}q, t')$-edifice on the same universe $\mathcal{X}$.*

Proof. The desired edifice is built by "merging" certain carefully chosen sets of vertices of $\mathcal{T}$.

Define the following colour-and-trim procedure on a vertex u of $\mathcal{T}$. If u is a leaf, then do nothing. Otherwise, let $\mathcal{V}_1,\ldots,\mathcal{V}_{t'}$ be subsets of $\mathcal{C}_{\mathcal{T}}(u)$ satisfying (W1)–(W3). For each $i \in [t']$, for each $v \in \mathcal{V}_i$, assign colour i to the edge from u to v. Delete all uncoloured edges out of u as well as the subtrees pointed to by these edges. Then recursively colour-and-trim the remaining vertices in $\mathcal{C}_{\mathcal{T}}(u)$.

Let $\mathcal{T}'$ be the fully edge-coloured (bt')-ary tree obtained by applying this colour-and-trim procedure to r, the root of $\mathcal{T}$. Reusing the varieties from $\mathcal{T}$ makes $\mathcal{T}'$ a (k,d,q,bt')-edifice.

For each vertex v of $\mathcal{T}'$, define the *rainbow* at v to be the sequence of colours on the unique path from r to v. Create a new edge-coloured rooted tree $\mathcal{T}''$ by merging vertices of $\mathcal{T}'$ that have the same rainbow into "supervertices" and defining the parent of a supervertex v'' to be the vertex u'' whose rainbow is obtained by deleting the last colour in the rainbow at v''; assign this deleted colour to the edge from u'' to v''. Property (W2) implies that each $\mathcal{V}_i$ is nonempty; property (W1) then implies that $\mathcal{T}''$ is a t'-ary tree with k levels.

For each vertex u'' of $\mathcal{T}''$, let $\langle u'' \rangle$ denote the set of vertices of $\mathcal{T}'$ that were merged to produce u''. Define the variety $X_{u''} \subseteq \mathcal{X}$ thus:

$$X_{u''} = \biguplus_{u \in \langle u'' \rangle} X_u \,.$$

By (W3), the above union is indeed a *disjoint* union (denoted by "$\uplus$").

We shall show that $\mathcal{T}''$ is the desired edifice. Properties (E1), (E2), and (E3) are immediate. Property (E4) follows from the same property of $\mathcal{T}'$ and the observation that whenever two vertices of $\mathcal{T}'$ are merged in $\mathcal{T}''$, so are their parents. For property (E5), first note that (W2) implies that at each level $j \in [k]$ there are exactly b^{k-j} vertices of $\mathcal{T}'$ that have a particular rainbow. Thus, for each leaf z'' of $\mathcal{T}''$ we have $|\langle z'' \rangle| = b^{k-1}$. Using property (E5) of $\mathcal{T}'$, we have

$$|X_{z''}| = \left| \biguplus_{z \in \langle z'' \rangle} X_z \right| = \sum_{z \in \langle z'' \rangle} |X_z| \geqslant \sum_{z \in [z']} q = b^{k-1} q \,.$$

Finally, we address property (E6). As in the proof of Theorem 3.5, it suffices to upper-bound $|X_{z''} \cap X_{u''}|$, where z'' is a leaf of $\mathcal{T}''$ and u'' is a vertex of $\mathcal{T}''$ that is not an ancestor of z'', whereas the parent y'' of u'' is. Suppose that u'' is at level $j < k$. Then

$$X_{z''} \cap X_{u''} = \Bigg(\bigcup_{z \in \langle z'' \rangle} X_z \Bigg) \cap \Bigg(\bigcup_{u \in \langle u'' \rangle} X_u \Bigg) = \bigcup_{z \in \langle z'' \rangle,\, u \in \langle u'' \rangle} (X_z \cap X_u) \,. \tag{9}$$

Since $|\langle z'' \rangle| = b^{k-1}$ and $|\langle u'' \rangle| = b^{k-j}$, this latter expression immediately leads to $|X_{z''} \cap X_{u''}| \leqslant b^{2k-j-1}(d+k-1)$, using property (E6) of $\mathcal{T}'$. However, this upper bound is too weak; to strengthen it, we consider the structure of $X_{z''}$ and $X_{u''}$ more carefully.

