Polynomial Pass Lower Bounds for Graph Streaming Algorithms*

Sepehr Assadi[†]
Princeton University
Princeton, NJ, USA

Yu Chen[‡]
University of Pennsylvania
Philadelphia, PA, USA

Sanjeev Khanna[‡]
University of Pennsylvania
Philadelphia, PA, USA

ABSTRACT

We present new lower bounds that show that a polynomial number of passes are necessary for solving some fundamental graph problems in the streaming model of computation. For instance, we show that any streaming algorithm that finds a weighted minimum s-t cut in an n-vertex undirected graph requires $n^{2-o(1)}$ space unless it makes $n^{\Omega(1)}$ passes over the stream.

To prove our lower bounds, we introduce and analyze a new four-player communication problem that we refer to as the *hidden-pointer chasing* problem. This is a problem in spirit of the standard pointer chasing problem with the key difference that the pointers in this problem are hidden to players and finding each one of them requires solving another communication problem, namely the set intersection problem. Our lower bounds for graph problems are then obtained by reductions from the hidden-pointer chasing problem.

Our hidden-pointer chasing problem appears flexible enough to find other applications and is therefore interesting in its own right. To showcase this, we further present an interesting application of this problem beyond streaming algorithms. Using a reduction from hidden-pointer chasing, we prove that any algorithm for submodular function minimization needs to make $n^{2-o(1)}$ value queries to the function unless it has a polynomial degree of adaptivity.

CCS CONCEPTS

• **Theory of computation → Streaming, sublinear and near linear time algorithms**; *Graph algorithms analysis*; *Lower bounds and information complexity*.

KEYWORDS

Graph streaming, Lower bounds, Communication complexity

*A full version of the paper including all the missing proofs is available on arXiv [10].

[†]Supported in part by the Simons Collaboration on Algorithms and Geometry. Majority of work done while the author was a graduate student at University of Pennsylvania and was supported in part by the National Science Foundation grant CCF-1617851. Email: sassadi@princeton.edu.

[‡]Supported in part by the National Science Foundation grants CCF-1617851 and CCF-1763514. Email: {chenyu2,sanjeev}@cis.upenn.edu.

STOC '19, June 23–26, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6705-9/19/06…$15.00
https://doi.org/10.1145/3313276.3316361

ACM Reference Format:
Sepehr Assadi, Yu Chen, and Sanjeev Khanna. 2019. Polynomial Pass Lower Bounds for Graph Streaming Algorithms. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC '19), June 23–26, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3313276.3316361

1 INTRODUCTION

Graph streaming algorithms are algorithms that solve computational problems on graphs, say, finding a maximum matching, when the input is presented as a sequence of edges, under the usual constraints of the streaming model, namely sequential access to the stream and limited memory. Formally, in the graph streaming model, the edges of a graph $G(V,E)$ are presented one by one in an arbitrary order. The algorithm can make one or a limited number of sequential passes over this stream, while using a small memory to process the graph, preferably $O(n \cdot \text{polylog}(n))$ memory, referred to as *semi-streaming* restriction [58] (n is the number of vertices in G).

It turns out allowing for multiple passes over the stream greatly enhances the capability of graph streaming algorithms. A striking example is the (global) minimum cut problem: While $\Omega(n^2)$ space is needed for computing an exact minimum cut in a single pass [113], a recent result of [104] implies that a minimum cut of an undirected unweighted graph can be computed in $\widetilde{O}(n)$ space in only two passes over the stream[1]. Several other examples of this phenomenon include algorithms for triangle counting [29, 87], approximate matching [82, 93], single-source shortest path [28, 59], maximal independent set [11, 61], and minimum dominating set [13, 70].

Multi-pass streaming algorithms have been gaining increasing attention in recent years and for many well-studied graph problems, space efficient algorithms have been designed that use at most a logarithmic number of passes (see, e.g. [3, 4, 28, 29, 41, 52, 58, 66, 70, 73, 80, 82–84, 93, 95, 106]). But for many other problems, such results have proved elusive. Examples include shortest path and diameter computation [89], random walks [90], and directed reachability and maximum flow [94] (see also [91]). At the same time, known techniques for proving streaming lower bounds are unable to prove essentially any bounds beyond logarithmic number of passes (see Section 1.1 for an exception to this rule and the inherent limitation behind it). For example, the best known lower bounds for several key problems such as shortest path, directed reachability, and perfect matchings, only imply $\Omega(\frac{\log n}{\log\log n})$ passes for semi-streaming algorithms [59, 66], while none of these problems so far admit an algorithm with $n^{2-\Omega(1)}$ space and $n^{o(1)}$ passes.

[1] The result of [104] is not stated as a streaming algorithm. However, the algorithm in [104] combined with the known graph streaming algorithms for cut sparsifiers (see, e.g. [94]) immediately imply the claimed result.

Our goal in this paper is to remedy this situation by **presenting new tools for proving stronger multi-pass graph streaming lower bounds**. To better understand the challenges along the way, we first briefly revisit the current state-of-affairs.

1.1 Landscape of Graph Streaming Lower Bounds

A vast body of work in graph streaming lower bounds concerns algorithms that make only one or a few passes over the stream. Examples of single-pass lower bounds include the ones for diameter [59], approximate matchings [14, 15, 62, 82], exact minimum/maximum cuts [113], and maximal independent sets [11, 46]. Examples of multi-pass lower bounds include the ones for BFS trees [59], perfect matchings [66], shortest path [66], and minimum vertex cover and dominating set [70]. These lower bounds are almost always obtained by considering communication complexity of the problem with *limited number of rounds* of communication which gives a lower bound on the space complexity of streaming algorithms with proportional number of passes to the limits on rounds of communication (see e.g. [6, 65]). The communication lower bounds are then typically proved via reductions from (variants of) the *pointer chasing* problem [39, 100, 101] for multi-pass lower bounds and the *indexing* problem [2, 85] and *boolean hidden (hyper-)matching* problem [60, 108] for single-pass lower bounds.

In the pointer chasing problem, Alice and Bob are given functions $f, g : [n] \to [n]$ and the goal is to compute $f(g(\cdots f(g(0))))$ for k iterations. Computing this function in less than k rounds requires $\widetilde{\Omega}(n/k)$ communication [112] (see also [51, 100–102]). The reductions from pointer chasing to graph streaming lower bounds are based on using vertices of the graph to encode $[n]$ and each edge to encode a pointer [59, 66]. Directly using pointer chasing does not imply lower bounds stronger than $\Omega(n)$ and hence variants of pointer chasing with multiple pointers such as multi-valued pointer chasing [59, 77] and set pointer chasing [66], were considered. Using multiple pointers however has the undesired side effect that the lower bound deteriorates exponentially with number of rounds. As such, these lower bounds do not go beyond $O(\log n)$ passes even for algorithms with $O(n)$ space.

There are however a number of results that prove lower bounds for a very large number of passes (even close to n). Examples include lower bounds for approximating clique and independent set [69], approximating dominating set [9], computing girth [59], estimating the number of triangles [25, 29, 47, 79], and finding minimum vertex cover or coloring [1]. These results are all proven by considering the communication complexity of the problem with *no limits on rounds* of communication. Such bounds then imply lower bounds on the product of space and number of passes of streaming algorithms (see, e.g. [6]). The communication lower bounds themselves are proven by reductions from a handful of communication problems, mainly the *set disjointness* problem [16, 24, 81, 103].

This approach suffers from two main drawbacks. Firstly, these lower bounds only exhibit space bounds that scale with the reciprocal of the number of passes and are hence unable to capture more nuanced space/pass trade-offs. More importantly, there is an inherit limitation to this approach since the computational model considered here is much stronger than the streaming model. This means that many problems of interest admit efficient communication protocols in this model and hence one simply cannot prove interesting lower bounds for them. An illustrating example is the directed s-t reachability problem which admits an $O(n)$ communication protocol, ruling out the possibility of essentially any non-trivial lower bound using this approach (even "harder" problems such as maximum matching admit non-trivial protocols with $\widetilde{O}(n^{3/2})$ communication [50, 74]).

1.2 Our Contributions

We introduce and analyze a new communication problem similar in spirit to standard pointer chasing, which we refer to as the *hidden-pointer chasing* (HPC) problem. What differentiate HPC from previous variants of pointer chasing is that the pointers are "hidden" from players and finding each one of them requires solving another communication problem, namely the *set intersection* problem, in which the goal is to *find* the *unique* element in the intersection of players input. We limit ourselves to the following informal definition of HPC here and postpone the formal definition to Section 3.1. There are four players in HPC paired into groups of size two each. Each pair of players inside a group shares n instances of the set intersection problem on n elements. The intersecting element in each instance of each group "points" to an instance in the other group. The goal is to start from a fixed instance and follow these pointers for a fixed number of steps. We prove the following communication complexity lower bound for HPC.

