Towards a Theory of Parameterized Streaming Algorithms *

Rajesh Chitnis and Graham Cormode
University of Warwick, UK.
rajeshchitnis@gmail.com, g.cormode@warwick.ac.uk

Abstract

Parameterized complexity attempts to give a more fine-grained analysis of the complexity of problems: instead of measuring the running time as a function of only the input size, we analyze the running time with respect to additional parameters. This approach has proven to be highly successful in delineating our understanding of NP-hard problems. Given this success with the TIME resource, it seems but natural to use this approach for dealing with the SPACE resource. First attempts in this direction have considered a few individual problems, with some success: Fafianie and Kratsch [MFCS'14] and Chitnis et al. [SODA'15] introduced the notions of streaming kernels and parameterized streaming algorithms respectively. For example, the latter shows how to refine the $\Omega(n^2)$ bit lower bound for finding a minimum Vertex Cover (VC) in the streaming setting by designing an algorithm for the parameterized k-VC problem which uses $O(k^2 \log n)$ bits.

In this paper, we initiate a systematic study of graph problems from the paradigm of parameterized streaming algorithms. We first define a natural hierarchy of space complexity classes of `FPS`, `SubPS`, `SemiPS`, `SupPS` and `BrutePS`, and then obtain tight classifications for several well-studied graph problems such as Longest Path, Feedback Vertex Set, Dominating Set, Girth, Treewidth, etc. into this hierarchy (see Figure 1 and Figure 2). On the algorithmic side, our parameterized streaming algorithms use techniques from the FPT world such as bidimensionality, iterative compression and bounded-depth search trees. On the hardness side, we obtain lower bounds for the parameterized streaming complexity of various problems via novel reductions from problems in communication complexity. We also show a general (unconditional) lower bound for space complexity of parameterized streaming algorithms for a large class of problems inspired by the recently developed frameworks for showing (conditional) kernelization lower bounds.

Parameterized algorithms and streaming algorithms are approaches to cope with TIME and SPACE intractability respectively. It is our hope that this work on parameterized streaming algorithms leads to two-way flow of ideas between these two previously separated areas of theoretical computer science.

2012 ACM Subject Classification Theory of Computation → Design and analysis of algorithms

Keywords and phrases Parameterized Algorithms, Streaming Algorithms, Kernels

Digital Object Identifier 10.4230/LIPIcs.IPEC.2019.

Acknowledgements We thank MohammadTaghi Hajiaghayi, Robert Krauthgamer and Morteza Monemizadeh for helpful discussions. Algorithm 1 was suggested to us by Arnold Filtser.

* Supported by ERC grant 2014-CoG 647557.

1 Introduction

Designing and implementing efficient algorithms is at the heart of computer science. Traditionally, efficiency of algorithms has been measured with respect to running time as a function of instance size. From this perspective, algorithms are said to be efficient if they can be solved in time which is bounded by some polynomial function of the input size. However, very many interesting problems are NP-complete, and so are grouped together as "not known to be efficient". This fails to discriminate within a large heterogenous group of problems, and in response the theory of *parameterized (time) algorithms* was developed in late 90's by Downey and Fellows [25]. Parameterized complexity attempts to delineate the complexity of problems by expressing the costs in terms of additional parameters. Formally, we say that a problem is *fixed-parameter tractable* (FPT) with respect to parameter k if the problem can be solved in time $f(k) \cdot n^{O(1)}$ where f is a computable function and n is the input size. For example, the problem of checking if a graph on n vertices has a vertex cover of size at most k can be solved in $2^k \cdot n^{O(1)}$ time. The study of various parameters helps to understand which parameters make the problem easier (FPT) and which ones cause it to be hard. The parameterized approach towards NP-complete problems has led to development of various algorithmic tools such as kernelization, iterative compression, color coding, and more [26, 19].

Kernelization: A key concept in fixed parameter tractability is that of kernelization which is an efficient preprocessing algorithm to produce a smaller, equivalent output called the "kernel". Formally, a kernelization algorithm for a parameterized problem Q is an algorithm which takes as an instance $\langle x, k\rangle$ and outputs in time polynomial in $(|x| + k)$ an equivalent[1] instance $\langle x', k'\rangle$ such that $\max\{|x'|, k'\} \leq f(k)$ for some computable function f. The output instance $\langle x', k'\rangle$ is called the kernel, while the function f determines the size of the kernel. Kernelizability is equivalent to fixed-parameter tractability, and designing compact kernels is an important question. In recent years, (conditional) lower bounds on kernels have emerged [5, 21, 22, 27, 33].

Streaming Algorithms: A very different paradigm for handling large problem instances arises in the form of streaming algorithms. The model is motivated by sources of data arising in communication networks and activity streams that are considered to be too big to store conveniently. This places a greater emphasis on the space complexity of algorithms. A streaming algorithm processes the input in one or a few read-only passes, with primary focus on the storage space needed. In this paper we consider streaming algorithms for graph problems over fixed vertex sets, where information about the edges arrives edge by edge [35]. We consider variants where edges can be both inserted and deleted, or only insertions are allowed. We primarily consider single pass streams, but also give some multi-pass results.

1.1 Parameterized Streaming Algorithms and Kernels

Given that parameterized algorithms have been extremely successful for the TIME resource, it seems natural to also use it attack the SPACE resource. In this paper, we advance the model of parameterized streaming algorithms, and start to flesh out a hierarchy of complexity classes. We focus our attention on graph problems, by analogy with FPT, where the majority of results have addressed graphs. From a space perspective, there is perhaps less headroom than when considering the time cost: for graphs on n vertices, the entire graph can be stored

[1] By equivalent we mean that $\langle x, k\rangle \in Q \Leftrightarrow \langle x', k'\rangle \in Q$

using $O(n^2)$ space[2]. Nevertheless, given that storing the full graph can be prohibitive, there are natural space complexity classes to consider. We formalize these below, but informally, the classes partition the dependence on n as: (i) (virtually) independent of n; (ii) sublinear in n; (iii) (quasi)linear in n; (iv) superlinear but subquadratic in n; and (v) quadratic in n.

Naively, several graph problems have strong lower bounds: for example, the problem of finding a minimum vertex cover on graphs of n vertices has a lower bound of $\Omega(n^2)$ bits. However, when we adopt the parameterized view, we seek streaming algorithms for (parameterized) graph problems whose space can be expressed as a function of *both* the number of vertices n and the parameter k. With this relaxation, we can separate out the problem space and start to populate our hierarchy. We next spell out our results, which derive from a variety of upper and lower bounds building on the streaming and FPT literature.

1.2 Our Results & Organization of the paper

For a graph problem with parameter k, there can be several possible choices for the space complexity needed to solve it in the streaming setting. In this paper, we first define some natural space complexity classes below:

1. *$\widetilde{O}(f(k))$ space:* Due to the connection to running time of FPT algorithms, we call the class of parameterized problems solvable using $\widetilde{O}(f(k))$ bits as FPS (fixed-parameterized streaming)[3].
2. *Sublinear space:* When the dependence on n is sublinear, we call the class of parameterized problems solvable using $\widetilde{O}(f(k) \cdot n^{1-\epsilon})$ bits as SubPS (sublinear parameterized streaming)
3. *Quasi-linear space:* Due to the connection to the semi-streaming model [31, 40], we call the set of problems solvable using $\widetilde{O}(f(k) \cdot n)$ bits as SemiPS (parameterized semi-streaming).
4. *Superlinear, subquadratic space:* When the dependence on n is superlinear (but subquadratic), we call the class of parameterized problems solvable using $\widetilde{O}(f(k) \cdot n^{1+\epsilon})$ bits (for some $1 > \epsilon > 0$) as SupPS (superlinear parameterized streaming).
5. *Quadratic space:* We call the set of graph problems solvable using $O(n^2)$ bits as BrutePS (brute-force parameterized streaming). Note that every graph problem is in BrutePS since we can just store the entire adjacency matrix using $O(n^2)$ bits (see Remark 2).

▶ Remark 1. Formally, we need t consider the following 7-tuple when we attempt to find its correct position in the aforementioned hierarchy of complexity classes:

[Problem, Parameter, Space,# of Passes, Type of Algorithm, Approx. Ratio, Type of Stream]

By type of algorithm, we mean that the algorithm could be deterministic or randomized. For the type of stream, the standard alternatives are (adversarial) insertion, (adversarial) insertion-deletion, random order, etc. Figure 3 gives a list of results for the k-VC problem (as a case study) in various different settings. Unless stated otherwise, throughout this paper, we consider the space requirement for 1-pass exact deterministic algorithms for problems with the standard parameter (size of the solution)on insertion-only streams.

▶ Remark 2. There are various different models for streaming algorithms depending on how much computation is allowed on the stored data. In this paper, we consider the most general

[2] Throughout the paper, by space we mean words/edges/vertices. Each word can be represented using $O(\log n)$ bits

[3] Throughout this paper, we use the $\tilde{O}$ notation to hide $\log^{O(1)} n$ factors

model by allowing *unbounded computation* at each edge update, and also at the end of the stream.

Our goal is to provide a tight classification of graph problems into the aforementioned complexity classes. We make progress towards this goal as follows: Section 2 shows how various techniques from the FPT world such as iterative compression, branching, bidimensionality, etc. can also be used to design parameterized streaming algorithms. First we investigate whether one can further improve upon the FPS algorithm of Chitnis et al. [13] for k-VC which uses $O(k^2 \cdot \log n)$ bits and one pass. We design two algorithms for k-VC which use $O(k \cdot \log n)$ bits[4]: an 2^k-pass algorithm using bounded-depth search trees (Section 2.1) and an $(k \cdot 2^{2k})$-pass algorithm using iterative compression (Section 2.2). Finally, Section 2.3 shows that any minor-bidimensional problem belongs to the class SemiPS.

Section 3 deals with lower bounds for parameterized streaming algorithms. First, in Section 3.1 we show that some parameterized problems are tight for the classes SemiPS and BrutePS. In particular, we show that k-Treewidth, k-Path and k-Feedback-Vertex-Set are tight for the class SemiPS, i.e., they belong to SemiPS but do not belong to the sub-class SubPS. Our SemiPS algorithms are based on problem-specific structural insights. Via reductions from the PERM problem [45], we rule out algorithms which use $\widetilde{O}(f(k) \cdot n^{1-\epsilon})$ bits (for any function f and any $\epsilon \in (0,1)$) for these problems by proving $\Omega(n \log n)$ bits lower bounds for constant values of k. Then we show that some parameterized problems such as k-Girth and k-Dominating-Set are tight for the class BrutePS, i.e, they belong to BrutePS but do not belong to the sub-class SupPS. Every graph problem belongs to BrutePS since we can store the entire adjacency matrix of the graph using $O(n^2)$ bits. Via reductions from the INDEX problem [37], we rule out algorithms which use $\widetilde{O}(f(k) \cdot n^{1+\epsilon})$ bits (for any function f and any $\epsilon \in (0,1)$) for these problems by proving $\Omega(n^2)$ bits lower bounds for constant values of k.

Section 3.2 shows a lower bound of $\Omega(n)$ bits for any algorithm that approximates (within a factor $\frac{\beta}{32}$) the size of min dominating set on graphs of arboricity $(\beta + 2)$, i.e., this problem has no $\widetilde{O}(f(\beta) \cdot n^{1-\epsilon})$ bits algorithm (since β is a constant), and hence does not belong to the class SubPS when parameterized by β. In Section 3.3 we obtain unconditional lower bounds on the space complexity of 1-pass parameterized streaming algorithms for a large class of graph problems inspired by some of the recent frameworks to show conditional lower bounds for kernels [5, 21, 22, 27, 33]. Finally, in Section E we show that any parameterized streaming algorithm for the d-SAT problem (for any $d \geq 2$) must (essentially) follow the naive algorithm of storing all the clauses.

Figure 1 provides a pictorial representation of the complexity classes, and the known classification of several graph problems (from this paper and some previous work) into these classes. Figure 2 summarizes our results, and clarifies the stream arrival model(s) under which they hold.

1.3 Prior work on Parametrized Streaming Algorithms

Prior work began by considering how to implement kernels in the streaming model. Formally, a streaming kernel [30] for a parameterized problem (I, k) is a streaming algorithm that receives the input I as a stream of elements, stores $f(k) \cdot \log^{O(1)} |I|$ bits and returns an equivalent instance[5]. This is especially important from the practical point of view since

[4] Which is essentially optimal since the algorithm also returns a VC of size k (if one exists)
[5] [30] required $f(k) = k^{O(1)}$, but we choose to relax this requirement

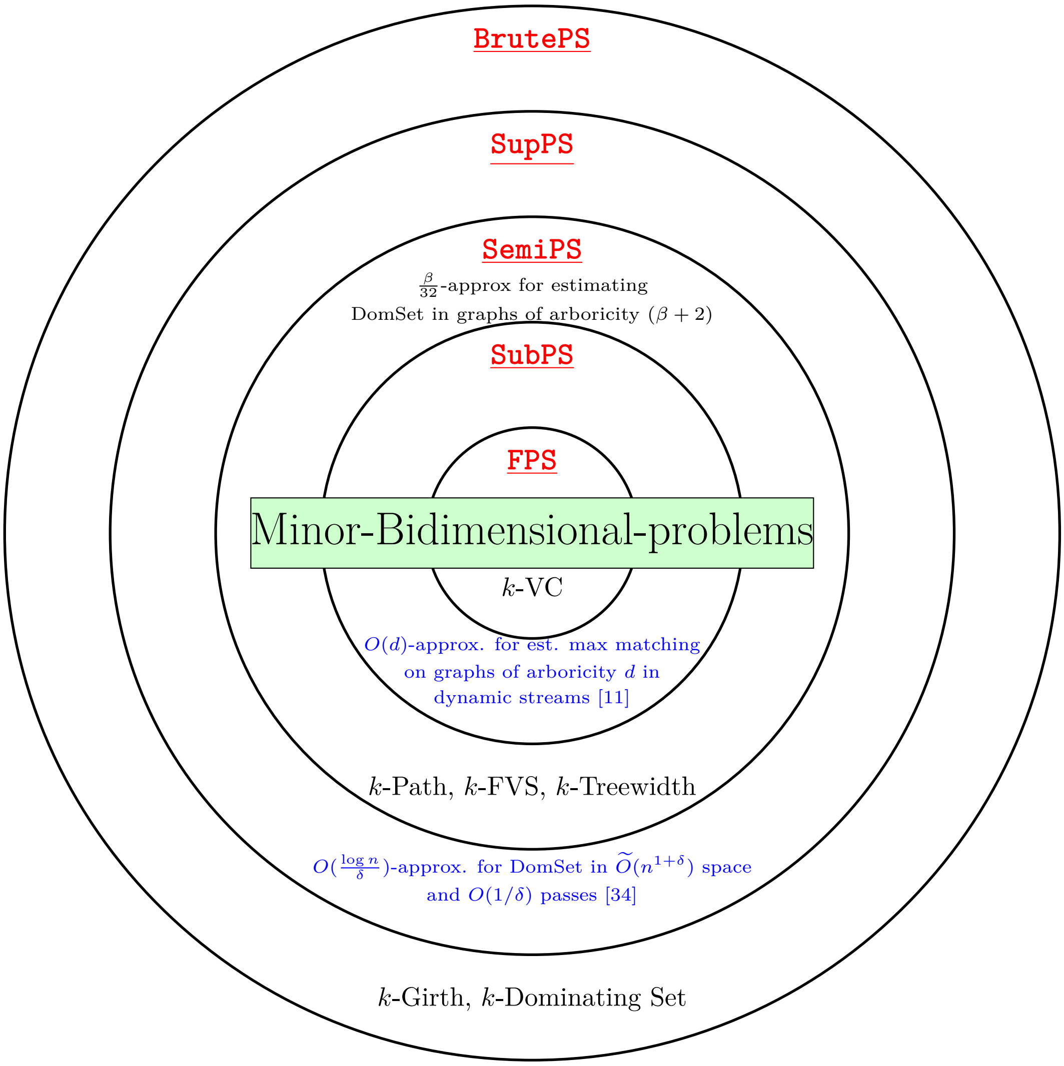

Figure 1 Pictorial representation of classification of some graph problems into complexity classes: our results are in black and previous work is referenced in blue. All results are for 1-pass deterministic algorithms on insertion-only streams unless otherwise specified. It was already known that k-VC $\in$ FPS [13, 11] using only 1-pass, but here we design an algorithm with optimal space storage at the expense of multiple passes.

several real-world situations can be modeled by the streaming setting, and streaming kernels would help to efficiently preprocess these instances. Fafianie and Kratsch [30] showed that the kernels for some problems like Hitting Set and Set Matching can be implemented in the streaming setting, but other problems such as Edge Dominating Set, Feedback Vertex Set, etc. do not admit (1-pass) streaming kernels.