Consider a generic $z \in \langle z'' \rangle$ and a generic $u \in \langle u'' \rangle$. There must exist $y_1, y_2 \in \langle y'' \rangle$ such that z is a descendant of y_1 and u is a descendant of y_2. The crucial observation is that if $y_1 \neq y_2$, then by (W3), $X_{y_1} \cap X_{y_2} = \varnothing$, whence by (E4), $X_z \cap X_u = \varnothing$. Therefore the pair (z, u) contributes to the latter union in eq. (9) only when $y_1 = y_2$. Therefore,

$$X_{z''} \cap X_{u''} = \bigcup_{y \in \langle y'' \rangle} \ \bigcup_{\substack{z \in \langle z'' \rangle,\, u \in \langle u'' \rangle \\ z, u \text{ descendants of } y}} (X_z \cap X_u) \,.$$

Since $|\langle y'' \rangle| = b^{k-j-1}$ and each $y \in \langle y'' \rangle$ has b^j descendants in $\langle z'' \rangle$ and b descendants in $\langle u'' \rangle$, we obtain $|X_{z''} \cap X_{u''}| \leqslant b^{k-j-1} b^j b (d+k-1) = b^k(d+k-1) \leqslant b^k(d+k-1)+k-1$, as required. □

Lemma 4.3. *The (k, d, q, t)-edifice constructed in Theorem 3.5 is $(\lfloor \delta q \rfloor, \lfloor 1/\delta \rfloor t/q)$-wide for all $\delta \in (0, 1]$.*

Proof. It suffices to prove the lemma in the case $\delta = 1$; a little thought shows that the general case then follows as a corollary.

Let $\mathcal{T}$ be the edifice constructed in Theorem 3.5. Let u be a non-leaf vertex of $\mathcal{T}$, at level $j+1$, where $j \in [k-1]$. Then the edges out of u are labelled by the t distinct rank-j edificial equations. Let us call two such equations $[\![\ell_j : f_{k-j}]\!]$ and $[\![\ell_j^+ : f_{k-j}^+]\!]$ *similar* if $\ell_j = \ell_j^+$ and $f_{k-j} - f_{k-j}^+$ is a constant polynomial. This similarity relation then naturally extends to $\mathcal{C}_{\mathcal{T}}(u)$. Similarity is easily seen to be an equivalence relation, each of whose equivalence classes has size exactly $|\mathbb{F}_q| = q$. Therefore there are exactly t/q equivalence classes; let $\mathcal{V}_1, \ldots, \mathcal{V}_{t/q} \subseteq \mathcal{C}_{\mathcal{T}}(u)$ be these classes.

To show that $\mathcal{T}$ is $(q, t/q)$-wide, we shall show that these classes $\{\mathcal{V}_i\}$ satisfy properties (W1), (W2), and (W3). The first two properties are immediate. For the third, consider arbitrary $v \neq v' \in \mathcal{V}_i$, for some i. Then v and v' are similar, which means that a point $\mathbf{x} = (x, y_1, \ldots, y_{k-1}) \in X_v \cap X_{v'}$ must satisfy a pair of similar, but distinct, edificial equations. Let these equations be $[\![\ell_j : f_{k-j}]\!]$ and $[\![\ell_j : f_{k-j}^+]\!]$. Consulting eq. (2), we find that

$$0 = y_j - y_j = \ell_j(y_1, \ldots, y_j, f_{k-j}(x)) - \ell_j(y_1, \ldots, y_j, f_{k-j}^+(x)) = \ell_j(0, \ldots, 0, f_{k-j}(x) - f_{k-j}^+(x)) \,.$$

By definition, the linear form $\ell_j(z_1, \ldots, z_j)$ has a nonzero z_j-coefficient, implying that $f_{k-j}(x) - f_{k-j}^+(x) = 0$. This is a contradiction, because $f_{k-j} - f_{k-j}^+$ is a *nonzero* constant polynomial. Therefore such a point $\mathbf{x}$ does not exist, i.e., $X_v \cap X_{v'} = \varnothing$. □

Theorem 4.4. *Let $c > 1$ be a constant. Let $\mathcal{A}$ be a p-pass streaming algorithm with the following guarantee. For all large enough n and m and all $\varepsilon \in (0, \frac{1}{2}]$, for all instances of* PARTIAL-COVER$_{n,m,\varepsilon}$, *with probability at least $2/3$, $\mathcal{A}$ returns the* value *of some α-approximate solution to the instance, where*

$$\alpha < \frac{\min\{n^{1/(p+1)}, \varepsilon^{-1/p}\}}{8c(p+1)^2} \,. \tag{10}$$

Then $\mathcal{A}$ must use $\Omega(n^c/p^3)$ bits of space. In particular $\mathcal{A}$ cannot be semi-streaming.