Result 1. *Any r-round protocol that with constant probability finds the $(r+1)$-th pointer in the hidden-pointer chasing problem requires $\Omega(n^2/r^2)$ communication.*

Result 1 implies a new approach towards proving graph streaming lower bounds that sits squarely in the middle of previous methods: HPC is a problem that admits an "efficient" protocol when there is no limit on rounds of communication and yet is "hard" with even a polynomial limitation on number of rounds. We use this result to prove strong pass lower bounds for some fundamental problems in graph streams via reductions from HPC.

Cut and Flow Problems. One of the main applications of Result 1 is the following result.

Result 2. *Any p-pass streaming algorithm that with a constant probability outputs the minimum s-t cut value in a weighted graph (undirected or directed) requires $\Omega(n^2/p^5)$ space.*

Prior to our work, the best lower bound known for this problem was an $n^{1+\Omega(1/p)}$ space lower bound for p-pass algorithms [66] (for weighted undirected graphs and unweighted directed graphs). Result 2 significantly improves upon this. In particular, it implies that $\widetilde{\Omega}(n^{1/5})$ passes are necessary for semi-streaming algorithms, exponentially improving upon the $\Omega(\frac{\log n}{\log\log n})$ lower bound of [66]. At the same time, Result 2 also shows that any streaming algorithm for this problem with a small number of passes, namely polylog(n) passes, requires $\widetilde{\Omega}(n^2)$ space, almost the same space as the trivial single-pass algorithm that stores the input graph entirely.

Our Result 2 should be contrasted with the results of [104] that imply an $\widetilde{O}(n^{5/3})$ space algorithm for unweighted minimum s-t cut on undirected graphs in only *two* passes (see Footnote 1).

By max-flow min-cut theorem, Result 2 also implies identical bounds for computing the value of maximum s-t flow in capacitated graphs, making progress on a question raised in [94] regarding the streaming complexity of maximum flow in directed graphs.

Lexicographically-First Maximal Independent Set. A maximal independent set (MIS) returned by the sequential greedy algorithm that visits the vertices of the graph in their lexicographical order is called the lexicographically-first MIS. We prove the following result for this problem.

Result 3. *Any p-pass streaming algorithm that with constant probability finds a lexicographically first maximal independent set of in a graph requires $\Omega(n^2/p^5)$ space.*

The lexicographically-first MIS has a rich history in computer science and in particular parallel algorithms [5, 30, 44, 92]. However, even though multiple variants of the independent set problem have been studied in the streaming model [11, 45, 46, 61, 67–69], we are not aware of any work on this particular problem (we remark that standard MIS problem admits an $\widetilde{O}(n)$ space $O(\log\log n)$ pass algorithm [61]). Besides being a fundamental problem in its own right, what makes this problem appealing for us is that it nicely illustrates the power of our techniques compared to previous approaches. The lexicographically-first MIS can be computed with $O(n)$ communication in the two-player communication model (or for any constant number of players) with no restriction on number of rounds by a direct simulation of the sequential algorithm. Hence, this problem perfectly fits the class of problems for which previous techniques cannot prove lower bounds beyond logarithmic passes. To our knowledge, this is the first super-logarithmic pass lower bound for any graph problem that admits an efficient protocol with no restriction on number of rounds.

Beyond Graph Streams: An Application to Submodular Minimization. We also use Result 1 to prove query/adaptivity tradeoffs for the submodular function minimization (SFM) problem. In SFM, we have a submodular function $f : 2^{[n]} \to [M]$ and our goal is to find a set $S^* \subseteq [n]$ that minimizes $f(S^*)$ by making value queries to f. SFM has been studied extensively over the years [42, 48, 63, 75, 76, 88, 107], culminating in the currently best algorithms of [88] and [42] with $\widetilde{O}(n^2)$ and $\widetilde{O}(n \cdot M^3)$ queries, respectively. The best lower bound for SFM is $\Omega(n)$ queries [71, 72] and determining the query complexity of this problem remains a fascinating open question [72, 104].

Another question in this area that has received a significant attention in recent years is to understand the query/adaptivity tradeoffs in submodular optimization [17–22, 54–57]. An algorithm for SFM is called *k-adaptive* iff it makes at most k rounds of adaptive queries, where the queries in each round are performed in parallel. We prove the following result using a reduction from HPC.

Result 4. *For any constant $\delta \in (0, 1)$, there exists an $\varepsilon := \varepsilon(\delta)$ in $(0, 1)$ such that any algorithm for submodular function minimization on a universe of size N with query complexity $N^{2-\delta}$ requires at least N^{ε} rounds of adaptive queries to succeed with constant probability.*

The only other adaptivity lower bound for SFM that we are aware of is an exponential lower bound on query complexity of *non-adaptive* algorithms (even for approximation) [21]. However, once we allow even two rounds of adaptivity, no lower bounds better than $\Omega(n)$ queries were known.

1.3 Our Techniques

Our reductions in this paper take a different path than previous pointer chasing based reductions that used edges of the graph to directly encode pointers. In particular, our hidden-pointer chasing problem allows us encode a single pointer among $\Theta(n)$ edges and thus work with graphs with density $\Omega(n^2)$ and still keep a polynomial dependence on number of rounds in the communication lower bound. This results in space lower bounds of the form $n^2/p^{O(1)}$ for p-pass streaming algorithms.

The main technical contribution of our paper is the communication complexity lower bound for HPC in Result 1. This result is proved by combining inductive arguments for round/communication tradeoffs (see, e.g. [100, 112]) with direct-sum arguments for information complexity (see, e.g. [24, 26, 31, 36]) to account for the role of set intersection inside HPC. To make this argument work, we also need to prove a stronger lower bound for set intersection than currently known results (see, e.g. [37]). In particular, we prove that any protocol that can even slightly reduce the "uncertainty" about the intersecting element must have a "large" communication and information complexity.

Our new lower bound for set intersection is also proved using tools from information complexity to reduce this problem to a primitive problem, namely set intersection itself on a universe of size two. This requires a novel argument to handle the protocols for set intersection that reduce the uncertainty about the intersecting element without necessarily making much "progress" on finding this element. Another challenge is that unlike typical direct-sum results in this context, say reducing disjointness to the AND problem; see, e.g. [24, 32, 34, 109], set intersection cannot be decomposed into *independent* instances of the primitive problem (this is similar-in-spirit to challenges in analyzing information complexity of set disjointness on *intersecting* distributions [43, 78] as opposed to (more standard) non-intersecting ones). Finally, we prove a lower bound for the primitive problem using the product structure of Hellinger distance for communication protocols (see, e.g. [24, 109]).

1.4 Further Related Work

Understanding space/pass tradeoffs for streaming algorithms dates all the way back to the early results on median-finding [98] more than four decades ago and has remained a focus of attention since; we refer the reader to [38, 39, 64, 65] and references therein.

A closely related line of work to graph streaming algorithms that have received a significant attention in recent years is on streaming algorithms for submodular optimization and in particular set cover and maximum coverage [9, 12, 13, 27, 40, 41, 49, 53, 70, 86, 96, 105]. Particularly relevant to our work, [41] uses a reduction from the multi-party tree pointer chasing problem [39] to prove an $\Omega(\frac{\log n}{\log\log n})$ pass lower bound for approximating set cover with m sets and n elements using $O(n \cdot \text{poly}\{\log n, \log m\})$ space (this can also be interpreted as a lower bound for the edge-cover problem on hyper-graphs with n vertices and m hyper-edges in the graph streaming model). For the set cover problem, a lower bound of $\Omega(\frac{m \cdot n^{1/\alpha}}{p})$ space for p-pass streaming α-approximation algorithms is established in [9] using a reduction from the set disjointness problem (this can also be interpreted as a lower bound for the dominating set problem on graphs with $n = m$ vertices in the graph streaming model).

Similar-in-spirit round/communication tradeoffs for distributed computation of many graph and related problems have also been studied in the literature [7, 8, 12, 33, 35, 50]. For example, [35] proves an $\Omega(\frac{\log n}{\log\log n})$ round lower bound for protocols with low communication that can approximate matchings in a communication model in which players correspond to vertices of an n-vertex graph. Similarly, [12] proves an $\Omega(\frac{\log n}{\log\log n})$ round lower bound for constrained submodular maximization in a communication model where n elements of a universe are partitioned between the players.

Adaptivity lower bounds for submodular optimization [17–22, 54–57] is another topic related to our work. For example, [22] proves that $\Omega(\frac{\log n}{\log\log n})$ rounds of adaptivity are necessary for constrained submodular maximization with polynomial query complexity. Additionally, [21] proved that no non-adaptive algorithm can obtain a better than $1/2$ approximation to submodular minimization with polynomially many queries. Finally, if one goes (way) beyond submodular optimization and considers minimizing a non-smooth convex function, then an $\widetilde{\Omega}(n^{1/3})$ lower bound on rounds of adaptivity is known for any algorithm that makes polynomially many queries [23, 99].