Chitnis et al. [13] studied how to circumvent the worst case bound of $\Omega(n^2)$ bits for Vertex Cover by designing a streaming algorithm for the parameterized k-Vertex-Cover (k-VC)[6]. They showed that the k-VC problem can be solved in insertion-only streams using storage

[6] That is, determine whether there is a vertex cover of size at most k?

Problem	Number of Passes	Type of Stream	Space Upper Bound	Space Lower Bound
$g(r)$-minor-bidimensional problems [Sec. 2.3]	1	Ins-Del.	$\widetilde{O}((g^{-1}(k+1))^{10}n)$ words	—
k-VC [Sec. 2.2]	$2^{2k} \cdot k$	Ins-only	$O(k)$ words	$\Omega(k)$ words
k-VC [Sec. 2.1]	2^k	Ins-only	$O(k)$ words	$\Omega(k)$ words
k-FVS, k-Path k-Treewidth [Sec. 3.1]	1	Ins-only	$O(k \cdot n)$ words	No $f(k) \cdot n^{1-\epsilon} \log^{O(1)} n$ bits algorithm
k-FVS, k-Path k-Treewidth [Sec. 3.1]	1	Ins-Del.	$\widetilde{O}(k \cdot n)$ words	No $f(k) \cdot n^{1-\epsilon} \log^{O(1)} n$ bits algorithm
k-Girth, k-DomSet, [Sec. 3.1]	1	Ins-Del.	$O(n^2)$ bits	No $f(k) \cdot n^{2-\epsilon} \log^{O(1)} n$ bits algorithm
$\frac{\beta}{32}$-approximation for size of min DomSet on graphs of arboricity β [Sec. 3.2]	1	Ins-only	$\widetilde{O}(n\beta)$ bits	No $f(\beta) \cdot n^{1-\epsilon}$ bits algorithm
AND-compatible problems and OR-compatible problems [Sec. 3.3]	1	Ins-only	$O(n^2)$ bits	No $\widetilde{O}(f(k) \cdot n^{1-\epsilon})$ bits algorithm
d-SAT with N variables [Sec. E]	1	Clause Arrival	$\widetilde{O}(d \cdot N^d)$ bits	$\Omega((N/d)^d)$ bits

Figure 2 Table summarizing our results (in the order in which they appear in the paper). All our algorithms are deterministic. All the lower bounds are unconditional, and hold even for randomized algorithms in insertion-only streams.

Problem	# of Passes	Type of Stream	Type of Algorithm	Approx. Ratio	Space Bound
k-VC	1	Ins-only	Det.	1	$O(k^2 \log n)$ bits [13]
k-VC	1	Ins-only	Rand.	1	$\Omega(k^2)$ bits [13]
k-VC	1	Ins-Del.	Rand.	1	$O(k^2 \log^{O(1)} n)$ bits [11]
k-VC	2^k	Ins-only	Det.	1	$O(k \log n)$ bits [Algorithm 4]
k-VC	$k \cdot 2^k$	Ins-only.	Det.	1	$O(k \log n)$ bits [Algorithm 2]
Estim. k-VC	$\Omega(k/\log n)$	Ins-only.	Rand.	1	$O(k \log n)$ bits [1, Theorem 16]
Estim. k-VC on Trees	1	Ins-only.	Det. Rand.	$(3/2-\epsilon)$	$\Omega(n)$ bits [29, Theorem 6.1] $\Omega(\sqrt{n})$ bits [29, Theorem 6.1]

Figure 3 Table summarizing some of the results for the k-VC problem in the different settings outlined in Remark 1.

of $O(k^2)$ space. They also showed an almost matching lower bound of $\Omega(k^2)$ bits for any streaming algorithm for k-VC. A sequence of papers showed how to solve the k-VC problem in more general streaming models: Chitnis et al. [13, 12] gave an $\widetilde{O}(k^2)$ space algorithm under a particular promise, which was subsequently removed in [11].

Recently, there have been several papers considering the problem of estimating the size of a maximum matching using $o(n)$ space in graphs of bounded arboricity. If the space is required to be sublinear in n, then versions of the problem that involve estimating the size of a maximum matching (rather than demonstrating such a matching) become the focus. Since the work of Esfandiari et al. [29], there have been several sublinear space algorithms [38, 39, 16, 11] which obtain $O(\alpha)$-approximate estimations of the size of maximum matching in graphs of arboricity α. The current best bounds [6, 16] for insertion-only streams is $O(\log^{O(1)} n)$ space and for insertion-deletion streams is $\widetilde{O}(\alpha \cdot n^{4/5})$. All of these results can be viewed as parameterized streaming algorithms (FPS or SubPS) for approximately estimating the size of maximum matching in graphs parameterized by the arboricity.

2 Parameterized Streaming Algorithms Inspired by FPT techniques

In this section we design parameterized streaming algorithms using three techniques from the world of parameterized algorithms, viz. branching, iterative compression and bidimensionality.

2.1 Multipass FPS algorithm for k-VC using Branching

The streaming algorithm (Algorithm 2) from Section 2.2 already uses optimal storage of $O(k \log n)$ bits but requires $O(2^k \cdot (n-k))$ passes. In this section, we show how to reduce the number of passes to 2^k (while still maintaining the same storage) using the technique of bounded-depth search trees (also known as branching). The method of bounded-depth search trees gives a folklore FPT algorithm for k-VC which runs in $2^{O(k)} \cdot n^{O(1)}$ time. The idea is simple: any vertex cover must contain at least one end-point of each edge. We now build a search tree as follows: choose an arbitrary edge, say $e = u - v$ in the graph. Start with the graph G at the root node of the search tree. Branch into two options, viz. choosing either u or v into the vertex cover[7]. The resulting graphs at the two children of the root node are $G - u$ and $G - v$. Continue the branching process. Note that at each step, we branch into two options and we only need to build the search tree to height k for the k-VC problem. Hence, the binary search tree has $2^{O(k)}$ leaf nodes. If the resulting graph at any leaf node is empty (i.e., has no edges) then G has a vertex cover of size $\leq k$ which can be obtained by following the path from the root node to the leaf node in the search tree. Conversely, if the resulting graphs at none of the leaf nodes of the search tree are empty then G does not have a vertex cover of size $\leq k$: this is because at each step we branched on all the (two) possibilities at each node of the search tree.

Simulating branching-based FPT algorithm using multiple passes: We now simulate the branching-based FPT algorithm described in the previous section using 2^k passes and $O(k \log n)$ bits of storage in the streaming model.

▶ **Definition 3.** *Let $V(G) = \{v_1, v_2, \ldots, v_n\}$. Fix some ordering ϕ on $V(G)$ as follows: $v_1 < v_2 < v_3 < \ldots < v_n$. Let $\texttt{Dict}_k$ be the dictionary ordering on the 2^k binary strings of $\{0,1\}^k$. Given a string $X \subseteq \{0,1\}^k$, let $\texttt{Dict}_k(\texttt{Next}(X))$ denote the string that comes immediately after X in the ordering $\texttt{Dict}_k$. We set $\texttt{Dict}_k(\texttt{Next}(1^k)) = \spadesuit$*

We formally describe our multipass algorithm in Algorithm 1. This algorithm crucially uses the fact that in each pass we see the edges of the stream in the *same* order.

▶ **Theorem 4.** [$\star$] *Algorithm 1 correctly solves the k-VC problem using 2^k passes and $O(k \log n)$ bits of storage.*

The proof of Theorem 4 is deferred to Appendix A. Note that the total storage of Algorithm 1 is $O(k \log n)$ bits which is essentially optimal since the algorithm also outputs a vertex cover of size at most k (if one exists).

[7] Note that if we choose u in the first branch then that does not imply that we cannot or will not choose v later on in the search tree

Algorithm 1 2^k-pass Streaming Algorithm for k-VC using $O(k \log n)$ bits via Branching

Input: An undirected graph $G = (V, E)$ and an integer k.
Output: A vertex cover S of G of size $\leq k$ (if one exists), and NO otherwise
Storage: i, j, S, X

Let $X = 0^k$, and suppose the edges of the graph are seen in the order $e_1, e_2, \ldots, e_m$
while $X \neq \spadesuit$ **do**
  $S = \emptyset, i = 1, j = 1$
  while $i \neq k + 1$ **do**
    Let $e_j = u - v$ such that $u < v$ under the ordering ϕ
    if Both $u \notin S$ and $v \notin S$ **then**
      if $X[i] = 0$ **then** $S \leftarrow S \cup \{u\}$
      else $S \leftarrow S \cup \{v\}$
      $i \leftarrow i + 1$
    $j \leftarrow j + 1$
  if $j = m + 1$ **then** Return S and abort
  else $X \leftarrow \texttt{Dict}_k(\texttt{Next}(X))$
if $X = \spadesuit$ **then** Return NO

The next natural question is whether one need exponential (in k) number of passes when we want to solve the k-VC problem using only $O(k \log n)$ bits. A lower bound of $(k/\log n)$ passes follows for such algorithms from the following result of Abboud et al.

▶ **Theorem 5.** (rewording of [1, Thm 16]) *Any algorithm for the k-VC problem which uses S bits of space and R passes must satisfy $RS \geq n^2$*

2.2 Multipass `FPS` algorithm for k-VC using Iterative Compression

The technique of *iterative compression* was introduced by Reed et al. [42] to design the first FPT algorithm for the k-OCT problem[8]. Since then, iterative compression has been an important tool in the design of faster parameterized algorithms [10, 14, 9] and kernels [20]. In Section B, using the technique of iterative compression, we design an algorithm (Algorithm 2) for k-VC which uses $O(k \log n)$ bits but requires $O(k \cdot 2^{2k})$ passes. Although this algorithm is strictly worse (same storage, but higher number of passes) compared to Algorithm 1, we include it here to illustrate that the technique of iterative compression can be used in the streaming setting.

As in the FPT setting, a natural problem to attack using iterative compression in the streaming setting would be the k-OCT problem. It is known that 0-OCT, i..e, checking if a given graph is bipartite, in the 1-pass model has an upper bound of $O(n \log n)$ bits [31] and a lower bound of $\Omega(n \log n)$ bits [45]. For $k \geq 1$, can we design a $g(k)$-pass algorithm for k-OCT which uses $\widetilde{O}(f(k) \cdot n)$ bits for some functions f and g, maybe using iterative compression? To the best of our knowledge, such an algorithm is not known even for 1-OCT.

2.3 Minor-Bidimensional problems belong to `SemiPS`

The theory of bidimensionality [23, 24] provides a general technique for designing (subexponential) `FPT` for `NP`-hard graph problems on various graph classes. In this section, we briefly

[8] Is there a set of size at most k whose deletion makes the graph odd cycle free, i.e. bipartite

sketch how we can use this technique to show that a large class of problems belong to the class `SemiPS`. All the details (including graph-theoretic definitions such as minors, treewidth, etc.) of this section are deferred to Appendix C

▶ **Definition 6 (minor-bidimensional).** *A graph problem Π is $g(r)$-minor-bidimensional if*

- *The value of Π on the $r \times r$ grid is $\geq g(r)$*
- *Π is closed under taking minors, i.e., the value of Π does not increase under the operations of vertex deletions, edge deletions, edge contractions.*

Hence, we obtain the following "win-win" approach for designing FPT algorithms for bidimensional problems:

- Either the graph has small treewidth and we can then use dynamic programming algorithms for bounded treewidth graphs; or
- The treewidth is large[9] which implies that the graph contains a large grid as a minor. This implies that the solution size is large, since the parameter is minor-bidimensional.

Several natural graph parameters are known to be minor-bidimensional. For example, treewidth is $\Omega(r)$-minor-dimensional and Feedback Vertex Set, Vertex Cover, Minimum Maximal Matching, Long Path, etc are $\Omega(r^2)$-minor-bidimensional. To design parameterized streaming algorithms, we will replace the dynamic programming step for bounded treewidth graphs by simply storing all the edges of such graphs. The main theorem of this section is that minor-bidimensional problems belong to the class `SemiPS` (proof deferred to Appendix C).

▶ **Theorem 7.** [$\star$][10] ***(minor-bidimensional problems $\in$ `SemiPS`)*** *Let Π be a $g(r)$-minor-dimensional problem. Then the k-Π problem on graphs with n vertices can be solved using*

- *$O((g^{-1}(k+1))^{10} \cdot n)$ space in insertion-only streams*
- *$\widetilde{O}((g^{-1}(k+1))^{10} \cdot n)$ space in insertion-deletion streams*

Theorem 7 implies the following results for specific graph problems[11]:

- Since Treewidth is $\Omega(r)$-minor-bidimensional, it follows that k-Treewidth has an $O(k^{10} \cdot n)$ space algorithm in insertion-only streams and $\widetilde{O}(k^{10} \cdot n)$ space algorithm in insertion-deletion streams.
- Since problems such as Long Path, Vertex Cover, Feedback Vertex Set, Minimum Maximal Matching, etc. are $\Omega(r^2)$-minor-bidimensional, it follows that their parameterized versions have $O(k^5 \cdot n)$ space algorithm in insertion-only streams and $\widetilde{O}(k^5 \cdot n)$ space algorithm in insertion-deletion streams.

In Section 3.1, we design algorithms for some of the aforementioned problems with smaller storage. In particular, we design problem-specific structural lemmas (for example, Lemma 30 and Lemma 35) to reduce the dependency of k on the storage from $k^{O(1)}$ to k.

▶ Remark 8. It is tempting to conjecture a lower bound complementing Theorem 7: for example, can we show that the bounds for minor-bidimensional problems are tight for `SemiPS`, i.e., they do not belong to `SubPS` or even `FPS`? Unfortunately, we can rule out such a converse to Theorem 7 via the two examples of Vertex Cover (VC) and Feedback Vertex Set (FVS) which are both $\Omega(r^2)$-minor-bidimensional. Chitnis et al. [13] showed that k-VC can be solved in $O(k^2)$ space and hence belongs to the class `FPS`. However, we show (Theorem 34) that k-FVS cannot belong to `SubPS` since it has a $\Omega(n \log n)$ bits lower bound for $k = 0$.

[9] Chuzhoy and Tan [15] showed that treewidth $= O(r^9 \cdot \log^{O(1)} r) \Rightarrow$ there is a $r \times r$ grid minor
[10] Proofs of all results marked with [$\star$] are deferred to the Appendix due to space constraints
[11] We omit the simple proofs of why these problems satisfy the conditions of Definition 23

3 Lower Bounds for Parameterized Streaming Algorithms

3.1 Tight Problems for the classes SemiPS and BrutePS

In this section we show that certain problems are tight for the classes SemiPS and BrutePS. All of the results hold for 1-pass in the insertion-only model. Our algorithms are deterministic, while the lower bounds also hold for randomized algorithms.

Tight Problems for the class SemiPS: We now show that some parameterized problems are tight for the class SemiPS, i.e.,

- They belong to SemiPS, i.e., can be solved using $\widetilde{O}(g(k) \cdot n)$ bits for some function g.
- They do not belong to SubPS, i.e., there is no algorithm which uses $\widetilde{O}(f(k) \cdot n^{1-\epsilon})$ bits for any function f and any constant $1 > \epsilon > 0$. We do this by showing $\Omega(n \cdot \log n)$ bits lower bounds for these problems for constant values of k.

For each of the problems considered in this section, a lower bound of $\Omega(n)$ bits (for constant values of k) was shown by Chitnis et al. [11]. To obtain the improved lower bound of $\Omega(n \cdot \log n)$ bits for constant k, we will reduce from the PERM problem defined by Sun and Woodruff [45].