Proof. This theorem is analogous to a combination of Theorems 3.4 and 3.8; the proof is along very similar lines.

We may as well assume that $\varepsilon^{-1/p} \leqslant n^{1/(p+1)}$, because if ε is too small for this to hold, then we simply consider the weaker problem of $(1-\varepsilon')$-partial covering, where $\varepsilon' = n^{-p/(p+1)}$.

Pick a sufficiently large prime power q. Put $n = q^{p+1}$, $d = (c-1)(p+1)+1$, $\widetilde{\delta} = (2\varepsilon)^{1/p}$, and $\delta = \lceil \widetilde{\delta} q \rceil / q$. By our assumption, we have $\widetilde{\delta} \geqslant 1/q$ and $\widetilde{\delta} \leqslant \delta \leqslant 2\widetilde{\delta}$.

Combining Theorem 3.5 with Lemmas 4.2 and 4.3 and working through some algebra, we find that there exists a $(p+1, (\delta q)^{p+1}(d+p), (\delta q)^p q, \lfloor 1/\delta \rfloor q^{d+p}(1-1/q))$-edifice $\mathcal{T}$ over a universe $\mathcal{X}$ with $|\mathcal{X}| = n$. Using the varieties of $\mathcal{T}$, we encode each instance π of $\text{MPJ}_{\mathcal{T}}$ as a collection $\mathcal{I}(\pi)$ of subsets of $\mathcal{X}$ exactly as in Theorem 3.4 and treat $\mathcal{I}(\pi)$ as an instance of $\text{PARTIAL-COVER}_{n,m,\varepsilon}$. As before, if $\text{MPJ}_{\mathcal{T}}(\pi) = 1$, then $\mathcal{I}(\pi)$ admits a total cover using $Q_1 := p+1$ sets.

For the case $\text{MPJ}_{\mathcal{T}}(\pi) = 0$, we refine the argument used for Theorem 3.4 as follows. Let X_1 be the variety of $\mathcal{T}$ at the unique leaf, v_1, in $\mathcal{T}|_\pi$. As before, the elements of X_1 cannot be covered by sets corresponding to ancestors of v_1, and each of the remaining sets in $\mathcal{I}(\pi)$ can cover at most $(\delta q)^{p+1}(d+p)$ such elements. Every $(1-\varepsilon)$-partial cover must, in particular, cover at least $|X_1| - \varepsilon|\mathcal{X}|$ elements of X_1. It follows that the cheapest such partial cover uses at least $Q_0 := (|X_1| - \varepsilon|\mathcal{X}|)/((\delta q)^{p+1}(d+p))$ sets. Now,

$$\frac{Q_0}{Q_1} = \frac{|X_1| - \varepsilon|\mathcal{X}|}{(\delta q)^{p+1}(p+d)(p+1)} \geqslant \frac{(\delta q)^p q - \varepsilon q^{p+1}}{(\delta q)^{p+1}(p+d)(p+1)} \tag{11}$$

$$= \frac{(\delta q)^p q - \frac{1}{2}\widetilde{\delta}^p q^{p+1}}{(\delta q)^{p+1} c (p+1)^2}$$

$$\geqslant \frac{1}{2\delta c (p+1)^2} \tag{12}$$

$$\geqslant \frac{1}{4(2\varepsilon)^{1/p} \cdot c(p+1)^2} \geqslant \frac{\varepsilon^{-1/p}}{8c(p+1)^2}, \tag{13}$$

where (11) uses the parameters of the edifice $\mathcal{T}$, (12) uses $\widetilde{\delta} \leqslant \delta$, and (13) uses $\delta \leqslant 2\widetilde{\delta}$.

Therefore, eq. (10) gives $\alpha < Q_0/Q_1$. As in Theorem 3.4, with an approximation this good, $\mathcal{A}$ can be used to determine $\text{MPJ}_{\mathcal{T}}(\pi)$ and must consequently use $\Omega(t/p^3)$ bits of space, where t is the arity of $\mathcal{T}$. Since $t = \lfloor 1/\delta \rfloor q^{d+p}(1-1/q) = \Omega(n^c)$, this space lower bound is $\Omega(n^c/p^3)$. □

5 Discussion

We conclude with a more technically detailed description of selected results from previous work, with the goal of shedding more light on some of our own results.