Organization. The rest of the paper is organized as follows. We set up our notation in Section 2. Section 3 contains a detailed technical overview of our approach, including the definition of the hidden-pointer chasing (HPC) problem (Section 3.1), a sketch of the reduction from HPC for proving Result 2 (Section 3.1), and the proof sketch of the communication lower bounds for HPC (Section 3.3) and (a new variant of) set intersection (Section 3.4). Finally, Section 4, presents the proof of Result 1 which is the main technical result of this paper. Due to space limitations, we only present the high level overview of the proofs here and postpone most of the formal arguments to the full version of the paper [10].

2 PRELIMINARIES

Notation. For any integer a, we define $[a] := \{1, \ldots, a\}$. For a tuple $(X_1, \ldots, X_n)$ and integer $i \in [n]$, $X^{<i} := (X_1, \ldots, X_{i-1})$ and $X_{-i} := (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n)$. We use capital 'san-serif' font to denote the random variables, e.g. X. $\mathcal{U}_S$ denotes the uniform distribution over S.

For random variables X, Y, $\mathbb{H}(\mathsf{X})$ denotes the Shannon entropy of X and $\mathbb{I}(\mathsf{X}\,;\mathsf{Y})$ denotes the mutual information. For distributions μ, ν, $\mathbb{D}(\mu \,||\, \nu)$ denotes the KL-divergence, $\Delta_{\mathsf{TV}}(\mu, \nu)$ denotes the total variation distance, and $\mathrm{h}(\mu, \nu)$ denotes the Hellinger distance. Necessary background on information theory, including the definitions and basic tools, is provided in the full version of the paper [10]

Communication Complexity and Information Complexity. We consider the standard communication model of Yao [110]. We use π to denote the protocol used by players and use $\mathsf{CC}(\pi)$ to denote the *communication cost* of π defined as the worst-case bit-length of the messages communicated between the players. We further use *internal information cost* [26] for protocols that measures the average amount of information each player learns about the input of the other in the protocol, defined formally as follows. Consider an input distribution $\mathcal{D}$ and a protocol π. Let $(\mathsf{X}, \mathsf{Y}) \sim \mathcal{D}$ and Π denote the random variables for the inputs and the transcript of the protocol (including the public randomness). The *information cost* of π with respect to $\mathcal{D}$ is $\mathsf{IC}_{\mathcal{D}}(\pi) := \mathbb{I}_{\mathcal{D}}(\Pi\,;\mathsf{X} \mid \mathsf{Y}) + \mathbb{I}_{\mathcal{D}}(\Pi\,;\mathsf{Y} \mid \mathsf{X})$. As one bit of communication can only reveal one bit of information, information cost of a protocol lower bounds its communication cost (see, e.g. [36] or the full version of the paper [10]).

We provide further relevant background and definitions on communication complexity and information complexity in the full version of the paper [10].

Set Intersection Problem. We use the set intersection problem in construction of our HPC problem. Set intersection (Set-Int) is a two-player communication problem in which Alice and Bob are given sets A and B from $[n]$, respectively, with the promise that there exists a unique element t such that $\{t\} = A \cap B$. The goal is for players to find the *target element* t. An $\Omega(n)$ communication lower bound for Set-Int follows directly from lower bounds for set disjointness [24, 32, 34, 81, 103]; see, e.g. [37] (this lower bound by itself is however not useful for our application).

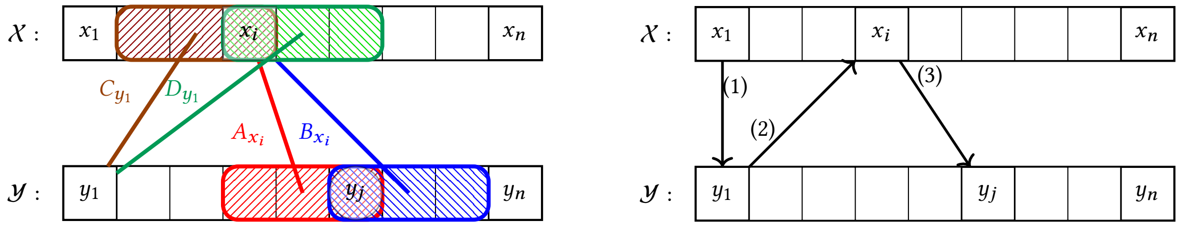
3 TECHNICAL OVERVIEW

We start with defining the hidden-pointer chasing (HPC) problem and briefly discuss a reduction from HPC that establishes the lower bound for minimum cut problem in Result 2. We then sketch the proof of the communication lower bound for HPC in Result 1. Along the way, we also present a new lower bound for set intersection that is needed for establishing Result 1. We emphasize that this section oversimplifies many details and the discussions will be informal for the sake of intuition.

3.1 The Hidden-Pointer Chasing Problem

The hidden-pointer chasing (HPC) problem is a four-party communication problem with players P_A, P_B, P_C, and P_D. Let $\mathcal{X} := \{x_1, \ldots, x_n\}$ and $\mathcal{Y} := \{y_1, \ldots, y_n\}$ be two disjoint universes.

(1) For any $x \in \mathcal{X}$, P_A and P_B are given an instance (A_x, B_x) of Set-Int over the universe $\mathcal{Y}$ where $A_x \cap B_x = \{t_x\}$ for $t_x \in \mathcal{Y}$.
(2) Similarly, for any $y \in \mathcal{Y}$, P_C and P_D are given an instance (C_y, D_y) of Set-Int over the universe $\mathcal{X}$ where $C_y \cap D_y = \{t_y\}$ for $t_y \in \mathcal{X}$.
(3) We define two mappings $f_{AB} : \mathcal{X} \to \mathcal{Y}$ and $f_{CD} : \mathcal{Y} \to \mathcal{X}$ such that:
    (a) for any $x \in \mathcal{X}$, $f_{AB}(x) = t_x \in \mathcal{Y}$ in the instance (A_x, B_x) of Set-Int.
    (b) for any $y \in \mathcal{Y}$, $f_{CD}(y) = t_y \in \mathcal{X}$ in the instance (C_y, D_y) of Set-Int.
(4) Let $x_1 \in \mathcal{X}$ be an arbitrary fixed element of $\mathcal{X}$ known to all players. The pointers $z_0, z_1, z_2, z_3, \ldots$ are defined inductively as follows: $z_0 := x_1, z_1 := f_{AB}(z_0), z_2 := f_{CD}(z_1), z_3 := f_{AB}(z_2), \cdots$.



(a) The sets $A_{x_i}, B_{x_i} \subseteq \mathcal{Y}$ of P_A and P_B for $x_i \in \mathcal{X}$, and $C_{y_1}, D_{y_1} \subseteq \mathcal{X}$ of P_C and P_D for $y_1 \in \mathcal{Y}$.

(b) $z_0 = x_1, z_1 = y_1, z_2 = x_i, z_3 = y_j$, implying that the answer to HPC_3 in this example is y_j.

Figure 1: Illustration of the HPC problem.

The k-*step hidden-pointer chasing* problem (HPC_k) is defined as the communication problem of finding the pointer z_k. See Figure 1 for an illustration.

We define a *phase* (similar to a round) for protocols that solve HPC. In an odd (resp. even) phase, only P_C and P_D (resp. P_A and P_B) are allowed to communicate with each other, and the phase ends once a message is sent to P_A or P_B (resp. P_C or P_D). A protocol is called a k-*phase* protocol iff it uses at most k phases.

It is easy to see that in $k+1$ phases, we can compute HPC_k with $O(k \cdot n)$ total communication by solving the Set-Int instances corresponding to $z_0, z_1, \ldots, z_k$ one at a time in each phase. We prove that if we only have k phases however, solving HPC_k requires a large communication.

Theorem 1. *Any k-phase protocol that outputs the correct solution to HPC_k with constant probability requires $\Omega(n^2/k^2 + n)$ bits of communication.*

We give a proof sketch of the $\Omega(n^2/k^2)$ term in Theorem 1 in Section 3.3 (the $\Omega(n)$ term follows immediately from set intersection lower bound). Before that, we show an application of this result in proving graph streaming lower bounds to illustrate our general approach.

3.2 A Streaming Lower Bound for Minimum Weighted s-t Cut Problem

We sketch the proof of Result 2 for directed graphs in this section. The proof is by a reduction from HPC. We show how to turn any instance of HPC_k for $k \geq 1$ into a weighted directed graph G such that the minimum s-t cut weight in G determines the pointer z_k in HPC_k. The rest of the proof then follows by standard arguments that relate communication complexity to space complexity of streaming algorithms. For the purpose of this proof, it would be more convenient to consider the maximum s-t flow problem instead and then use min-cut max-flow duality.