> PERM
> *Input*: Alice has a permutation $\delta : [N] \to [N]$ which is represented as a bit string B_δ of length $N \log N$ by concatenating the images of $1, 2, \ldots, N$ under δ. Bob has an index $I \in [N \log N]$.
> *Goal*: Bob wants to find the I-th bit of B_δ

Sun and Woodruff [45] showed that the one-way (randomized) communication complexity of PERM is $\Omega(N \cdot \log N)$. Using the PERM problem, we show $\Omega(n \cdot \log n)$ bit lower bounds for constant values of k for various problem such as k-Path, k-Treewidth, k-Feedback-Vertex-Set, etc. We also show a matching upper bound for these problems: for each k, these problems can be solved using $O(kn \cdot \log n)$ words in insertion-only streams and $\widetilde{O}(kn \cdot \log n)$ words in insertion-deletion streams. The proofs of these results are deferred to Appendix D.1. To the best of our knowledge, the only problems known previously to be tight for SemiPS were k-vertex-connectivity and k-edge-connectivity [18, 45, 28].

Tight Problems for the class BrutePS: We now show that some parameterized problems are tight for the class BrutePS, i.e.,

- They belong to BrutePS, i.e., can be solved using $O(n^2)$ bits. Indeed any graph problem can be solved by storing the entire adjacency matrix which requires $O(n^2)$ bits.
- They do not belong to SubPS, i.e., there is no algorithm which uses $\widetilde{O}(f(k) \cdot n^{1+\epsilon})$ bits for any function f and any $\epsilon \in (0, 1)$. We do this by showing $\Omega(n^2)$ bits lower bounds for these problems for constant values of k via reductions from the INDEX problem.

> **Index**
> *Input*: Alice has a string $B = b_1 b_2 \ldots b_N \in \{0, 1\}^N$. Bob has an index $I \in [N]$
> *Goal*: Bob wants to find the value b_I

There is a $\Omega(N)$ lower bound on the (randomized) one-way communication complexity of INDEX [37]. Via reduction from the INDEX problem, we are able to show $\Omega(n^2)$ bits for constant values of k for several problems such as k-Dominating-Set and k-Girth. The proofs of these reductions are deferred to Appendix D.2

▶ Remark 9. We usually only design FPT algorithms for NP-hard problems. However, parameterized streaming algorithms make sense for all graph problems since we are only comparing ourselves against the naive choice of storing all the $O(n^2)$ bits of adjacency matrix. Hence, here we consider the k-Girth problem as an example of a polynomial time solvable problem.

Finally in Section E, we show that for any $d \geq 2$, any streaming algorithm for d-SAT (in the clause arrival model) must essentially store all the clauses (and hence fits into the "brute-force" streaming setting). This is the only non-graph-theoretic result in this paper, and may be viewed as a "streaming analogue" of the Exponential Time Hypothesis.

3.2 Lower bound for approximating size of minimum Dominating Set on graphs of bounded arboricity

▶ **Theorem 10.** *Let $\beta \geq 1$ be any constant. Then any algorithm which $\frac{\beta}{32}$-approximates the size of a min dominating set on graphs of arboricity $\beta + 2$ requires $\Omega(n)$ space.*

Note that Theorem 10 shows that the naive algorithm which stores all the $O(n\beta)$ edges of an β-arboriticy graph is essentially optimal. Our lower bound holds even for randomized algorithms (required to have success probability $\geq 3/4$) and also under the vertex arrival model, i.e., we see at once all edges incident on a vertex. We (very) closely follow the outline from [2, Theorem 4] who used this approach for showing that any α-approximation for estimating size of a minimum dominating set in general graphs requires $\tilde{\Omega}(\frac{n^2}{\alpha^2})$ space. Because we are restricted to bounded arboriticy graphs, we cannot just sue their reduction as a black-box but need to adapt it carefully for our purposes.

Let $V(G) = [n+1]$, and $\mathcal{F}_\beta$ be the collection of all subsets of $[n]$ with cardinality β. Consider the following distribution $\mathcal{D}_{\mathsf{est}}$ for $\mathsf{DomSet}_{\mathsf{est}}$.

Distribution $\mathcal{D}_{\mathsf{est}}$: A hard input distribution for $\mathsf{DomSet}_{\mathsf{est}}$.

- **Alice.** The input of Alice is a collection of n sets $\mathcal{S}' = \{S'_1, S'_2, \ldots, S'_n\}$ where for each $i \in [n]$ we have that $S'_i = \{i\} \cup S_i$ with S_i being a set chosen independently and uniformly at random from $\mathcal{F}_\beta$.
- **Bob.** Pick $\theta \in \{0,1\}$ and $i^* \in [n]$ independently and uniformly at random; the input of Bob is a single set T defined as follows.
  - If $\theta = 0$, then $\overline{T} = [n] \setminus T$ is a set of size $\beta/8$ chosen uniformly at random from S_{i^*}.
  - If $\theta = 1$, then $\overline{T} = [n] \setminus T$ is a set of size $\beta/8$ chosen uniformly at random from $[n] \setminus S_{i^*}$.

Recall that $\mathrm{OPT}(\mathcal{S}', T)$ denotes the size of the minimum *dominating set* of the graph G whose edge set is given by $N[i] = \{i\} \cup S_i$ for each $i \in [n]$ and $N[n+1] = \{n+1\} \cup T$. It is easy to see that G has arboricity $\leq (\beta+2)$ since it has $(n+1)$ vertices and $\leq (\beta+1)n + (1+n-\frac{\beta}{8})$ edges. We first establish the following lemma regarding the parameter θ and $\mathrm{OPT}(\mathcal{S}', T)$ in the distribution $\mathcal{D}_{\mathsf{est}}$.

▶ **Lemma 11.** [⋆] *Let $\alpha = \frac{\beta}{32}$. Then, for $(\mathcal{S}', T) \sim \mathcal{D}_{\mathsf{est}}$ we have*

1. $\Pr\left(OPT(\mathcal{S}', T) = 2 \mid \theta = 0\right) = 1$.
2. $\Pr\left(OPT(\mathcal{S}', T) > 2\alpha \mid \theta = 1\right) = 1 - o(1)$.

The proof of Lemma 11 is deferred to Appendix F.1. The first observation is that the distribution $\mathcal{D}_{\mathsf{est}}$ is not a product distribution due to the correlation between the input given to Alice and Bob. However, we can express the distribution $\mathcal{D}_{\mathsf{est}}$ as a convex combination of a relatively small set of product distributions. The proof of Theorem 10 then follows by showing a lower bound on this set of product distributions. This proof is a bit technical, and we defer it to Appendix F.2 due to space constraints.

3.3 Streaming Lower Bounds Inspired by Kernelization Lower Bounds

Streaming algorithms and kernelization are two (somehwhat related) compression models. In kernelization, we have access to the whole input but our computation power is limited to polynomial time whereas in streaming algorithms we don't have access to the whole graph (have to pay for whatever we store) but have unbounded computation power on whatever part of the input we have stored.

A folklore result states that a (decidable) problem is FPT if and only if it has a kernel. Once the fixed-parameter tractability for a problem is established, the next natural goals are to reduce the running time of the FPT algorithm and reduce the size of the kernel. In the last decade, several frameworks have been developed to show (conditional) lower bounds on the size of kernels [5, 21, 22, 27, 33]. Inspired by these frameworks, we define a class of problems, which we call as AND-compatible and OR-compatible, and show (unconditionally) that none of these problems belong to the class `SubPS`.

▶ **Definition 12.** *We say that a graph problem* Π *is* AND-compatible *if there exists a constant* k *such that*

- *for every* $n \in \mathbb{N}$ *there exists a graph* G_{YES} *of size* n *such* $\Pi(G_{\text{YES}}, k)$ *is a YES instance*
- *for every* $n \in \mathbb{N}$ *there exists a graph* G_{NO} *of size* n *such* $\Pi(G_{\text{NO}}, k)$ *is a NO instance*
- *for every* $t \in \mathbb{N}$ *we have that* $\Pi\Big(\uplus_{i=1}^{t} G_i, k\Big) = \wedge_{i=1}^{t}\Pi(G_i, k)$ *where* $G = \uplus_{i=1}^{t} G_i$ *denotes the union of the vertex-disjoint graphs* $G_1, G_2, \ldots, G_t$

Examples of *AND-compatible* graph problems are k-Treewidth, k-Girth, k-Pathwidth, k-Coloring, etc.

▶ **Definition 13.** *We say that a graph problem* Π *is* OR-compatible *if there exists a constant* k *such that*

- *for every* $n \in \mathbb{N}$ *there exists a graph* G_{YES} *of size* n *such* $\Pi(G_{YES}, k)$ *is a YES instance*
- *for every* $n \in \mathbb{N}$ *there exists a graph* G_{NO} *of size* n *such* $\Pi(G_{NO}, k)$ *is a NO instance*
- *for every* $t \in \mathbb{N}$ *we have that* $\Pi(\uplus_{i=1}^{t} G_i, k) = \vee_{i=1}^{t}\Pi(G_i, k)$ *where* $G = \uplus_{i=1}^{t} G_i$ *denotes the union of the vertex-disjoint graphs* $G_1, G_2, \ldots, G_t$

A general example of an *OR-compatible* graph problem is the subgraph isomorphism problem parameterized by size of smaller graph: given a graph G of size n and a smaller graph H of size k, does G have a subgraph isomorphic to H? Special cases of this problem are k-Path, k-Clique, k-Cycle, etc.

▶ **Theorem 14.** *If* Π *is an* AND-compatible *or an* OR-compatible *graph problem then* $\Pi \notin$ *SubPS*

Proof. Let Π be an *AND-compatible* graph problem, and $G = \uplus_{i=1}^{t} G_i$ for some $t \in \mathbb{N}$. We claim that any streaming algorithm ALG for Π must use t bits. Intuitively, we need at least one bit to check that each of the instances (G_i, k) is a YES instance of Π (for all $1 \leq i \leq t$). Consider a set of t graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_t\}$: note that we don't fix any of these graphs yet. For every subset $X \subseteq [t]$ we define the instance (G_X, k) of Π where $G_X = \uplus_{j \in J} G_j$. Suppose that ALG uses less than t bits. Then by pigeonhole principle, there are two subsets I, I' of $[t]$ such that ALG has the same answer on (G_I, k) and $(G_{I'}, k)$. Since $I \neq I'$ (without loss of generality) there exists i^* such that $i^* \in I \setminus I'$. This is where we now fix each of the graphs in $\mathcal{G}$ to arrive at a contradiction: consider the input where $G_i = G_{\text{YES}}$ for all $(I \cup I') \setminus i^*$ and $G_{i^*} = G_{\text{NO}}$. Then, it follows that (G_I, k) is a NO instance but $(G_{I'}, k)$ is a YES instance.

Suppose that $\Pi \in \mathtt{SubPS}$, i.e., there is an algorithm for Π which uses $f(k) \cdot N^{1-\epsilon} \cdot \log^{O(1)} N$ bits (for some $1 > \epsilon > 0$) on a graph G of size N to decide whether (G, k) is a YES or NO instance. Let $G = \uplus_{i=1}^{t} G_i$ where $|G_i| = n$ for each $i \in [t]$. Then $|G| = N = nt$. By the previous paragraph, we have that

$$f(k) \cdot (nt)^{1-\epsilon} \cdot \log^{O(1)}(nt) \geq t \Rightarrow f(k) \cdot n^{1-\epsilon} \cdot \log^{O(1)}(nt) \geq t^{\epsilon}$$

Choosing $t = n^{\frac{2-\epsilon}{\epsilon}}$ we have that $f(k) \cdot \log^{O(1)} n^{1+(\frac{2-\epsilon}{\epsilon})} \geq n$, which is a contradiction for large enough n (since k and ϵ are constants).

We now prove the lower bound for AND-compatible problems. Recall that De Morgan's law states that $\neg(\vee_i P_i) = \wedge_i(\neg P_i)$. Hence, if Π is an *OR-compatible* graph problem then the complement[12] problem $\overline{\Pi}$ is an *AND-compatible* graph problem, and hence the lower bound follows from the previous paragraph. ◀

▶ Remark 15. Note that throughout this paper we have considered the model where we allow unbounded computation at each edge update, and also at the end of the stream. However, if we consider a **restricted** model of allowing only polynomial (in input size n) computation at each edge update and also at end of the stream, then it is easy to see that existing (conditional) lower bounds from the parameterized algorithms and kernelization setting translate easily to this restricted model. For example, the following two lower bounds for parameterized streaming algorithms follow immediately in the restricted (polytime computation) model:

- Let X be a graph problem that is $W[i]$-hard parameterized by k (for some $i \geq 1$). Then (in the polytime computation model) $X \notin \mathtt{FPS}$ unless $\mathtt{FPT} = W[i]$.
- Let X be a graph problem that is known to not have a polynomial kernel unless $\mathtt{NP} \subseteq$ co$\mathtt{NP}$/poly. Then (in the polytime computation model) X does not have a parameterized streaming algorithm which uses $k^{O(1)} \cdot \log^{O(1)} n$ bits, unless $\mathtt{NP} \subseteq$ co$\mathtt{NP}$/poly.

4 Conclusions & Open Problems

In this paper, we initiate a systematic study of graph problems from the paradigm of parameterized streaming algorithms. We define space complexity classes of FPS, SubPS, SemiPS, SupPS and BrutePS, and then obtain tight classifications for several well-studied graph problems such as Longest Path, Feedback Vertex Set, Girth, Treewidth, etc. into these classes. Our parameterized streaming algorithms use techniques of bidimensionality, iterative compression and branching from the FPT world. In addition to showing lower bounds for some parameterized streaming problems via communication complexity, we also show how (conditional) lower bounds for kernels and W-hard problems translate to lower bounds for parameterized streaming algorithms.

Our work leaves open several concrete questions. We list some of them below:

- The streaming algorithm (Algorithm 1) for k-VC (on insertion-only streams) from Section 2.1 has an optimal storage of $O(k \log n)$ bits but requires 2^k passes. Can we reduce the number of passes to poly(k), or instead show that we need passes which are superpolynomial in k if we restrict space usage to $O(k \log n)$ bits? The only known lower bound for such algorithms is $(k/\log n)$ passes (see Theorem 5).
- For $k \geq 1$ can we design algorithms which use $f(k) \cdot n \cdot \log^{O(1)} n$ bits and $g(k)$ passes for the k-OCT problem (for some functions f, g)? The technique of iterative compression seems like a natural tool to use here.

[12] By complement, we mean that $\overline{\Pi}(G, k)$ is YES if and only if $\Pi(G, k)$ is NO

References

1 Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Ami Paz. Smaller Cuts, Higher Lower Bounds. *CoRR*, abs/1901.01630, 2019. URL: `http://arxiv.org/abs/1901.01630`, `arXiv:1901.01630`.

2 Sepehr Assadi, Sanjeev Khanna, and Yang Li. Tight bounds for single-pass streaming complexity of the set cover problem. In *STOC*, pages 698–711, 2016.

3 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. Information Theory Methods in Communication Complexity. In *CCC*, pages 93–102, 2002.

4 Neta Barkay, Ely Porat, and Bar Shalem. Efficient Sampling of Non-strict Turnstile Data Streams. In *FCT*, pages 48–59, 2013.

5 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On Problems Without Polynomial Kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.

6 Marc Bury, Elena Grigorescu, Andrew McGregor, Morteza Monemizadeh, Chris Schwiegelshohn, Sofya Vorotnikova, and Samson Zhou. Structural Results on Matching Estimation with Applications to Streaming. *Algorithmica*, 81(1):367–392, 2019.

7 Amit Chakrabarti, Yaoyun Shi, Anthony Wirth, and Andrew Chi-Chih Yao. Informational Complexity and the Direct Sum Problem for Simultaneous Message Complexity. In *FOCS*, pages 270–278, 2001.

8 Chandra Chekuri and Julia Chuzhoy. Polynomial Bounds for the Grid-Minor Theorem. *J. ACM*, 63(5):40:1–40:65, 2016.

9 Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *J. Comput. Syst. Sci.*, 74(7):1188–1198, 2008.