In the external-memory setting, without a streaming restriction, an *eager* implementation of the greedy algorithm involves an inverted index and a priority queue of set sizes. Unfortunately, this involves arbitrary (non-local) memory accesses, leading to poor performance.

Relaxing the strict greedy requirement, Cormode, Karloff, and Wirth add a set to the solution if its contribution is at least $1/\beta$ times the best [9]. So that all disk accesses are sequential, initially they allocate sets to "buckets" (files) according to their size, with a bucket for each range $[\beta^j, \beta^{j+1})$, $i = 0, \ldots, \kappa$, where $\kappa = \max\lfloor \log_\beta |S_i| \rfloor$. Starting from $j = \kappa$ down to 0, as each set in bucket j is examined, sequentially, set S_i is added to *Sol* only if its contribution is at least β^j; otherwise, $(i, S_i \setminus C)$ is appended to the appropriate bucket. This is essentially the same thresholding as Algorithm 1, with the same pass/approximation tradeoff, but implemented so that the total amount of data handled is $O(\beta/(\beta-1))$ times the input size.

Blelloch, Simhadri, and Tangwongsan solve very large set cover instances on disk and in parallel in RAM [5]. They consider situations in which there is less than one word of memory per element. Their pre-bucketing is much like the geometric ranges of DFG, and their MaNIS scheme appears to be a randomised, and parallelisable, version of the pass through the sets in a bucket.

The Emek–Rosén scheme [12] is in some sense like DFG in its having a hierarchy of thresholds that are powers of 2. Its purpose however, is to facilitate partial covers with (item and) set costs. In the unweighted setting, as each set S is seen, it is deemed to cover some subset $T \subseteq S$, where $2^i \leqslant |T| < 2^{i+1}$, if each element in T was previously covered by some subset of size $< 2^i$, or was previously uncovered. This is somewhat like all the runs of DFG with $\beta = 2$ being folded into one. In parallel, the scheme records the cheapest set that covers each item (amongst equal-cheapest, choose the first that occurs in the stream). This step is similar to the folding in Algorithm 2.

We contrast the threshold chosen in our algorithm with that in the Emek–Rosén algorithm. In our two-pass algorithm (folded into one), $\tau = \sqrt{n}$, leading to a $2\sqrt{n}$ approximation (in fact, $|Sol| \leqslant \sqrt{n}(1 + |Opt|)$). Once the stream is done, the Emek–Rosén algorithm can choose a threshold $\tau = 2^i$. Items that are recorded as covered by some such $T \subseteq S$, with $|T| \geqslant \tau$, are certified to be covered by S; those "below the threshold" are instead covered by their cheapest set. This way, at most $O(n/\tau)$ T-sets are chosen and $O(\tau w(Opt))$ elements are cheapest-set covered. Since such a cheapest set has cost at most $w(Opt)$, by setting this threshold τ to be approximately $\varepsilon n / w(Opt)$, the algorithm returns an $O(w(Opt)/\varepsilon)$ weight solution. Of course, we do not know $w(Opt)$, but it suffices to choose the largest τ so that at most εn elements are cheapest-set covered. When $\varepsilon \leqslant 1/\sqrt{n}$ however, it is better to choose τ to leave at most $\sqrt{n}$ cheapest-set covered elements, hence $\tau = \Theta(\sqrt{n}/w(Opt))$.

This tradeoff allows the Emek–Rosén algorithm to account for set weights. In the unweighted case, however, our solution has at most $\sqrt{n}(1 + |Opt|)$ sets, whereas the Emek–Rosén solution has at most $\sqrt{n}(1 + 8|Opt|)$ sets. As mentioned in Section 4.1, the latter expression can become arbitrarily close, i.e., $\sqrt{n}(1 + (1 + \delta)^3|Opt|)$, with space increasing by a factor of $O(1/\delta)$.

Acknowledgment

The second author is grateful to Andrew McGregor for discussions about some recent work.

References

[1] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The online set cover problem. In *Proc. 35th Annual ACM Symposium on the Theory of Computing*, pages 100–105, 2003.

[2] A. Anagnostopoulos, L. Becchetti, I. Bordino, S. Leonardi, I. Mele, and P. Sankowski. Stochastic query covering for fast approximate document retrieval. *ACM Trans. Inf. Syst.*, 33(3):11:1–11:35, 2015.