The high level construction of G is as follows. The vertices in graph G consists of $k+1$ layers each of size n plus source and sink vertices s and t. The even layers of this graph correspond to elements in $\mathcal{X}$ while the odd layers correspond to $\mathcal{Y}$. The edges between the layers are then created by using the sets in the instances of Set-Int inside the HPC_k problem. The idea is to place the edges such that each vertex corresponding to x_i (resp. y_i) in an even layer (resp. odd layer) can send a "larger" flow to the vertex corresponding to the target element of the instance (A_{x_i}, B_{x_i}) (resp. target element of (C_{y_i}, D_{y_i})) than any other vertex in the next layer. By choosing the weight of edges carefully and adding some extra gadgets, we ensure that the maximum s-t flow should route the flow from s along the path that corresponds to pointers $z_0, z_1, \ldots, z_k$. The vertices in the last layer have capacities that encode their identity and hence the maximum s-t flow value in this graph reveals the identity of z_k, thus solving HPC_k. See Figure 2 for an illustration.

It is now easy to show that any $(k/3)$-pass streaming algorithm for minimum weighted s-t cut with space S can be turned into a k-phase protocol for HPC_k with communication cost $O(k \cdot S)$ using this reduction. As the graph G constructed above has $O(k \cdot n)$ vertices, we obtain the desired lower bound in Result 2 by the communication complexity lower bound for HPC in Theorem 1.

The formal proof of Result 2 as well as the other reductions that establish Results 3 and 4 appear in the full version of the paper [10].

3.3 Communication Complexity of Hidden-Pointer Chasing

We now sketch the proof of Theorem 1 which is the main technical contribution of this paper. Let $\mathcal{D}_{\mathsf{SI}}$ be a hard distribution on instances (A, B) for Set-Int. In this distribution A and B are each sets of size almost $n/3$ such that they intersect in a unique element in the universe chosen uniformly at random. We define the distribution $\mathcal{D}_{\mathsf{HPC}}$ over inputs of HPC as the distribution in which all instances (A_x, B_x) and (C_y, D_y) for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ are sampled independently from $\mathcal{D}_{\mathsf{SI}}$ (note that $\mathcal{D}_{\mathsf{HPC}}$ is not a product distribution as $\mathcal{D}_{\mathsf{SI}}$ is not a product distribution).

Fix any k-phase deterministic protocol π_{HPC} for HPC_k throughout and suppose towards a contradiction that $\mathsf{CC}(\pi_{\mathsf{HPC}}) = o(n^2/k^2)$ (the lower bound extends to randomized protocols by Yao's minimax principle [111]). For any $j \in [k]$, we define Π_j as the set of all messages communicated by π_{HPC} in phase j and $\Pi := (\Pi_1, \ldots, \Pi_k)$ as the transcript of the protocol π_{HPC}. We further define $Z = (z_1, \ldots, z_k)$, $E_j := (\Pi^{<j}, Z^{<j})$ for any $j > 1$, and $E_1 = z_0$. We think of E_j as the information "easily known" to players at the beginning of phase j. The main step of the proof of Theorem 1 is the following key lemma which we prove inductively.

Lemma 3.1 (Informal). *For all $j \in [k]$:*

$$\mathop{\mathbb{E}}_{(E_j, \Pi_j)} \left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{Z}_j \mid E_j, \Pi_j), \mathrm{dist}(\mathsf{Z}_j)) \right] = o(1).$$

Lemma 3.1 states that if the communication cost of a protocol is "small", i.e., is $o(n^2/k^2)$, then even after communicating the messages

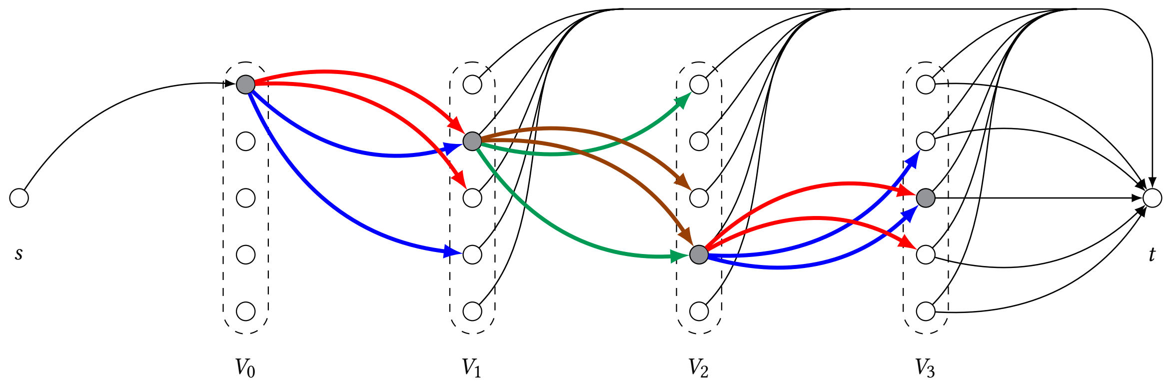


Figure 2: Illustration of the graph in the reduction for minimum s-t cut from HPC_3 with $n = 5$. The black (thin) edges form input-independent gadgets while blue, red , brown, and green (thick) edges depend on the inputs of P_A, P_B, P_C, and P_D, respectively. Marked nodes denote the vertices corresponding to pointers $z_0, \ldots, z_3$. The input-dependent edges incident on "non-pointer" vertices are omitted. This construction has parallel edges but they can be removed; see the full version [10].

in the first j phases of the protocol, distribution of z_j is still "close" to being uniform. This in particular implies that at the end of the protocol, i.e., at the end of phase k, the target pointer z_k is essentially distributed as in its original distribution (which is uniform over $\mathcal{Y}$ or $\mathcal{X}$ depending on whether k is odd or even). Hence π_{HPC} should not be able to find z_k at the end of phase k. The proof of Theorem 1 follows easily from this intuition.

Proof Sketch of Lemma 3.1. The first step of proof is to show that finding the target element of a *uniformly at random* chosen instance of Set-Int (as opposed to an instance corresponding to any particular pointer) in HPC is not possible with low communication. For any $x \in \mathcal{X}$ and any $y \in \mathcal{Y}$, define the random variables $\mathsf{T}_x \in \mathcal{Y}$ and $\mathsf{T}_y \in \mathcal{X}$, which correspond to the target elements of Set-Int on (A_x, B_x) and (C_y, D_y), respectively. The following lemma formalizes the above statement. For simplicity, we only state it for T_x for $x \sim \mathcal{U}_\mathcal{X}$; an identical bound also hold for T_y for $y \sim \mathcal{U}_\mathcal{Y}$.

Lemma 3.2 (Informal). *For $j \in [k]$:*

$$\mathop{\mathbb{E}}_{(E_j,\Pi_j)} \mathop{\mathbb{E}}_{x\sim\mathcal{U}_\mathcal{X}} \left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{T}_x \mid E_j, \Pi_j), \mathrm{dist}(\mathsf{T}_x))\right] = o(1).$$

Let us first see why Lemma 3.2 implies Lemma 3.1. The proof is by induction. Consider some phase $j \in [k]$ and suppose j is odd by symmetry. The goal is to prove that distribution of Z_j conditioned on $(E_j, \Pi_j) = (z_1, \ldots, z_{j-1}, \Pi_1, \ldots, \Pi_{j-1}, \Pi_j)$ is close to original distribution of Z_j (on average over choices of (E_j, Π_j)). Notice that since we assumed j is odd, Z_j is a function of the inputs to P_A and P_B. On the other hand, in an odd phase, only the players P_C and P_D communicate and hence Π_j is a function of the inputs to these players. Conditioning on E_j and using the rectangle property of deterministic protocols, together with the fact that inputs to P_A, P_B are independent of inputs to P_C, P_D, implies that $\mathsf{Z}_j \perp \Pi_j \mid E_j$. We now have:

(i) Conditioned on z_{j-1}, Z_j is the target element of the instance $(A_{z_{j-1}}, B_{z_{j-1}})$, i.e., $\mathsf{Z}_j = \mathsf{T}_{z_{j-1}}$.
(ii) z_{j-1} itself is distributed according to $\mathrm{dist}(\mathsf{Z}_{j-1} \mid E_{j-1}, \Pi_{j-1})$ (because we removed the conditioning on Π_j by the above argument).
(iii) $\mathrm{dist}(\mathsf{Z}_{j-1} \mid E_{j-1}, \Pi_{j-1})$ is close to the uniform distribution by induction.

As such we can now simply apply Lemma 3.2 (by replacing x with z_{j-1} since they essentially have the same distribution) and obtain that distribution of $\mathsf{Z}_j = \mathsf{T}_{z_{j-1}}$ with and without conditioning on (E_j, Π_j) is almost the same (averaged over choices of (E_j, Π_j)), proving the lemma.

Proof Sketch of Lemma 3.2. The proof of this lemma is based on a direct-sum style argument combined with a new result that we prove for Set-Int. The direct-sum argument implies that since x is chosen uniformly at random from n elements in $\mathcal{X}$, and protocol π_{HPC} is communicating $o(n^2)$ bits in total, then it can only reveal $o(n)$ bits of information about the instance (A_x, B_x). This part follows the standard direct-sum arguments for information complexity (see, e.g. [26, 36]) but we also need to take into account that if x is one of the pointers we conditioned on in E_j, then π_{HPC} may reveal more information about (A_x, B_x); fortunately, this event happens with negligible probability for $k \ll n$ and so the argument continues to hold.