10 Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5):21:1–21:19, 2008.

11 Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *SODA*, pages 1326–1344, 2016.

12 Rajesh Hemant Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Brief announcement: New streaming algorithms for parameterized maximal matching & beyond. In *SPAA*, pages 56–58, 2015.

13 Rajesh Hemant Chitnis, Graham Cormode, Mohammad Taghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *SODA*, pages 1234–1251, 2015.

14 Rajesh Hemant Chitnis, Marek Cygan, Mohammad Taghi Hajiaghayi, and Dániel Marx. Directed Subset Feedback Vertex Set Is Fixed-Parameter Tractable. *ACM Trans. Algorithms*, 11(4):28:1–28:28, 2015.

15 Julia Chuzhoy and Zihan Tan. Towards Tight(er) Bounds for the Excluded Grid Theorem. In *SODA*, pages 1445–1464, 2019.

16 Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and S. Muthukrishnan. The Sparse Awakens: Streaming Algorithms for Matching Size Estimation in Sparse Graphs. In *ESA*, pages 29:1–29:15, 2017.

17 Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

18 Michael S. Crouch, Andrew McGregor, and Daniel Stubbs. Dynamic Graphs in the Sliding-Window Model. In *ESA*, pages 337–348, 2013.

19 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

20 Frank K. H. A. Dehne, Michael R. Fellows, Frances A. Rosamond, and Peter Shaw. Greedy Localization, Iterative Compression, Modeled Crown Reductions: New FPT Techniques, an Improved Algorithm for Set Splitting, and a Novel $2k$ Kernelization for Vertex Cover. In *IWPEC*, pages 271–280, 2004.

21 Holger Dell. AND-Compression of NP-Complete Problems: Streamlined Proof and Minor Observations. *Algorithmica*, 75(2):403–423, 2016.

22 Holger Dell and Dieter van Melkebeek. Satisfiability Allows no Nontrivial Sparsification Unless the Polynomial-Time Hierarchy Collapses. In *STOC*, pages 251–260, 2010.

23 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6):866–893, 2005.

24 Erik D. Demaine and MohammadTaghi Hajiaghayi. The bidimensionality theory and its algorithmic applications. *Comput. J.*, 51(3):292–302, 2008.

25 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

26 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

27 Andrew Drucker. New Limits to Classical and Quantum Instance Compression. In *FOCS*, pages 609–618, 2012.

28 David Eppstein, Zvi Galil, Giuseppe F. Italiano, and Amnon Nissenzweig. Sparsification - a technique for speeding up dynamic graph algorithms. *J. ACM*, 44(5):669–696, 1997.

29 Hossein Esfandiari, Mohammad Taghi Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming Algorithms for Estimating the Matching Size in Planar Graphs and Beyond. In *SODA*, pages 1217–1233, 2015.

30 Stefan Fafianie and Stefan Kratsch. Streaming kernelization. In *MFCS*, pages 275–286, 2014.

31 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005.

32 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph Distances in the Data-Stream Model. *SIAM J. Comput.*, 38(5):1709–1727, 2008.

33 Lance Fortnow and Rahul Santhanam. Infeasibility of Instance Compression and Succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011.

34 Sariel Har-Peled, Piotr Indyk, Sepideh Mahabadi, and Ali Vakilian. Towards Tight Bounds for the Streaming Set Cover Problem. In *PODS*, pages 371–383, 2016.

35 Monika Rauch Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. Computing on data streams. *External memory algorithms*, 50:107–118, 1998.

36 T. S. Jayram and David P. Woodruff. Optimal Bounds for Johnson-Lindenstrauss Transforms and Streaming Problems with Sub-Constant Error. In *SODA*, pages 1–10, 2011.

37 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

38 Andrew McGregor and Sofya Vorotnikova. Planar Matching in Streams Revisited. In *APPROX/RANDOM*, pages 17:1–17:12, 2016.

39 Andrew McGregor and Sofya Vorotnikova. A Simple, Space-Efficient, Streaming Algorithm for Matchings in Low Arboricity Graphs. In *SOSA*, pages 14:1–14:4, 2018.

40 S. Muthukrishnan. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.

41 Alessandro Panconesi and Aravind Srinivasan. Randomized Distributed Edge Coloring via an Extension of the Chernoff-Hoeffding Bounds. *SIAM J. Comput.*, 26(2):350–368, 1997.

42 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004.

43 Neil Robertson and Paul D. Seymour. Graph minors. V. Excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986.

44 Neil Robertson, Paul D. Seymour, and Robin Thomas. Quickly Excluding a Planar Graph. *J. Comb. Theory, Ser. B*, 62(2):323–348, 1994.

45 Xiaoming Sun and David P. Woodruff. Tight Bounds for Graph Problems in Insertion Streams. In *APPROX-RANDOM*, pages 435–448, 2015.

A Multi-pass streaming algorithm for k-VC using Branching

▷ **Theorem 4.** Algorithm 1 correctly solves the k-VC problem using 2^k passes and $O(k \log n)$ bits of storage.

Proof. First we argue the correctness of Algorithm 1. Suppose that there is a string $X \in \{0,1\}^k$ such that $j = m+1$ and we return the set S. Note that initially we have $S = \emptyset$, and the counter i increases each time we add a vertex to S. Hence, size of S never exceeds k. Moreover, if an edge was not covered already (i.e., neither endpoint was in S) then we add at least one of those end-points in S (depending on whether $X[i]$ is 0 or 1) and increase i by one. Hence, if $j = m+1$ then this means that we have seen (and covered) all the edges and the current set S is indeed a vertex cover of size $\leq k$. Now suppose that the algorithm returns NO. We claim that indeed G cannot have a vertex cover of size k. Suppose to the contrary that G has a vertex cover S^* of size $\leq k$, but Algorithm 1 returned NO. We construct a string $X^* \in \{0,1\}^k$ for which Algorithm 1 would return the set S^*: we start with i=1 and the edge e_1. Since S^* is a vertex cover of G it must cover the edge e_1. Set $X^*[1]$ to be 0 or 1 depending on which of the two endpoints of e_1 is in S^* (if both endpoints are in S^*, then it does not matter what we set $X^*[1]$ to be). Continuing this way suppose we have filled the entries till $X^*[i]$ and the current edge under consideration is e_j. If e_j is not covered then $i \neq k$ since S^* is a vertex cover of G of size $\leq k$. In this case, we set $X^*[i+1]$ to be 0 or 1 depending on which of the two endpoints of e_1 is in S^*.

We now analyze the storage and number of passes required. The number of passes is at most 2^k since we have one pass for each string from $\{0,1\}^k$. During each pass, we store four quantities:

- The string $X \in \{0,1\}^k$ under consideration in this pass. This needs k bits.
- The index i of current bit of the k-bit binary string X under consideration in this pass. This needs $\log k$ bits.
- The index j of the current edge under consideration in this pass. This needs $\log n$ bits.
- The set S. Since size of S never exceeds k throughout the algorithm, this can be done using $k \log n$ bits.

◀

B Streaming algorithm for k-VC using iterative compression

B.1 FPT algorithm for k-VC using iterative compression

We first define a variant problem where we are given some additional information in the input in the form of a vertex cover of size of size $k+1$ (just more than the budget).

> **Compression-VC**
> *Input*: A graph G, a positive integer k and a vertex cover T of size $k+1$
> *Parameter*: k
> *Question*: Does there exist a set $X \subseteq V(G)$ with $|X| \leq k$ such that $G \setminus X$ has no edges?

▶ **Lemma 16 (power of iterative compression).** *k-VC can be solved by k calls to an algorithm for the* Compression-VC *problem.*

Proof. Let $e_1, e_2, \ldots, e_t$ be the edges of a maximal matching M in G, and let V_M be the set of vertices which are matched in M . If $t > k$ then there is no vertex cover of size k since any vertex cover needs to pick at least one vertex from every edge of the maximal matching. Hence, we have $t \leq k$. By maximality of M, it follows that the set V_M forms a vertex cover of size $2t \leq 2k$. For each $2k \geq r \geq k+1$ we now run the Compression-VC problem to see whether there exists a vertex cover of size $r-1$. If the answer is YES, then we continue with the compression. On the other hand, if the the Compression-VC problem answers NO for some $2k \geq r \geq k+1$ then clearly there is no vertex cover of G which has size $\leq k$. ◀

Now we solve the Compression-VC problem via the following problem whose only difference is that the vertex cover in the output must be disjoint from the one in the input:

Disjoint-VC
Input: A graph G, a positive integer k and a vertex cover T of size $k+1$
Parameter: k
Question: Does there exist a set $X \subseteq V(G)$ with $|X| \leq k$ such that $X \cap T = \emptyset$ and $G \setminus X$ has no edges?

▶ **Lemma 17 (adding disjointness).** Compression-VC *can be solved by $O(2^{|T|})$ calls to an algorithm for the* Disjoint-VC *problem.*

Proof. Given an instance $I = (G, T, k)$ of Compression-VC we guess the intersection Y of the given vertex T of size $k+1$ and the desired vertex cover X of size k in the output. We have at most $2^{|T|} - 1$ choices for Y since we can have all possible subsets of T except T itself. Then for each guess for Y, we solve the Disjoint-VC problem for the instance $I_Y = (G \setminus Y, T \setminus Y, k - |Y|)$. It is easy to see that if X is a solution for instance I of Compression-VC, then $X \setminus Y$ is a solution of instance I_Y of Disjoint-VC for $Y = T \cap X$. Conversely, if Z is a solution to some instance $I_Y = (G \setminus Y, T \setminus Y, k - |Y|)$ of Disjoint-VC, then $Z \cup Y$ is a solution for the instance $I = (G, T, k)$ of Compression-VC. ◀

Using a maximal matching, we either start with a vertex cover of size $\leq 2k$ or we can answer that G has no vertex cover of size $\leq k$. Hence, any algorithm for Disjoint-VC gives an algorithm for the k-VC problem, with an additional blowup of $O(2^{2k} \cdot k)$. Since our objective is to show that the $k-$VC problem is FPT, then it is enough to give an FPT algorithm for the Disjoint-VC problem (which has additional structure that we can exploit!). In fact we show that the Disjoint-VC problem can be solved in polynomial time.

▶ **Lemma 18.** *The* Disjoint-VC *problem can be solved in polynomial time.*

Proof. Let (G, T, k) be an instance of Disjoint-VC. Note that $G \setminus T$ has no edges since T is a vertex cover. Meanwhile, if $G[T]$ has even a single edge, then answer is NO since we cannot pick any vertices from T in the vertex cover. So the only edges are between T and $G \setminus T$. Since we cannot pick any vertex from T in vertex cover, we are forced to pick all vertices in $G \setminus T$ which have neighbors in T. Formally, we have to pick the set $X = \{x \notin T \ : \ \exists y \in T \text{ such that } x - y \in E(G)\}$. Note that picking X is both necessary and sufficient. So it simply remains to compare $|X|$ with k and answer accordingly. ◀

Consequently, we obtain a $O(2^{2k} \cdot k \cdot n^{O(1)})$ time algorithm for k-VC by composing these two reductions.

Algorithm 2 Multipass Streaming Algorithm for k-VC using Iterative Compression

Input: An undirected graph $G = (V, E)$, integer k.
Output: A vertex cover S of G of size at most k (if one exists), and NO otherwise
Storage: i, S, Y, V_M

```
1: Find a maximal matching M (upto size k) in 1 pass which saturates the vertices V_M
2: If |M| exceeds k, then return NO and abort
3: Let S = V_M
4: for i = |V_M| to k + 1 do
5:     Y = ∅
6:     while Y ∈ S_k, Y ≠ ♠ do
7:         if |{x ∈ V \ S : ∃y ∈ S \ Y s.t. {x, y} ∈ E(G)}| ≤ k − |Y| then
8:             S ← Y ∪ {x ∈ V \ S : ∃y ∈ S \ Y s.t. x − y ∈ E(G)}      ▷ Requires a pass through the data
9:             Break                       ▷ Found a solution, and reduce value of i by 1
10:        else
11:            Y ← Dict_{S_k}(Next(Y))                              ▷ Try the next subset
12:    if Y = ♠ then
13:        Return NO and abort
14: if i = k then
15:    Return S
```

B.2 Simulating the FPT algorithm in streaming using multiple passes

In this section, we show how to simulate the FPT algorithm of the previous section in the multi-pass streaming model. First, let us fix some order on all subsets of $[n]$.

▶ **Definition 19.** *Let $U = \{u_1, u_2, \ldots, u_n\}$ and $k \leq n$. Let $\mathcal{U}_k$ denote the set of all $\sum_{i=0}^{k} \binom{|U|}{i}$ subsets of U which have at most k elements, and $\mathtt{Dict}_{U_k}$ be the dictionary ordering on $\mathcal{U}_k$. Given a subset $X \in \mathcal{U}_k$, let $\mathtt{Dict}_{U_k}(\mathtt{Next}(X))$ denote the subset that comes immediately after X in the ordering $\mathtt{Dict}_U$. We denote the last subset in the dictionary order of $\mathcal{U}_k$ by $Last(\mathcal{U}_k)$ and use the notation that $\mathtt{Dict}_{U_k}(Last(\mathcal{U}_k)) = ♠$.*

We give our multipass algorithm as Algorithm 2, whose correctness follows from Section B.1. We now analyze the storage and number of passes required.

We first use one pass to store a maximal matching M (upto k edges). The remaining number of passes used by the algorithm is at most $2^{2k} \cdot k = O(2^{2k} \cdot k)$ since we have (k) iterations over the index i, we have 2^{2k} choices for the set $Y \in \mathcal{S}_k$ (since $|S| \leq 2k$) and we need one pass for each execution of Step 7 and Step 8. Throughout the algorithm, we store three quantities:

- We store the vertices V_M saturated by a maximal matching M (but only until the size of M exceeds k in which case we output NO). This needs at most $2k \log n$ bits
- The index i of current iteration. This needs $\log n$ bits.
- The set S. Since size of S never exceeds $2k$ throughout the algorithm, this can be done using $2k \log n$ bits.
- The current subset $Y \subseteq \mathcal{S}_k$ under consideration for being the intersection of S and new potential VC of size $\leq k$. Since $|S| \leq 2k$ and we store S explicitly, it follows that we can store Y and find $\mathtt{Next}(Y)$ using $O(k \log n)$ bits.

Hence, the total storage of the algorithm is $O(k \log n)$ bits which is essentially optimal since the algorithm also outputs a vertex cover of size at most k (if one exists).

C Minor-Bidimensional problems belong to SemiPS

The theory of bidimensionality [23, 24] provides a general technique for designing (subexponential) FPT for NP-hard graph problems on various graph classes. First, we introduce some graph theoretic concepts.

▶ **Definition 20 (treewidth).** *Let G be a given undirected graph. Let T be a tree and $B : V(T) \to 2^{V(G)}$. The pair (T, B) is a* tree decomposition *of an undirected graph G if every vertex $x \in V(T)$ of the tree T has an assigned set of vertices $B_x \subseteq V(G)$ (called a bag) such that the following properties are satisfied:*
- ***(P1)**: $\bigcup_{x \in V(T)} B_x = V(G)$.*
- ***(P2)**: For each $\{u, v\} \in E(G)$, there exists an $x \in V(T)$ such that $u, v \in B_x$.*
- ***(P3)**: For each $v \in V(G)$, the set of vertices of T whose bags contain v induce a connected subtree of T.*

The width *of a tree decomposition (T, B) is $\max_{x \in V(T)} |B_x| - 1$. The treewidth of a graph G, usually denoted by $\boldsymbol{tw}(G)$, is the minimum width over all tree decompositions of G.*

Intuitively, the treewidth of a graph captures how tree-like it is. Trees (and more generally forests) have treewidth 1.