[3] S. Arora and B. Barak. *Complexity Theory: A Modern Approach*. Cambridge University Press, Cambridge, 2009.

[4] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. 21st ACM Symposium on Principles of Database Systems*, pages 1–16, 2002.

[5] G. E. Blelloch, H. V. Simhadri, and K. Tangwongsan. Parallel and I/O efficient set covering algorithms. In *Proc. 24th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 82–90, 2012.

[6] F. Buekenhout. Diagrams for geometries and groups. *J. Combin. Theory Ser. A*, 27(2):121–151, 1979.

[7] A. Chakrabarti, G. Cormode, R. Kondapally, and A. McGregor. Information cost tradeoffs for augmented index and streaming language recognition. In *Proc. 51st Annual IEEE Symposium on Foundations of Computer Science*, pages 387–396, 2010.

[8] A. Chakrabarti, G. Cormode, and A. McGregor. Robust lower bounds for communication and stream computation. In *Proc. 40th Annual ACM Symposium on the Theory of Computing*, pages 641–649, 2008.

[9] G. Cormode, H. J. Karloff, and A. Wirth. Set cover algorithms for very large datasets. In *Proc. 19th ACM Conference on Information and Knowledge Management*, pages 479–488, 2010.

[10] E. D. Demaine, P. Indyk, S. Mahabadi, and A. Vakilian. On streaming and communication complexity of the set cover problem. In *Proc. 28th International Symposium on Distributed Computing*, pages 484–498. Springer, 2014.

[11] I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Proc. 46th Annual ACM Symposium on the Theory of Computing*, pages 624–633, 2014.

[12] Y. Emek and A. Rosén. Semi-streaming set cover. In *Proc. 41st International Colloquium on Automata, Languages and Programming*, pages 453–464, 2014.

[13] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998. Preliminary version in *Proc. 28th Annual ACM Symposium on the Theory of Computing*, pages 314–318, 1996.

[14] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2–3):207–216, 2005. Preliminary version in *Proc. 31st International Colloquium on Automata, Languages and Programming*, pages 531–543, 2004.

[15] L. Golab, H. Karloff, F. Korn, D. Srivastava, and B. Yu. On generating near-optimal tableaux for conditional functional dependencies. *Proc. VLDB Endowment*, 1(1):376–390, 2008.

[16] S. Guha and A. McGregor. Lower bounds for quantile estimation in random-order and multi-pass streaming. In *Proc. 34th International Colloquium on Automata, Languages and Programming*, pages 704–715, 2007.

[17] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974.

[18] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, NY, USA, 1972.

[19] H. Klauck, A. Nayak, A. Ta-Shma, and D. Zuckerman. Interaction in quantum communication and the complexity of set disjointness. In *Proc. 33rd Annual ACM Symposium on the Theory of Computing*, pages 124–133, 2001.

[20] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, Cambridge, 1997.

[21] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.

[22] F. Magniez, C. Mathieu, and A. Nayak. Recognizing well-parenthesized expressions in the streaming model. In *Proc. 41st Annual ACM Symposium on the Theory of Computing*, pages 261–270, 2010.

[23] P. B. Miltersen, N. Nisan, S. Safra, and A. Wigderson. On data structures and asymmetric communication complexity. *J. Comput. Syst. Sci.*, 57(1):37–49, 1998. Preliminary version in *Proc. 27th Annual ACM Symposium on the Theory of Computing*, pages 103–111, 1995.

[24] D. Moshkovitz. The projection games conjecture and the NP-hardness of ln n-approximating set-cover. In *Proc. 15th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 276–287, 2012.

[25] J. I. Munro and M. Paterson. Selection and sorting with limited storage. *TCS*, 12:315–323, 1980. Preliminary version in *Proc. 19th Annual IEEE Symposium on Foundations of Computer Science*, pages 253–258, 1978.

[26] S. Muthukrishnan. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2):117–236, 2005.

[27] N. Nisan. The communication complexity of approximate set packing and covering. In *Proc. 29th International Colloquium on Automata, Languages and Programming*, pages 868–875, 2002.

[28] B. Saha and L. Getoor. On maximum coverage in the streaming model & application to multi-topic blog-watch. In *Proc. 9th SIAM International Conference on Data Mining*, pages 697–708, 2009.

[29] P. Slavík. A tight analysis of the greedy algorithm for set cover. In *Proc. 28th Annual ACM Symposium on the Theory of Computing*, pages 435–441, 1996.

[30] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2003.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de