By above argument, proving Lemma 3.2 reduces to showing that if a protocol reveals $o(n)$ bits of information about an instance of Set-Int, then the distribution of the target element varies from the uniform distribution in total variation distance by only $o(1)$. This is the main part of the proof of Lemma 3.2 and is precisely the content of our next technical result in the following section.

3.4 A New Communication Lower Bound for Set Intersection

We say that a protocol π_{SI} *ε-solves* Set-Int on the distribution $\mathcal{D}_{\mathsf{SI}}$ iff it can alter the distribution of the target element from its original distribution by at least ε in total variation distance, i.e.,

$$\mathop{\mathbb{E}}_{\Pi_{\mathsf{SI}}\sim\Pi_{\mathsf{SI}}} \left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{T} \mid \Pi_{\mathsf{SI}}), \mathrm{dist}(\mathsf{T}))\right] \geq \varepsilon.$$

Here Π_{SI} and T are the random variables for the transcript of the protocol (including public randomness) and the target element, respectively.

To finish the proof of Lemma 3.2, we need to prove that a protocol that $\Omega(1)$-solves Set-Int has $\Omega(n)$ communication cost (even information cost). Note that ε-solving is an algorithmically simpler task than finding the target element. For example, a protocol may change the distribution of T to having $(1+\varepsilon)/n$ probability on $n/2$ elements and $(1-\varepsilon)/n$ probability on the remaining $n/2$. This ε-solves Set-Int yet the target element can only be found with probability $(1+\varepsilon)/n$ in this distribution. On the other hand, any protocol that finds the target element with probability $p \in (0,1)$ also p-solves Set-Int. Because of this, the lower bounds mentioned in Section 2 for set intersection do not suffice for our purpose. Instead, we prove the following theorem in this paper.

Theorem 2. *Any protocol π_{SI} that ε-solves* Set-Int *on distribution $\mathcal{D}_{\mathsf{SI}}$ has internal information cost $\mathsf{IC}_{\mathcal{D}_{\mathsf{SI}}}(\pi_{\mathsf{SI}}) = \Omega(\varepsilon^2 \cdot n)$.*

As information cost lower bounds communication cost, Theorem 2 also proves a communication lower bound for Set-Int (although we need the stronger result for information cost in our proofs). By our discussion earlier, Theorem 2 can be used to finalize the proof of Lemma 3.2 (and hence Theorem 1). We now give an overview of the proof of Theorem 2.

For an instance (A, B) of Set-Int, with a slight abuse of notation, we write $A := (a_1, \ldots, a_n)$ and $B := (b_1, \ldots, b_n)$ for $a_i, b_i \in \{0,1\}$ as characteristic vector of the sets given to Alice and Bob. Under this notation, the target element corresponds to the unique index $t \in [n]$ such that $(a_t, b_t) = (1,1)$. The proof of Theorem 2 is based on reducing Set-Int to a special case of this problem on only 2 coordinates, which we define as the Pair-Int problem. In Pair-Int, Alice and Bob are given (x_1, x_2) and (y_1, y_2) in $\{0,1\}^2$ and their goal is to find the unique index $k \in \{1,2\}$ such that $(x_k, y_k) = (1,1)$. We use $\mathcal{D}_{\mathsf{PI}}$ to denote the hard distribution for this problem which is equivalent to $\mathcal{D}_{\mathsf{SI}}$ for $n = 2$.

Given a protocol π_{SI} for ε-solving Set-Int on $\mathcal{D}_{\mathsf{SI}}$, we design a protocol π_{PI} for finding the index k in instances of Pair-Int sampled from $\mathcal{D}_{\mathsf{PI}}$ with probability $1/2 + \Omega(\varepsilon)$. The reduction is as follows.

Reduction: Alice and Bob publicly sample $i, j \in [n]$ uniformly at random without replacement. Then, Alice sets $a_i = x_1$ and $a_j = x_2$ and Bob sets $b_i = y_1$ and $b_j = y_2$, using their given inputs in Pair-Int. The players sample the remaining coordinates of (A, B) in $[n] \setminus \{i, j\}$ using a combination of public and private randomness that we explain later in the proof sketch of Lemma 3.4. This sampling ensures that the resulting instance (A, B) of Set-Int is sampled from $\mathcal{D}_{\mathsf{SI}}$ such that its target element is i when $k = 1$ and is j when $k = 2$. After this, the players run the protocol π_{SI} on (A, B) and let Π_{SI} be the transcript of this protocol. Using this, Bob computes the distribution $\mathrm{dist}(\mathsf{T} \mid \Pi_{\mathsf{SI}}) = (p_1, \ldots, p_n)$ which assigns probabilities to elements in $[n]$ as being the target element. Finally, Bob checks the value of p_i and p_j and return $k = 1$ if $p_i > p_j$ and $k = 2$ otherwise (breaking the ties consistently when $p_i = p_j$). The remainder of the proof consists of three main steps:

(*i*) Proving the correctness of protocol π_{PI}:

Lemma 3.3 (Informal). *Protocol π_{PI} outputs the correct answer with probability $\frac{1}{2} + \Omega(\varepsilon)$.*

(*ii*) Proving an upper bound on "information cost" of π_{PI} (the reason for quotations is that strictly speaking this quantity is not the information cost of π_{PI} but rather a lower bound for it).

Lemma 3.4 (Informal). *Let Π_{PI} denote the random variable for the transcript of the protocol π_{PI} and K be the random variable for the index k in distribution $\mathcal{D}_{\mathsf{PI}}$. We have,*

$$\mathbb{I}_{\mathcal{D}_{\mathsf{PI}}}(\mathsf{X}_1, \mathsf{X}_2 ; \Pi_{\mathsf{PI}} \mid \mathsf{Y}_1, \mathsf{Y}_2, \mathsf{K}) + \mathbb{I}_{\mathcal{D}_{\mathsf{PI}}}(\mathsf{Y}_1, \mathsf{Y}_2 ; \Pi_{\mathsf{PI}} \mid \mathsf{X}_1, \mathsf{X}_2, \mathsf{K}) \leq \frac{1}{n-1} \cdot \mathsf{IC}_{\mathcal{D}_{\mathsf{SI}}}(\pi_{\mathsf{SI}}).$$

(*iii*) Proving a lower bound on "information cost" (as used in Part (*ii*)) of protocols for Pair-Int:

Lemma 3.5 (Informal). *If π_{PI} outputs the correct answer on $\mathcal{D}_{\mathsf{PI}}$ with probability at least $\frac{1}{2} + \Omega(\varepsilon)$, then,*

$$\mathbb{I}_{\mathcal{D}_{\mathsf{PI}}}(\mathsf{X}_1, \mathsf{X}_2 ; \Pi_{\mathsf{PI}} \mid \mathsf{Y}_1, \mathsf{Y}_2, \mathsf{K}) + \mathbb{I}_{\mathcal{D}_{\mathsf{PI}}}(\mathsf{Y}_1, \mathsf{Y}_2 ; \Pi_{\mathsf{PI}} \mid \mathsf{X}_1, \mathsf{X}_2, \mathsf{K}) = \Omega(\varepsilon^2).$$

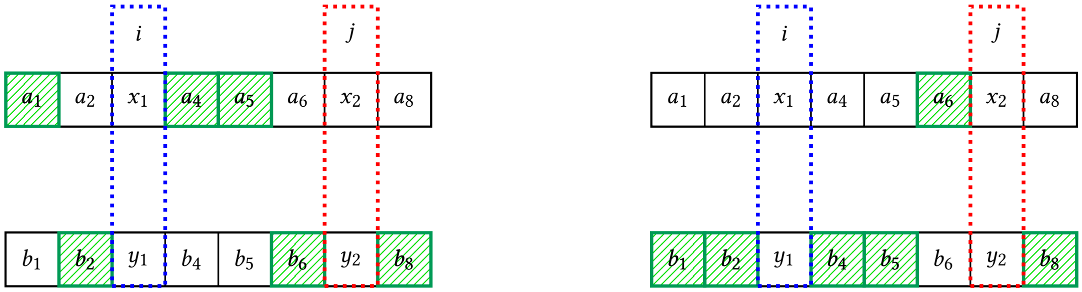
By Lemma 3.4, $\mathsf{IC}_{\mathcal{D}_{\mathsf{SI}}}(\pi_{\mathsf{SI}})$ is $\Omega(n)$ times larger than LHS of Lemma 3.5, and this, combined with Lemma 3.3, implies that information cost of π_{SI} needs to be $\Omega(\varepsilon^2) \cdot \Omega(n)$, proving Theorem 2.