▶ **Definition 21 (minor).** *Let H, G be two undirected graphs. We say that H is a minor of G if H can be obtained from G by a sequence of edge deletions, vertex deletions or edge contractions.*

One of the foundational results of graph theory is the Excluded Grid Minor Theorem of Robertson and Seymour [43] which states that large treewidth forces large grid minors:

▶ **Theorem 22.** [43] *There is a function $f : \mathbb{N} \to \mathbb{N}$ such that for $r \geq 1$ any graph of treewidth $\geq f(r)$ contains the $r \times r$ grid as a minor.*

Robertson and Seymour [43] did not provide an explicit bound on f, but proved it was bounded by a tower of exponentials. The first explicit bounds on f were given by Robertson, Seymour and Thomas [44] who showed that $f(r) = 2^{O(r^5)}$ suffices and there are graphs which force $f(r) = \Omega(r^2 \cdot \log r)$. The question whether $f(r)$ can be shown to be bounded by a polynomial in r was open for a long time until Chekuri and Chuzhoy [8] showed that $f(r) = O(r^{98} \cdot \log^{O(1)} r)$ suffices. The current best bound is $f(r) = O(r^9 \cdot \log^{O(1)} r)$ due to Chuzhoy and Tan [15]. Henceforth, for ease of presentation, we will use the weaker bound $f(r) = O(r^{10})$.

The theory of bidimensionality [23, 24] exploits the idea that many problems can be solved efficiently via dynamic programming on graphs of bounded treewidth, and have large values on grid-like graphs.

▶ **Definition 23 (minor-bidimensional).** *A graph problem Π is said to be $g(r)$-minor-bidimensional if*
- *The value of Π on the $r \times r$ grid is $\geq g(r)$*
- *Π is closed under taking minors, i.e., the value of Π does not increase under the operations of vertex deletions, edge deletions, edge contractions.*

Hence, we obtain a "win-win" approach for designing FPT algorithms for bidimensional problems as follows:
- Either the graph has small treewidth and we can then use dynamic programming algorithms for bounded treewidth graphs; or

- The treewidth is large which implies that the graph contains a large grid as a minor. This implies that the solution size is large, since the parameter is minor-bidimensional.

Several natural graph parameters are known to be minor-bidimensional. For example, treewidth is $\Omega(r)$-minor-dimensional and Feedback Vertex Set, Vertex Cover, Minimum Maximal Matching, Long Path, etc are $\Omega(r^2)$-minor-bidimensional. To design parameterized streaming algorithms, we will replace the dynamic programming step for bounded treewidth graphs by simply storing all the edges of such graphs. The following (folklore) lemma shows that bounded treewidth graphs cannot have too many edges.

▶ **Lemma 24.** *Let $G = (V, E)$ be a graph on n vertices. Then $|E(G)| \leq \textbf{tw}(G) \cdot |V(G)|$*

Proof. Let $\textbf{tw}(G) = k$. We first show that there is a vertex $v \in G$ whose degree in G is at most k. Among all tree-decompositions of G of width k, let (T, B) be one which minimizes $|T|$. Since T is a tree, it has a leaf say t. Let t' be the unique neighbor of t in T. Minimality of $|T|$ implies that $B_t \nsubseteq B_{t'}$, since otherwise deleting t (and the bag B_t) would still give a tree-decomposition of G. Hence, there is a vertex $v \in B_t$ and $v \notin B_{t'}$. This implies that all neighbors of v in G must be in the bag B_t, i.e., v has degree at most $|B_t| - 1 = k$. Now, delete the vertex v. It follows from the definition of treewidth that $\textbf{tw}(G - v) \leq \textbf{tw}(G) = k$, and hence we can conclude that $G - v$ also has a vertex of degree at most k. Continuing this way, we obtain $|E(G)| \leq \textbf{tw}(G) \cdot |V(G)|$. ◀

Note that cliques are a tight example (up to factor 2) for the bound in Lemma 24.

▶ **Lemma 25.** *Let Π be a $g(r)$-minor-dimensional problem. Any graph G having more than $O((g^{-1}(k+1))^{10} \cdot |V(G)|)$ edges is a NO (resp. YES) instance of k-Π if Π is a minimization (resp. maximization) problem.*

Proof. Suppose G has more than $\tau(g^{-1}(k+1))^{10} \cdot n$ edges. By Lemma 24, it follows that $\textbf{tw}(G) \geq \tau(g^{-1}(k+1))^{10}$. This implies G has the $g^{-1}(k+1) \times g^{-1}(k+1)$ grid as a minor [15]. Since Π is minor-dimensional, this implies that the value of Π is at least $g(g^{-1}(k+1)) = k+1$, i.e., G is a NO (resp. YES) instance of k-Π if Π is a minimization (resp. maximization) problem. ◀

Lemma 25 implies streaming algorithms for Π in both insertion-only and insertion-deletion streams. First, we define a data structure that we need.

▶ **Definition 26** (**k-sparse recovery algorithm**)**.** *A k-sparse recovery algorithm is a data structure which accepts insertions and deletions of elements from $[n]$ so that, if the current number of elements stored in it is at most k, then these can be recovered in full.*

Barkay et al. [4] showed that a k-sparse recovery algorithm can be constructed deterministically using $\widetilde{O}(k)$ space.

▶ **Theorem 27.** *Let $M \geq 1$. Then we can check if a graph stream contains at most M edges (and also store all these edges) using*

- *$O(M)$ space in insertion-only streams*
- *$\widetilde{O}(M)$ space in insertion-deletion streams*

Proof. The algorithm in insertion-only streams simply stores all the edges. It also maintains a counter (using $O(\log n)$ bits) to count how many edges have been seen so far. If the counter exceeds M then the graph has more than M edges. Otherwise, we have stored the entire graph which uses $O(M)$ space since the number of edges is $\leq M$.

In insertion-deletion streams we also keep a counter (to count how many edges are currently present) and also maintain an M-sparse recovery algorithm $\mathcal{X}$. At the end of the stream, if the counter exceeds M then the graph stream has more than M edges. Otherwise we recover the whole graph by extracting the $\leq M$ edges from $\mathcal{X}$. The counter can be implemented in $O(\log n)$ bits, and $\mathcal{X}$ can be implemented in $\widetilde{O}(M)$ space [4]. ◀

Now we are ready to show the main theorem of this section: minor-bidimensional problems belong to the class `SemiPS`.

▷ **Theorem 7.** (**minor-bidimensional problems are in `SemiPS`**) Let Π be a $g(r)$-minor-dimensional problem. Then the k-Π problem on graphs with n vertices can be solved using

- $O((g^{-1}(k+1))^{10} \cdot n)$ space in insertion-only streams
- $\widetilde{O}((g^{-1}(k+1))^{10} \cdot n)$ space in insertion-deletion streams

Proof. We invoke Theorem 27 with $M = O((g^{-1}(k+1))^{10} \cdot n)$. By Lemma 25, we know that if G has more than $O(g^{-1}(k+1))^{10} \cdot n$ edges then G is a NO (resp. YES) instance of k-Π if Π is a minimization (resp. maximization) problem. Hence, we use the algorithms from Theorem 27 to check if G has at most M edges: if it has more edges then we say NO (resp. YES) if Π is a minimization (resp. maximization) problem, and otherwise we store the entire graph. ◀

Theorem 7 implies the following results for specific graph problems[13]:

- Since Treewidth is $\Omega(r)$-minor-bidimensional, it follows that k-Treewidth has an $O(k^{10} \cdot n)$ space algorithm in insertion-only streams and $\widetilde{O}(k^{10} \cdot n)$ space algorithm in insertion-deletion streams.
- Since problems such as Long Path, Vertex Cover, Feedback Vertex Set, Minimum Maximal Matching, etc. are $\Omega(r^2)$-minor-bidimensional, it follows that their parameterized versions have $O(k^5 \cdot n)$ space algorithm in insertion-only streams and $\widetilde{O}(k^5 \cdot n)$ space algorithm in insertion-deletion streams.

In Appendix D.1, we design algorithms for some of the aforementioned problems with smaller storage. In particular, we design problem-specific structural lemmas (for example, Lemma 30 and Lemma 35) to reduce the dependency of k on the storage from $k^{O(1)}$ to k.

▶ Remark 28. It is tempting to try to prove a lower bound complementing Theorem 7: for example, can we show that the bounds for minor-bidimensional problems are tight for `SemiPS`, i.e., they do not belong to `SubPS` or even `FPS`? Unfortunately, we can rule out such a converse to Theorem 7 via the two examples of Vertex Cover (VC) and Feedback Vertex Set (FVS) which are both $\Omega(r^2)$-minor-bidimensional. Chitnis et al. [13] showed that k-VC can be solved in $O(k^2)$ space and hence belongs to the class `FPS`. However, in this paper we show (Theorem 34) that k-FVS cannot belong to `SubPS` since it has a $\Omega(n \log n)$ bits lower bound for $k = 0$.

[13] We omit the simple proofs of why these problems satisfy the conditions of Definition 23

D Tight Problems for the classes SemiPS and BrutePS

D.1 Tight Problems for the class SemiPS

k-Path
Input: An undirected graph G on n nodes
Parameter: k
Question: Does G have a path of length at least k? (or alternatively, a path on at least $k+1$ vertices)

▶ **Theorem 29.** *The k-Path problem has a lower bound of $\Omega(n \cdot \log n)$ bits even for $k = 5$.*

Proof. Let $n = 2N + 2$. We start with an instance of Perm of size N. Alice has a permutation δ which she uses to build a perfect matching from $[N]$ to $[N]$ as follows: let $W = \{w_1, w_2, \ldots, w_N\}$ and $X = \{x_1, x_2, \ldots, x_N\}$ denote two sets of size N each. Alice's edge set consists of a perfect matching built as follows: for each $i \in [N]$ there is an edge between w_i and $x_{\delta(i)}$. Suppose Bob has the index $I \in [N]$. This corresponds to the ℓ-th bit of $\delta(j)$ for some $j \in [N]$ and $\ell \in [\log N]$. Bob adds two new vertices v, y and adds edges using the index I as follows:

- Bob adds an edge between v and w_j
- Let $S_\ell \subseteq X$ where $S_\ell = \{x_r \ : \ \ell\text{-th bit of } r \text{ is } 0\}$. Bob adds edges from y to each vertex of S_ℓ.

Let the graph constructed this way be G'. It is easy to see that G' has a path of length 5 if and only $x_j \in S_\ell$, i.e., the ℓ-th bit of $\delta(j)$ is zero. Hence, the lower bound of $\Omega(N \log N)$ of Perm translates to an $\Omega(n \log n)$ lower bound for 5-Path. ◀

▶ **Lemma 30.** *Any graph on n vertices with at least nk edges has a path on $k+1$ vertices*

Proof. Preprocess the graph to enforce that the minimum degree $\geq k$ by iteratively deleting vertices of degree $< k$. Then we have a graph G' which has n' vertices and $\geq n'k$ edges whose min degree is $\geq k$. Now consider an arbitrary path P in this graph G', say $v_1 - v_2 - v_3 - \ldots - v_r$. At each intermediate vertex v_j, at most $j-1$ neighbors have been visited, and so at least $k-j+1$ possibilities are open. Hence, there is always a possible next step up to node $k+1$, i.e. there is a path of length k. ◀

▶ **Theorem 31.** *The k-Path problem can be solved using*

- *$O(k \cdot n)$ space in insertion-only streams*
- *$\widetilde{O}(k \cdot n)$ space in insertion-deletion streams*

Proof. We invoke Theorem 27 with $M = nk$. By Lemma 30, we know that if G has more than M edges then it has a k-Path. Hence, we use the algorithms from Theorem 27 to check if G has at most M edges: if it has more edges then we say YES, and otherwise we store the entire graph. ◀

k-Treewidth
Input: An undirected graph G
Parameter: k
Question: Is the treewidth of G at most k?

▶ **Theorem 32.** *[45, Theorem 7] The k-Treewidth problem has a lower bound of $\Omega(n \cdot \log n)$ bits even for $k = 1$.*

Proof. Let $n = 2N + 1$. We start with an instance of Perm of size N. Alice has a permutation δ which she uses to build a perfect matching from $[N]$ to $[N]$ as follows: let $W = \{w_1, w_2, \ldots, w_N\}$ and $X = \{x_1, x_2, \ldots, x_N\}$ denote two sets of size N each. Alice's edge set consists of a perfect matching built as follows: for each $i \in [N]$ there is an edge between w_i and $x_{\delta(i)}$. Suppose Bob has the index $I \in [N]$. This corresponds to the ℓ-th bit of $\delta(j)$ for some $j \in [N]$ and $\ell \in [\log N]$. Bob adds a new vertex v and adds edges using the index I as follows:

- Bob adds an edge between v and w_j
- Let $S_\ell \subseteq X$ where $S_\ell = \{x_r \ : \ \ell\text{-th bit of } r \text{ is } 0\}$. Bob adds edges from v to each vertex of S_ℓ.

Let the graph constructed this way be G'. It is easy to see that G' has no cycles if and only $x_j \notin S_\ell$, i.e., the ℓ-th bit of $\delta(j)$ is 1. Recall that a graph has treewidth 1 if and only if it has no cycles. Hence, the lower bound of $\Omega(N \log N)$ of Perm translates to a $O(n \log n)$ lower bound for k-Treewidth with $k = 1$. ◀

▶ **Theorem 33.** *The k-Treewidth problem can be solved using*

- *$O(k \cdot n)$ space in insertion-only streams*
- *$\widetilde{O}(k \cdot n)$ space in insertion-deletion streams*

Proof. We invoke Theorem 27 with $M = nk$. By Lemma 24, we know that if G has more than M edges then $\mathbf{tw}(G) > k$. Hence, we use the algorithms from Theorem 27 to check if G has at most M edges: if it has more edges then we say NO, and otherwise we store the entire graph. ◀

k-**FVS**
Input: An undirected graph $G = (V, E)$
Parameter: k
Question: Does there exist a set $X \subseteq V$ such that $\lvert X \rvert \leq k$ and $G \setminus X$ has no cycles?

▶ **Theorem 34.** *[45, Theorem 7] The k-FVS problem has a lower bound of $\Omega(n \cdot \log n)$ bits even for $k = 0$.*

Proof. We use exactly the same reduction as in Theorem 32. Recall that graph has a FVS of size 0 if and only if it has no cycles. Also a graph has treewidth 1 if and only if it has no cycles. The proof of Theorem 32 argues that G' has no cycles if and only $x_j \notin S_\ell$, i.e., the ℓ-th bit of $\delta(j)$ is 1. Since G' has $n = 2n + 1$ vertices, the lower bound of $\Omega(N \log N)$ of Perm translates to a $O(n \log n)$ lower bound for k-FVS with $k = 0$. ◀

▶ **Lemma 35.** *If G has a feedback vertex set of size k then $\lvert E(G) \rvert \leq n(k + 1)$*

Proof. Let X be a feedback vertex of G of size k. Then $G \setminus X$ is a forest and has at most $n - k - 1$ edges. Each vertex of X can have degree $\leq n - 1$ in G. Hence, we have that $\lvert E(G) \rvert \leq (n - k - 1) + k(n - 1) \leq n + kn = n(k + 1)$ ◀

▶ **Theorem 36.** *The k-FVS problem can be solved using*

- *$O(k \cdot n)$ space in insertion-only streams*
- *$\widetilde{O}(k \cdot n)$ space in insertion-deletion streams*

Proof. We invoke Theorem 27 with $M = n(k + 1)$. By Lemma 35, we know that if G has more than M edges then G cannot have a feedback vertex set of size k. Hence, we use the algorithms from Theorem 27 to check if G has at most M edges: if it has more edges then we say NO, and otherwise we store the entire graph. ◀

D.2 Tight Problems for the class BrutePS

> **k-Dominating Set**
> *Input*: An undirected graph $G = (V, E)$
> *Parameter*: k
> *Question*: Is there a set $S \subseteq V(G)$ of size $\leq k$ such that each $v \in V \setminus S$ has at least one neighbor in S?