Proof Sketch of Lemma 3.3. Let us again consider a protocol π_{SI} such that $\mathrm{dist}(\mathsf{T} \mid \Pi_{\mathsf{SI}})$ is putting $(1+\varepsilon)/n$ mass over $n/2$ elements and $(1-\varepsilon)/n$ mass on the remaining ones. Suppose that the correct answer to the instance of Pair-Int is index 1. We know that in this case, the index i chosen by π_{PI} will be the target index t in the instance (A, B). A key observation here is that the index j however can be any of the coordinates in instance (A, B) other than the target element with the same probability. As such, parameters p_i and p_j used to decide the answer in π_{PI} are distributed as follows: p_i is sampled from $\mathrm{dist}(\mathsf{T} \mid \Pi_{\mathsf{SI}})$ and hence has value $(1+\varepsilon)/n$ with probability $(1+\varepsilon)/2$ and $(1-\varepsilon)/n$ with probability $(1-\varepsilon)/2$. On the other hand, p_j is chosen uniformly at random from $(p_1, \ldots, p_n)$ and hence is $(1+\varepsilon)/n$ or $(1-\varepsilon)/n$ with the same probability of half. Thus $p_i > p_j$ with probability $1/2 + \Omega(\varepsilon)$ and hence π_{PI} has $\Omega(\varepsilon)$ advantage over random guessing.

The proof of Lemma 3.3 then formalizes the observations above and extend this argument to any protocol π_{SI} that ε-solves Set-Int no matter how it alters the distribution of the target element.

Proof Sketch of Lemma 3.4. We first note that the LHS in Lemma 3.4 is *not* the internal information cost of π_{PI} due to further conditioning on K (this can only be smaller than $\mathsf{IC}_{\mathcal{D}_{\mathsf{PI}}}(\pi_{\mathsf{PI}})$). Hence, Lemma 3.4 is proving a "weaker" statement than a direct-sum result for information cost of π_{PI}. The reason for settling for this weaker statement has to do with the fact that the coordinates in distribution $\mathcal{D}_{\mathsf{SI}}$ are *not* chosen independently.

The intuition behind the proof is as follows. The LHS in Lemma 3.5 is the information revealed about the input of players (in Pair-Int) averaged over choices of $k = 1$ and $k = 2$. Let us assume $k = 1$ by symmetry. In this case, this quantity is simply the information revealed about (x_2, y_2) by the protocol as $(x_1, y_1) = (1,1)$ and hence has no entropy. However, when $k = 1$, (x_2, y_2) is embedded in index j, i.e., $(x_2, y_2) = (a_j, b_j)$ and has the same distribution as all other coordinates in A_{-i}, B_{-i}. As such, since the protocol π_{SI} called inside π_{PI} is oblivious to the choice of j, the information revealed about (a_j, b_j) in average is smaller than the information revealed by π_{SI}



(a) An example with $\ell = 3$ and $S = \{1, 4, 5\}$: $\{a_1, a_4, a_5, b_2, b_6, b_8\}$ is sampled publicly. $\{a_2, a_6, a_8\}$ and $\{b_1, b_4, b_5\}$ are sampled privately.

(b) An example with $\ell = 1$ and $S = \{6\}$: $\{a_6, b_1, b_2, b_4, b_5, b_8\}$ is sampled publicly. $\{a_1, a_2, a_4, a_5, a_8\}$ and $\{b_6\}$ are sampled privately.

Figure 3: Illustration of the process of sampling of instances of Set-Int **in** π_{PI} **for** $n = 8$**. In these examples,** $i = 3$ **and** $j = 7$ **and hence** $(a_3, a_7) = (x_1, x_2)$ **and** $(b_3, b_7) = (y_1, y_2)$**. of** ℓ **and** S**.**

about A_{-i}, B_{-i} (which itself is at most the information cost of π_{SI}) by a factor of $n - 1$.

This outline oversimplifies many details. One such detail is the way of ensuring a "symmetric treatment" of both indices i and j. This is crucial for the above argument to work for both $k = 1$ and $k = 2$ cases simultaneously, without the players knowing which index the "averaging" of information is being done for (index j in the context of the discussion above). The key step in making this information-theoretic argument work is the following public-private sampling: Alice and Bob use public randomness to pick an integer $\ell \in [n-2]$ uniformly at random and then pick a set S of size ℓ uniformly at random from $[n] \setminus \{i, j\}$. Next, the players sample $a_{i'}$ and $b_{j'}$ for $i' \in S$ and $j' \in ([n] \setminus \{i, j\}) \setminus S$ from $\mathcal{D}_{\mathsf{SI}}$ again using public randomness. Finally, each player samples the remaining coordinates in the input using private randomness from $\mathcal{D}_{\mathsf{SI}}$. Figure 3 gives an example.

Proof Sketch of Lemma 3.5. Let $\Pi_{[x_1x_2,\, y_1y_2]}$ denote the transcript of the protocol condition on the inputs (x_1, x_2) and (y_1, y_2) to Alice and Bob. Suppose towards a contradiction that the LHS of Lemma 3.5 is $o(\varepsilon^2)$. By focusing on the conditional terms when $k = 1$, we can show that distribution of $\Pi_{[1x_2',\, 1y_2']}$ and $\Pi_{[1x_2'',\, 1y_2'']}$ for all choices of (x_2', y_2') and (x_2'', y_2'') in the support of $\mathcal{D}_{\mathsf{PI}}$ are quite close. This is intuitively because the information revealed about (x_2, y_2) by π_{PI} conditioned on $k = 1$ is small (the same result holds for $\Pi_{[x_2'1,\, y_2'1]}$ and $\Pi_{[x_2''1,\, y_2''1]}$ by $k = 2$ terms).

Up until this point, there is no contradiction as the answer to inputs $(1, *)$,$(1, *)$ to Alice and Bob is always 1 and hence there is no problem with the corresponding transcripts in $\Pi_{[1*,\, 1*]}$ to be similar (similarly for $\Pi_{[*1,\, *1]}$ separately). However, we combine this with the cut-and-paste property of randomized protocols based on Hellinger distance to argue that in fact the distribution of $\Pi_{[10,\, 10]}$ and $\Pi_{[01,\, 01]}$ are also similar. This then implies that $\Pi_{[1*,\, 1*]}$ has almost the same distribution as $\Pi_{[*1,\, *1]}$, and now this is a contradiction as the answer to the protocol (a function of the transcript) needs to be different between these two types of inputs.

This concludes the high-level overview of our proofs (for more details, see the full version of the paper [10]).

4 COMMUNICATION COMPLEXITY OF HIDDEN-POINTER CHASING

We give the proof of Theorem 1 in this section. We start with defining our hard distribution of instances for HPC_k and then use this distribution to prove the lower bound.

A Hard Distribution for HPC. The hard distribution for HPC is simply the product of distribution $\mathcal{D}_{\mathsf{SI}}$ for every $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

Distribution $\mathcal{D}_{\mathsf{HPC}}$ on tuples $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ from the universes $\mathcal{X}$ and $\mathcal{Y}$:

1. For any $x \in \mathcal{X}$, sample $(A_x, B_x) \sim \mathcal{D}_{\mathsf{SI}}$ from the universe $\mathcal{Y}$ *independently.*
2. For any $y \in \mathcal{Y}$, sample $(C_y, D_y) \sim \mathcal{D}_{\mathsf{SI}}$ from the universe $\mathcal{X}$ *independently.*

The following simple observation is in order.

Observation 4.1. *Distribution $\mathcal{D}_{\mathsf{HPC}}$ is not a product distribution. However, in this distribution:*

(i) The inputs to P_A and P_B are independent of the inputs to P_C and P_D, i.e., $(\boldsymbol{A}, \boldsymbol{B}) \perp (\boldsymbol{C}, \boldsymbol{D})$.
(ii) For any $x \in \mathcal{X}$, (A_x, B_x) is independent of all other $(A_{x'}, B_{x'})$ for $x' \neq x \in \mathcal{X}$. Similarly for all $y, y' \in \mathcal{Y}$ and (C_y, D_y) and $(C_{y'}, D_{y'})$.

Based on this observation, we also have the following simple property (proof is a simple application of rectangle property of protocols and is deferred to the full version [10]).

Proposition 4.2. *Let π_{HPC} be any deterministic protocol for HPC_k on $\mathcal{D}_{\mathsf{HPC}}$. Then, for any transcript Π of π_{HPC}, $(\boldsymbol{A}, \boldsymbol{B}) \perp (\boldsymbol{C}, \boldsymbol{D}) \mid \boldsymbol{\Pi} = \Pi$.*

4.1 Proof of Theorem 1: A Communication Lower Bound for HPC_k

We prove the lower bound for any arbitrary deterministic protocol π_{HPC} and then apply Yao's minimax principle [111] to extend it to randomized protocols as well. We first setup some notation.

Notation. Fix any k-phase *deterministic* protocol π_{HPC} for HPC_k throughout the proof. We use $j = 1$ to k to index the phases of this protocol, as well as the pointers $z_1, \ldots, z_k$. For any $j \in [k]$, we define Π_j as the set of all messages communicated by π_{HPC} in phase j and $\Pi := (\Pi_1, \ldots, \Pi_k)$ as the transcript of the protocol π_{HPC}.

For any $x \in \mathcal{X}$ and any $y \in \mathcal{Y}$, we define the random variables $\mathsf{T}_x \in \mathcal{Y}$ and $\mathsf{T}_y \in \mathcal{X}$, which correspond to the target elements of the Set-Int problem on (A_x, B_x) and (C_y, D_y), respectively.

We further define $\mathsf{E}_j := (\Pi^{<j}, \mathsf{Z}^{<j})$ for any $j > 1$ and $\mathsf{E}_1 = z_0$, i.e., the first pointer. We can think of E_j as the information "easily known" to all players at the beginning of phase j.

The main step of the proof of Theorem 1 is the following key lemma which we prove inductively.

Lemma 4.3. *Let $\mathsf{CC}(\pi_{\mathsf{HPC}}) := \mathsf{CC}_{\mathcal{D}_{\mathsf{HPC}}}(\pi_{\mathsf{HPC}})$. There exists an absolute constant $c > 0$ such that for all $j \in [k]$:*

$$\mathop{\mathbb{E}}_{(E_j,\Pi_j)}\Big[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{Z}_j \mid E_j, \Pi_j), \mathrm{dist}(\mathsf{Z}_j))\Big] \leq j \cdot c \cdot \Big(\frac{\sqrt{\mathsf{CC}(\pi_{\mathsf{HPC}}) + k \cdot \log n} + k}{n}\Big).$$

We first use Lemma 4.3 to prove Theorem 1 and then present a proof of Lemma 4.3.

Proof of Theorem 1 (assuming Lemma 4.3). The $\Omega(n)$ term in the lower bound trivially follows from the $\Omega(n)$ lower bound for set intersection (e.g. Theorem 2 with constant ε). In the following we prove the first (and the main) term. Note that for this purpose, we can assume $k = o(\sqrt{n})$ as otherwise the dominant term would already be the second term.

Let π_{HPC} be any deterministic protocol for HPC_k for $k = o(\sqrt{n})$ with communication cost $\mathsf{CC}_{\mathcal{D}_{\mathsf{HPC}}}(\pi_{\mathsf{HPC}}) = o(n^2/k^2)$. Recall that $\mathrm{dist}(\mathsf{Z}_k) = \mathcal{U}_{\mathcal{X}}$ if k is even and $\mathrm{dist}(\mathsf{Z}_k) = \mathcal{U}_{\mathcal{Y}}$ if k is odd. Let us assume by symmetry that k is even. By Lemma 4.3, we have,

$$\begin{aligned}
&\mathop{\mathbb{E}}_{(E_k,\Pi_k)}\Big[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{Z}_k \mid E_k, \Pi_k), \mathcal{U}_{\mathcal{X}})\Big] \\
&\leq k \cdot c \cdot \Big(\frac{\sqrt{\mathsf{CC}(\pi_{\mathsf{HPC}}) + k \cdot \log n} + k}{n}\Big) \\
&= k \cdot c \cdot \Big(o(\frac{1}{k}) + o(\frac{\sqrt{\log n}}{n^{3/4}}) + o(\frac{k}{n})\Big) \\
&= o(\frac{k}{k}) + o(\frac{k \cdot \sqrt{\log n}}{n^{3/4}}) + o(\frac{k^2}{n}) = o(1), \qquad (1)
\end{aligned}$$

as c is an absolute constant.

On the other hand, (E_k, Π_k) contains the whole transcript Π of the protocol and hence the output of the protocol π_{HPC} is fixed conditioned on (E_k, Π_k). We use $O(E_k, \Pi_k)$ to denote this output. We have,

$$\begin{aligned}
&\Pr_{(E_k,\Pi_k)}(\pi_{\mathsf{HPC}} \text{ is correct}) \\
&= \mathop{\mathbb{E}}_{(E_k,\Pi_k)} \Pr_{\mathsf{Z}_k \mid (E_k,\Pi_k)}(\mathsf{Z}_k = O(E_k, \Pi_k)) \\
&\leq \mathop{\mathbb{E}}_{(E_k,\Pi_k)}\Big[\Pr_{\mathsf{Z}_k \sim \mathcal{U}_{\mathcal{X}}}(\mathsf{Z}_k = O(E_k, \Pi_k)) + \Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{Z}_k \mid E_k, \Pi_k), \mathcal{U}_{\mathcal{X}})\Big] \\
&\leq \frac{1}{n} + \mathop{\mathbb{E}}_{(E_k,\Pi_k)}\Big[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{Z}_k \mid E_k, \Pi_k), \mathcal{U}_{\mathcal{X}})\Big] \underset{\text{Eq (1)}}{\leq} \frac{1}{n} + o(1).
\end{aligned}$$

Hence, π_{HPC} cannot output the correct solution with at least a constant probability of success, proving the lower bound for deterministic algorithms.

To finalize, we can extend this (distributional) lower bound to randomized protocols by the easy direction of Yao's minimax principle [111], namely an averaging argument that picks the "best" randomness of the protocol. This concludes the proof. □

4.2 Proof of Lemma 4.3

The proof of Lemma 4.3 consists of two main steps. We first show that finding the target element of a *uniformly at random* chosen instance of Set-Int (as opposed to the instance corresponding to any particular pointer) in HPC is not possible unless we make a large communication. Then, we prove inductively that in each phase j, the distribution of the pointer z_j is close to uniform and hence by the argument in the first step, we should not be able to find the target element t_{z_j} associated with z_j and use this to finalize the proof. The following lemma captures the first part (we only write this for $x \sim \mathcal{U}_{\mathcal{X}}$; an analogous statement also holds for $y \sim \mathcal{U}_{\mathcal{Y}}$).

Lemma 4.4. *There exists an absolute constant $c > 0$ such that for any $j \in [k]$,*

$$\mathop{\mathbb{E}}_{(E_j,\Pi_j)}\mathop{\mathbb{E}}_{x \sim \mathcal{U}_{\mathcal{X}}}\Big[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{T}_x \mid E_j, \Pi_j), \mathrm{dist}(\mathsf{T}_x))\Big] \leq c \cdot \Big(\frac{\sqrt{\mathsf{CC}(\pi_{\mathsf{HPC}}) + j \cdot \log n} + j}{n}\Big).$$

The proof of this lemma is based on a direct-sum style argument combined with Theorem 2. For intuition, consider a protocol that uses $o(n^2)$ communication in its first j phases and assume by way of contradiction that it can reduce the LHS of one of the equations in Lemma 4.4 by $\Omega(1)$. Using a direct-sum style argument, we can then argue that the transcript of the first j phases of this protocol only reveal $o(n)$ bits of information about a uniformly at random chosen instance (A_x, B_x) of Set-Int but is enough to $\Omega(1)$-solve the instance (A_x, B_x), which is in contradiction with our bounds in Theorem 2. Note that in this discussion, for the sake of simplicity, we neglected the role of extra conditioning on $Z^{<j}$ in E_j in the LHS of equations; handling this extra conditioning results in the extra additive factor in RHS. Proof of Lemma 4.4 is quite technical and is postponed to the full version of the paper [10].

Before getting to the proof of Lemma 4.3, we also need the following simple claim based on the rectangle property of the protocol π_{HPC} (proof appears in full version [10]).

Claim 4.5. *For any $j \in [k]$ and choice of (E_j, Π_j), $\mathrm{dist}(\mathsf{Z}_j \mid E_j, \Pi_j) = \mathrm{dist}(\mathsf{Z}_j \mid E_j)$.*

We are now finally ready to prove Lemma 4.3.

Proof of Lemma 4.3. Let c be the constant in Lemma 4.4. We prove Lemma 4.3 by induction. We start with the proof of the base case for $j = 1$ and then prove the inductive step.

Base case. Recall that we defined $E_1 = z_0$ which is deterministically fixed. This, together with Claim 4.5, implies that $\mathrm{dist}(\mathsf{Z}_1 \mid E_1, \Pi_1) = \mathrm{dist}(\mathsf{Z}_1)$, which finalizes proof of the base case.