▶ **Theorem 37.** *The k-Dominating Set problem has a lower bound of $\Omega(n^2)$ bits for 1-pass algorithms, even when $k = 3$.*

Proof. Let $r = \sqrt{N}$. We start with an instance of Index where Alice has a bit string $B \in \{0,1\}^N$. Fix a canonical bijection $\phi : [N] \to [r] \times [r]$. We now construct a graph with vertex set $Y = y_1, y_2, \ldots, y_r$ and $W = w_1, w_2, \ldots, w_r$. For each $I \in [N]$ we do the following:

- If $B[I] = 1$ then add the edge $y_{i'} - w_{i''}$ where $\phi_I = (i', i'')$
- If $B[I] = 0$ then do not add any edge

Suppose Bob has the index $I^* \in [N]$. Let $\phi(I^*) = (\alpha, \beta)$ where $\alpha, \beta \in [r]$. Bob adds four new vertices x_1, x_2, z_1 and z_2. He also adds the following edges:

- The edge $x_1 - x_2$
- The edge $z_1 - z_2$
- An edge from x_1 to each vertex of $Y \setminus y_\alpha$
- An edge from z_1 to each vertex of $W \setminus w_\beta$

Let the final constructed graph be G. A simple observation is that if a vertex has degree exactly 1, then its unique neighbor can be assumed to be part of a minimum dominating set. We now show that G has a dominating set of size 3 if and only $B[I^*] = 1$.

First suppose that $B[I^*] = 1$, i.e., $y_\alpha - w_\beta$ forms an edge in G. Then we claim that $\{x_1, z_1, y_\alpha\}$ form a dominating set of size 3. This is because x_1 dominates $x_2 \cup (Y \setminus y_\alpha)$, z_1 dominates $z_2 \cup (W \setminus w_\beta)$ and finally y_α dominates w_β.

Now suppose that G has a dominating set S of size 3 but $y_\alpha - w_\beta \notin E(G)$. Since x_2, z_2 have degree 1 we can assume that $\{x_1, z_1\} \subseteq S$. Let $S \setminus \{x_1, z_1\} = u$. We now consider different possibilities for u and obtain a contradiction in each case:

- $u = x_2$ or $u = z_2$: In this case the vertex y_α is not dominated by S
- $u \in (Y \setminus y_\alpha)$: In this case the vertex y_α is not dominated by S
- $u \in (W \setminus w_\beta)$: In this case the vertex w_β is not dominated by S
- $u = y_\alpha$: In this case the vertex w_β is not dominated by S since $y_\alpha - w_\beta \notin E(G)$
- $u = w_\beta$: In this case the vertex y_α is not dominated by S since $y_\alpha - w_\beta \notin E(G)$

Hence, the 3-Dominating Set problem on graphs with $2r + 4 = O(\sqrt{N})$ vertices can be used to solve instances of the Index problem of size N. Since Index has a lower bound of $\Omega(N)$, it follows that the 3-Dominating Set problem on graphs of n vertices has a lower bound of $\Omega(n^2)$ bits. ◀

> **k-Girth**
> *Input*: An undirected graph G
> *Parameter*: k
> *Question*: Is the length of smallest cycle of G equal to k?

▶ **Theorem 38.** *The k-Girth problem has a lower bound of $\Omega(n^2)$ bits for 1-pass algorithms, even when $k = 3$.*

Proof. Let $r = \sqrt{N}$. We start with an instance of INDEX where Alice has a bit string $B \in \{0,1\}^N$. Fix a canonical bijection $\phi : [N] \to [r] \times [r]$. We now construct a graph with vertex set $Y = y_1, y_2, \ldots, y_r$ and $W = w_1, w_2, \ldots, w_r$. For each $I \in [N]$ we do the following:

- If $B[I] = 1$ then add the edge $y_{i'} - w_{i''}$ where $\phi_I = (i', i'')$
- If $B[I] = 0$ then do not add any edge

Suppose Bob has the index $I^* \in [N]$. Let $\phi(I^*) = (\alpha, \beta)$ where $\alpha, \beta \in [r]$. Bob adds a new vertex z and adds the edges $z - y_\alpha$ and $z - w_\beta$. Let the final constructed graph be G. It is easy to see that $G \setminus z$ is bipartite, and hence has girth ≥ 4 (we say the girth is ∞ if the graph has no cycle). The only edges incident on z are to y_α and w_β. Hence, G has a cycle of length 3 if and only if the edge $y_\alpha - w_\beta$ is present in G, i.e., $B[I^*] = 1$. Hence, the 3-Girth problem on graphs with $2r+1 = O(\sqrt{N})$ vertices can be used to solve instances of the INDEX problem of size N. Since INDEX has a lower bound of $\Omega(N)$, it follows that the 3-Girth problem on graphs of n vertices has a lower bound of $\Omega(n^2)$ bits. ◀

▶ Remark 39. We usually only design `FPT` algorithms for `NP`-hard problems. However, parameterized streaming algorithms make sense for all graph problems since we are only comparing ourselves against the naive choice of storing all the $O(n^2)$ edges. Hence, here we consider the k-Girth problem as an example of a polynomial time solvable problem.

Super-linear lower bounds for multi-pass algorithms for k-Girth were shown in Feigenbaum et al. [32].

E $\Omega((N/d)^d)$ bits lower bound for d-SAT

In this section we show lower bounds for space complexity of streaming algorithms for satisfiability problems. We fix the notation as follows: there are N variables and M clauses. The variable set is fixed, and the clauses arrive one-by-one.

▶ **Theorem 40.** *Any streaming algorithm for d-SAT requires storage of $\Omega((N/d)^d)$ bits, where N is the number of variables*

Proof. For simplicity, we show the result for 2-SAT; the generalization to d-SAT for other $d > 2$ is simple. Let $n = (N/2)^2$. We reduce from the INDEX problem. Let Alice have a string $B = b_1 b_2 \ldots b_n \in \{0,1\}^n$. We now map B to an instance ϕ_B of 2-SAT defined over N variables. The N variables are partitioned into $d = 2$ sets X, Y of $N/2$ variables each. Fix a canonical mapping $\psi : [(N/2)^2] \to [N/2]^2$. For each index $L \in [(N/2)^2]$, we add the following clauses depending on the value of b_L:

- If $b_L = 0$, then add the clause $(x_i \vee y_j)$ where $\psi(L) = (i, j)$.
- If $b_L = 1$, then add the clause $(\overline{x_i} \vee y_j)$ where $\psi(L) = (i, j)$.

Observe that the sub-instance constructed so far is trivially satisfiable, by setting all $y \in Y$ to true. Suppose Bob has the index $L^* \in [n]$. To solve the instance of INDEX, we need to retrieve the value of the bit b_{L^*}. Let $\psi(L^*) = (i^*, j^*)$. We add two new clauses as follows:

- Add the clause $(\overline{y}_{j^*})$ [14]
- Add the clause $(x_{i^*} \vee y_{j^*})$

This completes the construction of the 2-SAT instance ϕ_B. Now we claim that ϕ_B is satisfiable if and only if $b_{L^*} = 0$. Consider a clause of the form $(x \vee y)$ of ϕ_B:

- If $y \neq y_{j^*}$, we can set y to be true and satisfy this clause.

[14] If we insist that all clauses should have cardinality exactly 2, then we can simply create a new "dummy" variable z, and add the clauses $(y_{j^*} \vee z), (y_{j^*} \vee \overline{z})$ to achieve the same effect.

- If $y = y_{j^*}$ but $x \neq x_{i^*}$, then we can satisfy this clause by setting $x = 1$ (or 0, if x appears in complemented form). This is the only time we need to set x, and each such x appears in at most one such clause, so there is no clash[15].

This only leaves the clause on the variables x_{i^*} and y_{j^*}. We must set $y_{j^*} = 0$ to satisfy the clause $\overline{y}_{j^*}$. If $b_{L^*} = 1$ then we have both the clauses $(x_{i^*} \vee y_{j^*})$ and $(\overline{x}_{i^*} \vee y_{j^*})$, and hence the instance ϕ_B is not satisfiable. However, if $b_{L^*} = 0$ then we only have the clause $(x_{i^*} \vee y_{j^*})$, and hence the instance ϕ_B is satisfiable by setting $x_{i^*} = 1$. Hence, the lower bound of $\Omega(n) = \Omega((N/2)^2)$ translates from Index to 2-SAT. ◀

Note that the naive algorithm for d-SAT which stores all the clauses in memory requires $\widetilde{O}(\binom{N}{d}) = \widetilde{O}(d \cdot N^d)$ bits, and therefore Theorem 40 shows that d-SAT is hard from a space perspective (essentially have to store all the clauses) for all $d \geq 2$, whereas there is a transition from P to NP-complete for the time cost when going from 2-SAT to 3-SAT.

F Lower bound for approximating size of minimum Dominating Set on graphs of bounded arboricity

Notation: We use bold face letters to represent random variables. For any random variable $\boldsymbol{X}$, $\textsc{supp}(\boldsymbol{X})$ denotes its support set. We define $|\boldsymbol{X}| := \log |\textsc{supp}(\boldsymbol{X})|$. For any k-dimensional tuple $X = (X_1, \ldots, X_k)$ and any $i \in [k]$, we define $X^{<i} := (X_1, \ldots, X_{i-1})$, and $X^{-i} := (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_k)$. The notation "$X \in_R U$" indicates that X is chosen uniformly at random from a set U. Finally, we use upper case letters (e.g. M) to represent matrices and lower case letter (e.g. v) to represent vectors.

F.1 Proof of Lemma 11

▷ **Lemma 11.** Let $\alpha = \frac{\beta}{32}$. Then for $(\mathcal{S}', T) \sim \mathcal{D}_{\mathsf{est}}$:

1. $\Pr(\text{OPT}(\mathcal{S}', T) = 2 \mid \theta = 0) = 1$.
2. $\Pr(\text{OPT}(\mathcal{S}', T) > 2\alpha \mid \theta = 1) = 1 - o(1)$.

Proof. The first claim is immediate since by construction, when $\theta = 0$ we have that $T \cup S'_{i^*} = [n+1]$ and hence $\{n+1, i^*\}$ forms a dominating set of size 2.

We now prove the second claim, i.e., when $\theta = 1$. The vertex $(n+1)$ dominates all vertices in the set $T \cup \{n+1\}$. It remains to dominate vertices of $\overline{T} = [n] \setminus T$. Since $i \in S'_i$ for each $i \in [n]$ it follows that the set $\{j \ : \ j \in \overline{T}\} \cup \{n+1\}$ forms a dominating set of size $1 + \frac{\beta}{8}$ for G. Fix a collection $\widehat{\mathcal{S}'}$ of 2α sets in $\mathcal{S}' \setminus \{S'_{i^*}\}$, and let $\widehat{\mathcal{S}'} = \{S'_{\mu_1}, S'_{\mu_2}, \ldots, S'_{\mu_{2\alpha}}\}$. Let $T_0 = \overline{T} \setminus \{\mu_1, \mu_2, \ldots, \mu_{2\alpha}\}$, and note that $|T_0| = |\overline{T}| - 2(\frac{\beta}{32}) = \frac{\beta}{16}$. Hence, we have that $\widehat{\mathcal{S}} = \{S_{\mu_1}, S_{\mu_2}, \ldots, S_{\mu_{2\alpha}}\}$ has to cover T_0, where $S_{\mu_j} = S'_{\mu_j}$ for each $1 \leq j \leq 2\alpha$ and the sets $\{S_{\mu_1}, S_{\mu_2}, \ldots, S_{\mu_{2\alpha}}\}$ are chosen independent of T_0 (according to the distribution $\mathcal{D}_{\mathsf{est}}$). We first analyze the probability that $\widehat{\mathcal{S}}$ covers T_0 and then take union bound over all choices of 2α sets from $\mathcal{S}' \setminus \{S'_{i^*}\}$.

Fix any choice of T_0; for each element $k \in T_0$, and for each set $S_j \in \widehat{\mathcal{S}}$, define an indicator random variable $\boldsymbol{X}_k^j \in \{0,1\}$, where $\boldsymbol{X}_k^j = 1$ iff $k \in S_j$. Let $\boldsymbol{X} := \sum_j \sum_k \boldsymbol{X}_k^j$ and notice that:

[15] Note that we do not have to know in what form x appears in the input, as our question is just whether the instance is satisfiable, not to provide a satisfying assignment.

$$\mathrm{E}[\boldsymbol{X}] = \sum_j \sum_k \mathrm{E}[\boldsymbol{X}_k^j] = (2\alpha) \cdot (\frac{\beta}{16}) \cdot (\frac{\beta}{n}) = \frac{\alpha\beta^2}{8n}$$

We have,

$$\Pr\left(\widehat{\mathcal{S}} \text{ covers } T_0\right) \leq \Pr\left(\boldsymbol{X} \geq \frac{\beta}{16}\right) = \Pr\left(\boldsymbol{X} \geq \frac{n}{2\alpha\beta} \cdot \mathrm{E}[\boldsymbol{X}]\right)$$

It is easy to verify that the $\boldsymbol{X}_k^j$ variables are negatively correlated. Hence, applying the extended Chernoff bound[16] due to Panconesi and Srinivasan [41] we get

$$\Pr\left(\boldsymbol{X} \geq \frac{n}{2\alpha\beta} \cdot \mathrm{E}[\boldsymbol{X}]\right) \leq 3\exp\left(\frac{-\epsilon^2\,\mathrm{E}[\boldsymbol{X}]}{3}\right) \text{ where } 1+\epsilon = \frac{n}{2\alpha\beta}$$

Finally, by union bound,

$$\begin{aligned}\Pr(\mathrm{OPT}(\mathcal{S}', T) \leq 2\alpha) &\leq \Pr\left(\exists\, \widehat{\mathcal{S}} \text{ covers } T_0\right) \leq \binom{n}{2\alpha} \cdot 3\exp\left(\frac{-\epsilon^2\,\mathrm{E}[\boldsymbol{X}]}{3}\right) \\ &\leq \exp\left(2\alpha \cdot \log n\right) \cdot 3\exp\left(\frac{-\epsilon^2\,\mathrm{E}[\boldsymbol{X}]}{3}\right)\end{aligned}$$

Since $\alpha = \frac{\beta}{32}$, one can easily check that $\exp\left(\frac{-\epsilon^2\,\mathrm{E}[\boldsymbol{X}]}{3}\right) \leq \exp\left(-3\alpha \cdot \log n\right)$ and hence we have

$$\begin{aligned}\Pr(\mathrm{OPT}(\mathcal{S}', T) \leq 2\alpha) &\leq \Pr\left(\exists\, \widehat{\mathcal{S}} \text{ covers } \overline{T}\right) \leq \binom{n}{2\alpha} \cdot 3\exp\left(\frac{-\epsilon^2\,\mathrm{E}[\boldsymbol{X}]}{3}\right) \\ &\leq \exp\left(2\alpha \cdot \log n\right) \cdot 3\exp\left(\frac{-\epsilon^2\,\mathrm{E}[\boldsymbol{X}]}{3}\right) \\ &\leq \exp\left(2\alpha \cdot \log n\right) \cdot 3\exp\left(-3\alpha \cdot \log n\right) \\ &= o(1)\end{aligned}$$

◀

F.2 The Lower Bound for the Distribution $\mathcal{D}_{\mathsf{est}}$

Observe that distribution $\mathcal{D}_{\mathsf{est}}$ is not a product distribution due to the correlation between the input given to Alice and Bob. However, we can express the distribution as a convex combination of a relatively small set of product distributions. To do so, we need the following definition. For integers k, t and n, a collection P of t subsets of $[n]$ is called a *random* (k,t)*-partition* iff the t sets in P are constructed as follows: Pick k elements from $[n]$, denoted by S, uniformly at random, and partition S randomly into t sets of equal size. We refer to each set in P as a *block*.

[16] Let $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_r$ be a sequence of negatively correlated Boolean random variables, and let $\boldsymbol{X} = \sum_{i=1}^r \boldsymbol{X}_i$. Then $\Pr(|\boldsymbol{X} - \mathrm{E}[\boldsymbol{X}]| \geq \epsilon \cdot \mathrm{E}[\boldsymbol{X}]) \leq 3 \cdot \exp(\frac{-\epsilon^2\,\mathrm{E}[\boldsymbol{X}]}{3})$

An alternative definition of the distribution $\mathcal{D}_{\text{est}}$.

Parameters: $k = 2\beta$ $p = \frac{\beta}{8}$ $t = 16$

1. For any $i \in [n]$, let P_i be a random (k, t)-partition in $[n]$ (chosen independently).
2. The input to Alice is $\mathcal{S}' = (S'_1, \ldots, S'_n)$, where for each i we have $S'_i = \{i\} \cup S_i$ and S_i is created by picking $t/2$ blocks from P_i uniformly at random.
3. The input to Bob is a set T where $\overline{T}$ is created by first picking an $i^* \in [n]$ uniformly at random, and then picking a block from P_{i^*} uniformly at random.

To see that the two formulations of the distribution $\mathcal{D}_{\text{est}}$ are indeed equivalent, notice that (i) the input given to Alice in the new formulation is a collection of sets of size β chosen independently and uniformly at random (by the independence of P_i's), and (ii) the complement of the set given to Bob is a set of size $\frac{\beta}{8}$ which, for $i^* \in_R [n]$, with probability half, is chosen uniformly at random from S_{i^*}, and with probability half, is chosen from $[n] \setminus S_{i^*}$ (by the randomness in the choice of each block in P_{i^*}).

Fix any δ-error protocol Π_{DS} (set $\delta = 1/4$) for $\text{DomSet}_{\text{est}}$ on the distribution $\mathcal{D}_{\text{est}}$. Recall that $\mathbf{\Pi}_{\text{DS}}$ denotes the random variable for the concatenation of the message of Alice with the public randomness used in the protocol Π_{DS}. We further use $\boldsymbol{\mathcal{P}} := (\boldsymbol{P}_1, \ldots, \boldsymbol{P}_t)$ to denote the random partitions $(P_1, \ldots, P_t)$, $\boldsymbol{I}$ for the choice of the special index i^*, and $\boldsymbol{\theta}$ for the parameter $\theta \in \{0, 1\}$, whereby $\theta = 0$ iff $\overline{T} \subseteq S_{i^*}$.

We make the following simple observations about the distribution $\mathcal{D}_{\text{est}}$. The proofs are straightforward.

▶ Remark 41. In the distribution $\mathcal{D}_{\text{est}}$,

1. The random variables $\boldsymbol{\mathcal{S}}$, $\boldsymbol{\mathcal{P}}$, and $\mathbf{\Pi}_{\text{DS}}(\boldsymbol{\mathcal{S}})$ are all independent of the random variable $\boldsymbol{I}$.
2. For any $i \in [m]$, conditioned on $\boldsymbol{P}_i = P$, and $\boldsymbol{I} = i$, the random variables $\boldsymbol{S}_i$ and $\overline{\mathbf{T}}$ are independent of each other. Moreover, $\textsc{supp}(\boldsymbol{S}_i)$ and $\textsc{supp}(\overline{\mathbf{T}})$ contain, respectively, $\binom{t}{\frac{t}{2}}$ and t elements and both $\boldsymbol{S}_i$ and $\overline{\mathbf{T}}$ are uniform over their support.
3. For any $i \in [m]$, the random variable $\boldsymbol{S}_i$ is independent of both $\boldsymbol{\mathcal{S}}^{-i}$ and $\boldsymbol{\mathcal{P}}^{-i}$.

Our goal now is to lower bound $\text{ICost}_{\mathcal{D}_{\text{est}}}(\Pi_{\text{DS}})$ and ultimately $\|\Pi_{\text{DS}}\|$. We start by simplifying the expression for $\text{ICost}_{\mathcal{D}_{\text{est}}}(\Pi_{\text{DS}})$.

▶ **Lemma 42.** $\text{ICost}_{\mathcal{D}_{\text{est}}}(\Pi_{DS}) \geq \sum_{i=1}^{n} I(\mathbf{\Pi}_{DS}; \boldsymbol{S}_i \mid \boldsymbol{P}_i)$

Proof. We have,

$$\text{ICost}_{\mathcal{D}_{\text{est}}}(\Pi_{\text{DS}}) = I(\mathbf{\Pi}_{\text{DS}}; \boldsymbol{\mathcal{S}}) \geq I(\mathbf{\Pi}_{\text{DS}}; \boldsymbol{\mathcal{S}} \mid \boldsymbol{\mathcal{P}})$$

where the inequality holds since (i) $H(\mathbf{\Pi}_{\text{DS}}) \geq H(\mathbf{\Pi}_{\text{DS}} \mid \boldsymbol{\mathcal{P}})$ and (ii) $H(\mathbf{\Pi}_{\text{DS}} \mid \boldsymbol{\mathcal{S}}) = H(\mathbf{\Pi}_{\text{DS}} \mid \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}})$ as $\mathbf{\Pi}_{\text{DS}}$ is independent of $\boldsymbol{\mathcal{P}}$ conditioned on $\boldsymbol{\mathcal{S}}$. We now bound the conditional mutual information term in the above equation.

$$
\begin{aligned}
I(\mathbf{\Pi}_{\mathsf{DS}};\boldsymbol{\mathcal{S}}\mid\boldsymbol{\mathcal{P}}) &= \sum_{i=1}^{m} I(\boldsymbol{S}_i;\mathbf{\Pi}_{\mathsf{DS}}\mid\boldsymbol{\mathcal{P}},\boldsymbol{\mathcal{S}}^{<i}) \\
&\qquad\qquad\text{(the chain rule for the mutual information, Claim 46-(5))} \\
&= \sum_{i=1}^{m} H(\boldsymbol{S}_i\mid\boldsymbol{\mathcal{P}},\boldsymbol{\mathcal{S}}^{<i}) - H(\boldsymbol{S}_i\mid\mathbf{\Pi}_{\mathsf{DS}},\boldsymbol{\mathcal{P}},\boldsymbol{\mathcal{S}}^{<i}) \\
&\geq \sum_{i=1}^{m} H(\boldsymbol{S}_i\mid\boldsymbol{P}_i) - H(\boldsymbol{S}_i\mid\mathbf{\Pi}_{\mathsf{DS}},\boldsymbol{P}_i) \\
&= \sum_{i=1}^{m} I(\boldsymbol{S}_i;\mathbf{\Pi}_{\mathsf{DS}}\mid\boldsymbol{P}_i)
\end{aligned}
$$

The inequality holds since:

(i) $H(\boldsymbol{S}_i\mid\boldsymbol{P}_i) = H(\boldsymbol{S}_i\mid\boldsymbol{P}_i,\boldsymbol{\mathcal{P}}^{-i},\boldsymbol{\mathcal{S}}^{<i}) = H(\boldsymbol{S}_i\mid\boldsymbol{\mathcal{P}},\boldsymbol{\mathcal{S}}^{<i})$ because conditioned on $\boldsymbol{P}_i$, $\boldsymbol{S}_i$ is independent of $\boldsymbol{\mathcal{P}}^{-i}$ and $\boldsymbol{\mathcal{S}}^{<i}$ (Remark 41-(3)), hence the equality holds by Claim 46-(3).

(ii) $H(\boldsymbol{S}_i\mid\mathbf{\Pi}_{\mathsf{DS}},\boldsymbol{P}_i) \geq H(\boldsymbol{S}_i\mid\mathbf{\Pi}_{\mathsf{DS}},\boldsymbol{P}_i,\boldsymbol{\mathcal{P}}^{-i},\boldsymbol{\mathcal{S}}^{<i}) = H(\boldsymbol{S}_i\mid\mathbf{\Pi}_{\mathsf{DS}},\boldsymbol{\mathcal{P}},\boldsymbol{\mathcal{S}}^{<i})$ since conditioning reduces the entropy, i.e., Claim 46-(3).

◀

Equipped with Lemma 42, we only need to bound $\sum_{i\in[n]} I(\mathbf{\Pi}_{\mathsf{DS}};\boldsymbol{S}_i\mid\boldsymbol{P}_i)$. Note that,

$$\sum_{i=1}^{n} I(\mathbf{\Pi}_{\mathsf{DS}};\boldsymbol{S}_i\mid\boldsymbol{P}_i) = \sum_{i=1}^{n} H(\boldsymbol{S}_i\mid\boldsymbol{P}_i) - \sum_{i=1}^{n} H(\boldsymbol{S}_i\mid\mathbf{\Pi}_{\mathsf{DS}},\boldsymbol{P}_i) \tag{1}$$

Furthermore, for each $i\in[n]$, $|\textsc{supp}(\boldsymbol{S}_i\mid\boldsymbol{P}_i)| = \binom{t}{\frac{t}{2}}$ and $\boldsymbol{S}_i$ is uniform over its support (Remark 41-(2)); hence, by Claim 46-(1),

$$\sum_{i=1}^{n} H(\boldsymbol{S}_i\mid\boldsymbol{P}_i) = \sum_{i=1}^{n} \log\binom{t}{\frac{t}{2}} = 13.64n \tag{2}$$

since $t=16$. Consequently, we only need to bound $\sum_{i=1}^{n} H(\boldsymbol{S}_i\mid\mathbf{\Pi}_{\mathsf{DS}},\boldsymbol{P}_i)$. In order to do so, we show that Π_{DS} can be used to estimate the value of the parameter θ, and hence we only need to establish a lower bound for the problem of estimating θ.

▶ **Lemma 43.** *Any δ-error protocol Π_{DS} over the distribution $\mathcal{D}_{est}$ can be used to determine the value of θ with error probability $\delta+o(1)$.*

Proof. Alice sends the message $\Pi_{\mathsf{DS}}(\mathcal{S})$ as before. Using this message, Bob can compute an α-estimation of the set cover problem using $\Pi_{\mathsf{DS}}(\mathcal{S})$ and his input. If the estimation is less than 2α, we output $\theta=0$ and otherwise we output $\theta=1$. The bound on the error probability follows from Lemma 11. ◀

▶ Remark 44. We assume that in $\mathsf{DomSet}_{\mathsf{est}}$ over the distribution $\mathcal{D}_{\mathsf{est}}$, Bob is additionally provided with the special index i^*.

Note that this assumption can only make our lower bound stronger since Bob can always ignore this information and solve the original $\mathsf{DomSet}_{\mathsf{est}}$.

Let γ be the function that estimates θ used in Lemma 43; the input to γ is the message given from Alice, the public coins used by the players, the set $\overline{T}$, and (by Remark 44) the special index i^*. We have,

$$\Pr(\gamma(\mathbf{\Pi}_{\mathsf{DS}},\overline{\mathbf{T}},\boldsymbol{I}) \neq \boldsymbol{\theta}) \leq \delta + o(1)$$

Hence, by Fano's inequality (Claim 47),

$$
\begin{aligned}
H_2(\delta + o(1)) &\geq H(\boldsymbol{\theta} \mid \boldsymbol{\Pi}_{\mathsf{DS}}, \overline{\mathbf{T}}, \boldsymbol{I}) \\
&= \mathop{\mathrm{E}}_{i \sim \boldsymbol{I}} \left[H(\boldsymbol{\theta} \mid \boldsymbol{\Pi}_{\mathsf{DS}}, \overline{\mathbf{T}}, \boldsymbol{I} = i) \right] \\
&= \frac{1}{n} \sum_{i=1}^{m} H(\boldsymbol{\theta} \mid \boldsymbol{\Pi}_{\mathsf{DS}}, \overline{\mathbf{T}}, \boldsymbol{I} = i) \qquad (3)
\end{aligned}
$$

We now show that each term above is lower bounded by $H(\boldsymbol{S}_i \mid \boldsymbol{\Pi}_{\mathsf{DS}}, \boldsymbol{P}_i)/t$ and hence we obtain the desired upper bound on $H(\boldsymbol{S}_i \mid \boldsymbol{\Pi}_{\mathsf{DS}}, \boldsymbol{P}_i)$ in Equation (1).

▶ **Lemma 45.** *For any $i \in [n]$, $H(\boldsymbol{\theta} \mid \boldsymbol{\Pi}_{DS}, \overline{\mathbf{T}}, \boldsymbol{I} = i) \geq H(\boldsymbol{S}_i \mid \boldsymbol{\Pi}_{DS}, \boldsymbol{P}_i)/t$.*

Proof. We have,

$$
\begin{aligned}
H(\boldsymbol{\theta} \mid \boldsymbol{\Pi}_{\mathsf{DS}}, \overline{\mathbf{T}}, \boldsymbol{I} = i) &\geq H(\boldsymbol{\theta} \mid \boldsymbol{\Pi}_{\mathsf{DS}}, \overline{\mathbf{T}}, \boldsymbol{P}_i, \boldsymbol{I} = i) \\
&\qquad \text{(conditioning on random variables reduces entropy, Claim 46-(3))} \\
&= \mathop{\mathrm{E}}_{P \sim \boldsymbol{P}_i \mid \boldsymbol{I} = i} \left[H(\boldsymbol{\theta} \mid \boldsymbol{\Pi}_{\mathsf{DS}}, \overline{\mathbf{T}}, \boldsymbol{P}_i = P, \boldsymbol{I} = i) \right]
\end{aligned}
$$

For brevity, let E denote the event $(\boldsymbol{P}_i = P, \boldsymbol{I} = i)$. We can write the above equation as,

$$
H(\boldsymbol{\theta} \mid \boldsymbol{\Pi}_{\mathsf{DS}}, \overline{\mathbf{T}}, \boldsymbol{P}_i, \boldsymbol{I} = i) = \mathop{\mathrm{E}}_{P \sim \boldsymbol{P}_i \mid \boldsymbol{I} = i} \mathop{\mathrm{E}}_{\overline{T} \sim \overline{\mathbf{T}} \mid E} \left[H(\boldsymbol{\theta} \mid \boldsymbol{\Pi}_{\mathsf{DS}}, \overline{\mathbf{T}} = \overline{T}, E) \right]
$$

Note that by Remark 41-(2), conditioned on the event E, $\overline{T}$ is chosen to be one of the blocks of $P = (B_1, \ldots, B_t)$ uniformly at random. Hence,

$$
H(\boldsymbol{\theta} \mid \boldsymbol{\Pi}_{\mathsf{DS}}, \overline{\mathbf{T}}, \boldsymbol{P}_i, \boldsymbol{I} = i) = \mathop{\mathrm{E}}_{P \sim \boldsymbol{P}_i \mid \boldsymbol{I} = i} \left[\sum_{j=1}^{t} \frac{H(\boldsymbol{\theta} \mid \boldsymbol{\Pi}_{\mathsf{DS}}, \overline{\mathbf{T}} = B_j, E)}{t} \right]
$$

Define a random variable $\boldsymbol{X} := (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_t)$, where each $\boldsymbol{X}_j \in \{0, 1\}$ and $\boldsymbol{X}_j = 1$ iff $\boldsymbol{S}_i$ contains the block B_j. Note that conditioned on E, $\boldsymbol{X}$ uniquely determines the set $\boldsymbol{S}_i$. Moreover, notice that conditioned on $\overline{\mathbf{T}} = B_j$ and E, $\boldsymbol{\theta} = 0$ iff $\boldsymbol{X}_j = 1$. Hence,

$$
H(\boldsymbol{\theta} \mid \boldsymbol{\Pi}_{\mathsf{DS}}, \overline{\mathbf{T}}, \boldsymbol{P}_i, \boldsymbol{I} = i) = \mathop{\mathrm{E}}_{P \sim \boldsymbol{P}_i \mid \boldsymbol{I} = i} \left[\sum_{j=1}^{t} \frac{H(\boldsymbol{X}_j \mid \boldsymbol{\Pi}_{\mathsf{DS}}, \overline{\mathbf{T}} = B_j, E)}{t} \right]
$$

Now notice that $\boldsymbol{X}_j$ is independent of the event $\overline{\mathbf{T}} = B_j$ since $\boldsymbol{S}_i$ is chosen independent of $\overline{\mathbf{T}}$ conditioned on E (Remark 41-(2)). Similarly, since $\boldsymbol{\Pi}_{\mathsf{DS}}$ is only a function of $\boldsymbol{S}$ and $\boldsymbol{S}$ is independent of $\overline{\mathbf{T}}$ conditioned on E, $\boldsymbol{\Pi}_{\mathsf{DS}}$ is also independent of the event $\overline{\mathbf{T}} = B_j$.

Consequently, by Claim 46-(6), we can "drop" the conditioning on $\overline{\mathbf{T}} = B_j$,

$$\begin{aligned}
H(\boldsymbol{\theta} \mid \mathbf{\Pi}_{\mathsf{DS}}, \overline{\mathbf{T}}, \boldsymbol{P}_i, \boldsymbol{I} = i) &= \underset{P \sim \boldsymbol{P}_i \mid \boldsymbol{I}=i}{\mathrm{E}} \left[\sum_{j=1}^{t} \frac{H(\boldsymbol{X}_j \mid \mathbf{\Pi}_{\mathsf{DS}}, E)}{t} \right] \\
&\geq \underset{P \sim \boldsymbol{P}_i \mid \boldsymbol{I}=i}{\mathrm{E}} \left[\frac{H(\boldsymbol{X} \mid \mathbf{\Pi}_{\mathsf{DS}}, E)}{t} \right] \\
&\qquad \text{(sub-additivity of the entropy, Claim 46-(4))} \\
&= \underset{P \sim \boldsymbol{P}_i \mid \boldsymbol{I}=i}{\mathrm{E}} \left[\frac{H(\boldsymbol{S}_i \mid \mathbf{\Pi}_{\mathsf{DS}}, E)}{t} \right] \\
&\qquad \text{($\boldsymbol{S}_i$ and $\boldsymbol{X}$ uniquely define each other conditioned on E)} \\
&= \underset{P \sim \boldsymbol{P}_i \mid \boldsymbol{I}=i}{\mathrm{E}} \left[\frac{H(\boldsymbol{S}_i \mid \mathbf{\Pi}_{\mathsf{DS}}, \boldsymbol{P}_i = P, \boldsymbol{I} = i)}{t} \right] \\
&\qquad \text{(E is defined as $(\boldsymbol{P}_i = P, \boldsymbol{I} = i)$)} \\
&= \frac{H(\boldsymbol{S}_i \mid \mathbf{\Pi}_{\mathsf{DS}}, \boldsymbol{P}_i, \boldsymbol{I} = i)}{t}
\end{aligned}$$

Finally, by Remark 41-(1), $\boldsymbol{S}_i$, $\mathbf{\Pi}_{\mathsf{DS}}$, and $\boldsymbol{P}_i$ are all independent of the event $\boldsymbol{I} = i$, and hence by Claim 46-(6), $H(\boldsymbol{S}_i \mid \mathbf{\Pi}_{\mathsf{DS}}, \boldsymbol{P}_i, \boldsymbol{I} = i) = H(\boldsymbol{S}_i \mid \mathbf{\Pi}_{\mathsf{DS}}, \boldsymbol{P}_i)$, which concludes the proof. ◀

By plugging in the bound from Lemma 45 in Equation (3) we have,

$$\sum_{i=1}^{n} H(\boldsymbol{S}_i \mid \mathbf{\Pi}_{\mathsf{DS}}, \boldsymbol{P}_i) \leq H_2(\delta + o(1)) \cdot (nt) = 0.812 \times (16n) = 12.992n$$

since $\delta = 1/4$ and $t = 16$. Finally, by plugging in this bound together with the bound from Equation (2) in Equation (1), we get,

$$\sum_{i=1}^{n} I(\mathbf{\Pi}_{\mathsf{DS}}; \boldsymbol{S}_i \mid \boldsymbol{P}_i) \geq 0.64n$$

By Lemma 42,

$$\mathsf{IC}^{1/4}_{\mathcal{D}_{\mathsf{est}}}(\mathsf{DomSet}_{\mathsf{est}}) = \min_{\Pi_{\mathsf{DS}}} \left(\mathsf{ICost}_{\mathcal{D}_{\mathsf{est}}}(\Pi_{\mathsf{DS}}) \right) = \Omega(n)$$

To conclude, since the information complexity is a lower bound on the communication complexity (see Proposition 52), we obtain a lower bound of $\Omega(n)$ for $\mathsf{DomSet}_{\mathsf{est}}$ over the distribution $\mathcal{D}_{\mathsf{est}}$. This completes the proof of Theorem 10

G Prerequisites for proof of Theorem 10

In this section, we provide the necessary prerequisites needed in the proof of Theorem 10. The material below is taken as from [2].

G.1 Tools from Information Theory

We briefly review some basic concepts from information theory needed for establishing our lower bounds. For a broader introduction to the field, we refer the reader to the excellent text by Cover and Thomas [17].

In the following, we denote the *Shannon Entropy* of a random variable $\boldsymbol{A}$ by $H(\boldsymbol{A})$ and the *mutual information* of two random variables $\boldsymbol{A}$ and $\boldsymbol{B}$ by $I(\boldsymbol{A};\boldsymbol{B}) = H(\boldsymbol{A}) - H(\boldsymbol{A} \mid \boldsymbol{B}) = H(\boldsymbol{B}) - H(\boldsymbol{B} \mid \boldsymbol{A})$. If the distribution $\mathcal{D}$ of the random variables is not clear from the context, we use $H_{\mathcal{D}}(\boldsymbol{A})$ (resp. $I_{\mathcal{D}}(\boldsymbol{A};\boldsymbol{B})$). We use H_2 to denote the binary entropy function where for any real number $0 < \delta < 1$, $H_2(\delta) = \delta \log \frac{1}{\delta} + (1-\delta) \log \frac{1}{1-\delta}$.

We use the following basic properties of entropy and mutual information (proofs can be found in [17, Chapter 2]).

▷ Claim 46. Let $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ be three random variables.

1. $0 \leq H(\boldsymbol{A}) \leq |\boldsymbol{A}|$. $H(\boldsymbol{A}) = |\boldsymbol{A}|$ iff $\boldsymbol{A}$ is uniformly distributed over its support.
2. $I(\boldsymbol{A};\boldsymbol{B}) \geq 0$. The equality holds iff $\boldsymbol{A}$ and $\boldsymbol{B}$ are *independent*.
3. *Conditioning on a random variable reduces entropy*: $H(\boldsymbol{A} \mid \boldsymbol{B},\boldsymbol{C}) \leq H(\boldsymbol{A} \mid \boldsymbol{B})$. The equality holds iff $\boldsymbol{A}$ and $\boldsymbol{C}$ are independent conditioned on $\boldsymbol{B}$.
4. *Subadditivity of entropy*: $H(\boldsymbol{A},\boldsymbol{B} \mid \boldsymbol{C}) \leq H(\boldsymbol{A} \mid \boldsymbol{C}) + H(\boldsymbol{B} \mid \boldsymbol{C})$.
5. *The chain rule for mutual information*: $I(\boldsymbol{A},\boldsymbol{B};\boldsymbol{C}) = I(\boldsymbol{A};\boldsymbol{C}) + I(\boldsymbol{B};\boldsymbol{C} \mid \boldsymbol{A})$.
6. For any event E independent of $\boldsymbol{A}$ and $\boldsymbol{B}$, $H(\boldsymbol{A} \mid \boldsymbol{B}, E) = H(\boldsymbol{A} \mid \boldsymbol{B})$.
7. For any event E independent of $\boldsymbol{A},\boldsymbol{B}$ and $\boldsymbol{C}$, $I(\boldsymbol{A};\boldsymbol{B} \mid \boldsymbol{C}, E) = I(\boldsymbol{A};\boldsymbol{B} \mid \boldsymbol{C})$.

The following claim (Fano's inequality) states that if a random variable $\boldsymbol{A}$ can be used to estimate the value of another random variable $\boldsymbol{B}$, then $\boldsymbol{A}$ should "consume" most of the entropy of $\boldsymbol{B}$.

▷ Claim 47 (Fano's inequality). For any binary random variable $\boldsymbol{B}$ and any (possibly randomized) function f that predicts $\boldsymbol{B}$ based on $\boldsymbol{A}$, if $\Pr(\mathrm{f}(\boldsymbol{A}) \neq \boldsymbol{B}) = \delta$, then $H(\boldsymbol{B} \mid \boldsymbol{A}) \leq H_2(\delta)$.

We also use the following simple claim, which states that conditioning on independent random variables can only increase the mutual information.

▷ Claim 48. For any random variables $\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}$, and $\boldsymbol{D}$, if $\boldsymbol{A}$ and $\boldsymbol{D}$ are independent conditioned on $\boldsymbol{C}$, then $I(\boldsymbol{A};\boldsymbol{B} \mid \boldsymbol{C}) \leq I(\boldsymbol{A};\boldsymbol{B} \mid \boldsymbol{C},\boldsymbol{D})$.

Proof. Since $\boldsymbol{A}$ and $\boldsymbol{D}$ are independent conditioned on $\boldsymbol{C}$, by Claim 46-(3), $H(\boldsymbol{A} \mid \boldsymbol{C}) = H(\boldsymbol{A} \mid \boldsymbol{C},\boldsymbol{D})$ and $H(\boldsymbol{A} \mid \boldsymbol{C},\boldsymbol{B}) \geq H(\boldsymbol{A} \mid \boldsymbol{C},\boldsymbol{B},\boldsymbol{D})$. We have,

$$\begin{aligned} I(\boldsymbol{A};\boldsymbol{B} \mid \boldsymbol{C}) &= H(\boldsymbol{A} \mid \boldsymbol{C}) - H(\boldsymbol{A} \mid \boldsymbol{C},\boldsymbol{B}) = H(\boldsymbol{A} \mid \boldsymbol{C},\boldsymbol{D}) - H(\boldsymbol{A} \mid \boldsymbol{C},\boldsymbol{B}) \\ &\leq H(\boldsymbol{A} \mid \boldsymbol{C},\boldsymbol{D}) - H(\boldsymbol{A} \mid \boldsymbol{C},\boldsymbol{B},\boldsymbol{D}) = I(\boldsymbol{A};\boldsymbol{B} \mid \boldsymbol{C},\boldsymbol{D}) \end{aligned}$$

◀

G.2 Communication Complexity and Information Complexity

Communication complexity and information complexity play an important role in our lower bound proofs. We now provide necessary definitions for completeness.

G.2.1 Communication complexity.

Our lowers bounds for single-pass streaming algorithms are established through communication complexity lower bounds. Here, we briefly provide some context necessary for our purpose; for a more detailed treatment of communication complexity, we refer the reader to the excellent text by Kushilevitz and Nisan [37].

We focus on the *two-player one-way communication* model. Let P be a relation with domain $\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$. Alice receives an input $X \in \mathcal{X}$ and Bob receives $Y \in \mathcal{Y}$, where (X, Y) are chosen from a joint distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. In addition to private randomness, the players also have an access to a shared public tape of random bits R. Alice sends a single message $M(X, R)$ and Bob needs to output an answer $Z := Z(M(X, R), Y, R)$ such that $(X, Y, Z) \in P$.

We use Π to denote a protocol used by the players. Unless specified otherwise, we always assume that the protocol Π can be randomized (using both public and private randomness), *even against a prior distribution* $\mathcal{D}$ *of inputs*. For any $0 < \delta < 1$, we say Π is a δ-error protocol for P over a distribution $\mathcal{D}$, if the probability that for an input (X, Y), Bob outputs some Z where $(X, Y, Z) \notin P$ is at most δ (the probability is taken over the randomness of both the distribution and the protocol).

▶ **Definition 49.** *The* communication cost *of a protocol* Π *for a problem* P *on an input distribution* $\mathcal{D}$*, denoted by* $\|\Pi\|$*, is the worst-case size of the message sent from Alice to Bob in the protocol* Π*, when the inputs are chosen from the distribution* $\mathcal{D}$*.*
The communication complexity $\mathsf{CC}^{\delta}_{\mathcal{D}}(P)$ *of a problem* P *with respect to a distribution* $\mathcal{D}$ *is the minimum communication cost of a* δ*-error protocol* Π *over* $\mathcal{D}$*.*

G.2.2 Information complexity

Our definition is tuned specifically for *one-way protocols*, similar in the spirit of [3, 36].

▶ **Definition 50.** *Consider an input distribution* $\mathcal{D}$ *and a protocol* Π *(for some problem* P*). Let* $\boldsymbol{X}$ *be the random variable for the input of Alice drawn from* $\mathcal{D}$*, and let* $\boldsymbol{\Pi} := \boldsymbol{\Pi}(\boldsymbol{X})$ *be the random variable denoting the message sent from Alice to Bob* concatenated *with the public randomness* $\boldsymbol{R}$ *used by* Π*. The information cost* $\mathsf{ICost}_{\mathcal{D}}(\Pi)$ *of a one-way protocol* Π *with respect to* $\mathcal{D}$ *is* $I_{\mathcal{D}}(\boldsymbol{\Pi}; \boldsymbol{X})$*.*
The information complexity $\mathsf{IC}^{\delta}_{\mathcal{D}}(P)$ *of* P *with respect to a distribution* $\mathcal{D}$ *is the minimum* $\mathsf{ICost}_{\mathcal{D}}(\Pi)$ *taken over all one-way* δ*-error protocols* Π *for* P *over* $\mathcal{D}$*.*

Note that any public coin protocol is a distribution over private coins protocols, run by first using public randomness to sample a random string $\boldsymbol{R} = R$ and then running the corresponding private coin protocol Π^R. We also use $\boldsymbol{\Pi}^R$ to denote the random variable of the message sent from Alice to Bob, assuming that the public randomness is $\boldsymbol{R} = R$. We have the following well-known claim.

▷ Claim 51. For any distribution $\mathcal{D}$ and any protocol Π, let $\boldsymbol{R}$ denote the public randomness used in Π; then, $\mathsf{ICost}_{\mathcal{D}}(\Pi) = \mathrm{E}_{R\sim\boldsymbol{R}}\left[I_{\mathcal{D}}(\boldsymbol{\Pi}^R; \boldsymbol{X} \mid \boldsymbol{R} = R)\right]$.

Proof. Let $\boldsymbol{\Pi} = (\boldsymbol{M}, \boldsymbol{R})$, where $\boldsymbol{M}$ denotes the message sent by Alice and $\boldsymbol{R}$ is the public randomness. We have,

$$\mathsf{ICost}_{\mathcal{D}}(\Pi) = I(\boldsymbol{\Pi}; \boldsymbol{X}) = I(\boldsymbol{M}, \boldsymbol{R}; \boldsymbol{X}) = I(\boldsymbol{R}; \boldsymbol{X}) + I(\boldsymbol{M}; \boldsymbol{X} \mid \boldsymbol{R})$$

(the chain rule for mutual information, Claim 46-(5))

$$= \mathop{\mathrm{E}}_{R\sim\boldsymbol{R}}\left[I_{\mathcal{D}}(\boldsymbol{\Pi}^R; \boldsymbol{X} \mid \boldsymbol{R} = R)\right]$$

($\boldsymbol{M} = \boldsymbol{\Pi}^R$ whenever $\boldsymbol{R} = R$ and $I(\boldsymbol{R}; \boldsymbol{X}) = 0$ by Claim 46-(2))

◀

The following well-known proposition (see, e.g., [7]) relates communication complexity and information complexity.

▶ **Proposition 52.** *For every* $0 < \delta < 1$ *and every distribution* $\mathcal{D}$*:* $\mathsf{CC}^{\delta}_{\mathcal{D}}(P) \geq \mathsf{IC}^{\delta}_{\mathcal{D}}(P)$.

Proof. Let Π be a protocol with the minimum communication complexity for P on $\mathcal{D}$ and $\boldsymbol{R}$ denotes the public randomness of Π; using Claim 51, we can write,

$$\begin{aligned}\mathsf{IC}^{\delta}_{\mathcal{D}}(P) &= \mathop{\mathrm{E}}_{R\sim\boldsymbol{R}}\left[I_{\mathcal{D}}(\boldsymbol{\Pi}^{R};\boldsymbol{X} \mid \boldsymbol{R}=R)\right] \leq \mathop{\mathrm{E}}_{R\sim\boldsymbol{R}}\left[H_{\mathcal{D}}(\boldsymbol{\Pi}^{R} \mid \boldsymbol{R}=R)\right] \\ &\leq \mathop{\mathrm{E}}_{R\sim\boldsymbol{R}}\left[\left|\boldsymbol{\Pi}^{R}\right|\right] \leq \|\Pi\| = \mathsf{CC}^{\delta}_{\mathcal{D}}(P)\end{aligned}$$

◀