Induction step. Let us now prove the lemma inductively for $j > 1$.

$$\mathop{\mathbb{E}}_{(E_j,\Pi_j)}\left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{Z}_j \mid E_j,\Pi_j),\mathrm{dist}(\mathsf{Z}_j))\right]$$
$$\underset{\text{Claim 4.5}}{=} \mathop{\mathbb{E}}_{E_j}\left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{Z}_j \mid E_j),\mathrm{dist}(\mathsf{Z}_j))\right]$$
$$= \mathop{\mathbb{E}}_{(Z^{<j},\Pi^{<j})}\left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{Z}_j \mid Z^{<j},\Pi^{<j}),\mathrm{dist}(\mathsf{Z}_j))\right]$$
(by definition of $E_j := (Z^{<j},\Pi^{<j})$)
$$= \mathop{\mathbb{E}}_{(Z^{<j},\Pi^{<j})}\left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{T}_{z_{j-1}} \mid Z^{<j-1},z_{j-1},\Pi^{<j}),\mathrm{dist}(\mathsf{Z}_j))\right].$$
(by definition, the pointer $\mathsf{Z}_j = \mathsf{T}_{z_{j-1}}$)

We can write the RHS above as:

$$\mathop{\mathbb{E}}_{(E_j,\Pi_j)}\left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{Z}_j \mid E_j,\Pi_j),\mathrm{dist}(\mathsf{Z}_j))\right] = \mathop{\mathbb{E}}_{(Z^{<j-1},\Pi^{<j})}\ \mathop{\mathbb{E}}_{z_{j-1}\sim \mathsf{Z}_{j-1}\mid(Z^{<j-1},\Pi^{<j})}\left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{T}_{z_{j-1}} \mid Z^{<j-1},\Pi^{<j}),\mathrm{dist}(\mathsf{Z}_j))\right].$$

This is because $\mathsf{T}_{z_{j-1}} \perp (\mathsf{Z}_{j-1} = z_{j-1}) \mid Z^{<j-1},\Pi^{<j}$: if $j-1$ is odd, $\mathsf{T}_{z_{j-1}}$ is a function of $(\boldsymbol{C},\boldsymbol{D})$ and if $j-1$ is even, $\mathsf{T}_{z_{j-1}}$ is a function of $(\boldsymbol{A},\boldsymbol{B})$. On the other hand, if $j-1$ is odd, then Z_{j-1} is a function of $(\boldsymbol{A},\boldsymbol{B})$ and if even, then Z_{j-1} is a function of $(\boldsymbol{C},\boldsymbol{D})$. Finally, by Proposition 4.2, $(\boldsymbol{A},\boldsymbol{B}) \perp (\boldsymbol{B},\boldsymbol{D}) \mid \Pi^{<j}$, proving the conditional independence.

Now notice that distribution of z_{j-1} in the expectation-term above is $\mathrm{dist}(\mathsf{Z}_{j-1} \mid E_{j-1},\Pi_{j-1})$. By symmetry, let us assume $j-1$ is odd and hence $z_{j-1} \in \mathcal{Y}$. Since total variation distance is bounded by 1 always, we can upper bound RHS above with:

$$\mathop{\mathbb{E}}_{(E_j,\Pi_j)}\left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{Z}_j \mid E_j,M_j),\mathrm{dist}(\mathsf{Z}_j))\right]$$
$$\leq \mathop{\mathbb{E}}_{(Z^{<j-1},\Pi^{<j})}\left[\mathop{\mathbb{E}}_{(z_{j-1}\sim\mathcal{U}_{\mathcal{Y}})}\left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{T}_{z_{j-1}} \mid Z^{<j-1},\Pi^{<j}),\mathrm{dist}(\mathsf{Z}_j))\right]\right] + \mathop{\mathbb{E}}_{(Z^{<j-1},\Pi^{<j})}\left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{Z}_{j-1} \mid E_{j-1},\Pi_{j-1}),\mathcal{U}_{\mathcal{Y}})\right]$$
$$= \mathop{\mathbb{E}}_{(E_{j-1},\Pi_{j-1})}\ \mathop{\mathbb{E}}_{y\sim\mathcal{U}_{\mathcal{Y}}}\left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{T}_y \mid E_{j-1},\Pi_{j-1}),\mathrm{dist}(\mathsf{Z}_j))\right] + \mathop{\mathbb{E}}_{(E_{j-1},\Pi_{j-1})}\left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{Z}_{j-1} \mid E_{j-1},\Pi_{j-1}),\mathrm{dist}(\mathsf{Z}_{j-1}))\right],$$

where in the first term above we only changed the name of variable z_{j-1} to y and in the second term we used $\mathrm{dist}(\mathsf{Z}_{j-1}) = \mathcal{U}_{\mathcal{Y}}$. By Lemma 4.4, we can bound the first term and by induction, we can bound the second one. Hence,

$$\mathop{\mathbb{E}}_{(E_j,\Pi_j)}\left[\Delta_{\mathsf{TV}}(\mathrm{dist}(\mathsf{Z}_j \mid E_j,\Pi_j),\mathrm{dist}(\mathsf{Z}_j))\right]$$
$$\leq c\cdot\left(\frac{\sqrt{\mathrm{CC}(\pi_{\mathsf{HPC}})+j\cdot\log n}+j}{n}\right) + (j-1)\cdot c\cdot\left(\frac{\sqrt{\mathrm{CC}(\pi_{\mathsf{HPC}})+k\cdot\log n}+k}{n}\right)$$
$$\leq j\cdot c\cdot\left(\frac{\sqrt{\mathrm{CC}(\pi_{\mathsf{HPC}})+k\cdot\log n}+k}{n}\right).$$
(where we replaced $j \leq k$ by k in the first term)

This concludes the proof. □

REFERENCES

[1] Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Ami Paz. 2019. Smaller Cuts, Higher Lower Bounds. *CoRR* abs/1901.01630 (2019).

[2] Farid M. Ablayev. 1993. Lower Bounds for One-way Probabilistic Communication Complexity. In *Automata, Languages and Programming, 20nd International Colloquium, ICALP93, Lund, Sweden, July 5-9, 1993, Proceedings.* 241–252.

[3] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. 2012. Analyzing Graph Structure via Linear Measurements. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '12).* SIAM, 459–467. http://dl.acm.org/citation.cfm?id=2095116.2095156

[4] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. 2012. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012.* 5–14. https://doi.org/10.1145/2213556.2213560

[5] Noga Alon, László Babai, and Alon Itai. 1986. A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. *J. Algorithms* 7, 4 (1986), 567–583.

[6] Noga Alon, Yossi Matias, and Mario Szegedy. 1996. The space complexity of approximating the frequency moments. In *STOC*. ACM, 20–29.

[7] Noga Alon, Noam Nisan, Ran Raz, and Omri Weinstein. 2015. Welfare Maximization with Limited Interaction. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015.* 1499–1512.

[8] Sepehr Assadi. 2017. Combinatorial Auctions Do Need Modest Interaction. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017.* 145–162.

[9] Sepehr Assadi. 2017. Tight Space-Approximation Tradeoff for the Multi-Pass Streaming Set Cover Problem. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017.* 321–335.

[10] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. 2019. Polynomial Pass Lower Bounds for Graph Streaming Algorithms. *CoRR* abs/1904.04720 (2019).

[11] Sepehr Assadi, Yu Chen, and Sanjeev Khanna. 2019. Sublinear Algorithms for (Δ + 1) Vertex Coloring. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019.* 767–786.

[12] Sepehr Assadi and Sanjeev Khanna. 2018. Tight Bounds on the Round Complexity of the Distributed Maximum Coverage Problem. In *Proceedings of the Twenty-Nine Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018.*

[13] Sepehr Assadi, Sanjeev Khanna, and Yang Li. 2016. Tight bounds for single-pass streaming complexity of the set cover problem. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016.* 698–711.

[14] Sepehr Assadi, Sanjeev Khanna, and Yang Li. 2017. On Estimating Maximum Matching Size in Graph Streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19.* 1723–1742.

[15] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. 2016. Maximum Matchings in Dynamic Graph Streams and the Simultaneous Communication Model. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016.* 1345–1364.

[16] László Babai, Peter Frankl, and Janos Simon. 1986. Complexity classes in communication complexity theory (preliminary version). In *27th Annual Symposium on Foundations of Computer Science, 27-29 October 1986.* 337–347.

[17] Eric Balkanski, Adam Breuer, and Yaron Singer. 2018. Non-monotone Submodular Maximization in Exponentially Fewer Iterations. *CoRR* abs/1807.11462. To appear in NIPS 2018. (2018).

[18] Eric Balkanski, Aviad Rubinstein, and Yaron Singer. 2016. The Power of Optimization from Samples. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain.* 4017–4025.

[19] Eric Balkanski, Aviad Rubinstein, and Yaron Singer. 2017. The limitations of optimization from samples. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017.* 1016–1027.

[20] Eric Balkanski, Aviad Rubinstein, and Yaron Singer. 2018. An Exponential Speedup in Parallel Running Time for Submodular Maximization without Loss in Approximation. *CoRR* abs/1804.06355. To appear in SODA 2019. (2018).

[21] Eric Balkanski and Yaron Singer. 2017. Minimizing a Submodular Function from Samples. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA.* 814–822.

[22] Eric Balkanski and Yaron Singer. 2018. The adaptive complexity of maximizing a submodular function. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018.* 1138–1151.

[23] Eric Balkanski and Yaron Singer. 2018. Parallelization does not Accelerate Convex Optimization: Adaptivity Lower Bounds for Non-smooth Convex Minimization. *CoRR* abs/1808.03880 (2018).

[24] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. 2002. An Information Statistics Approach to Data Stream and Communication Complexity. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Proceedings.* 209–218.

[25] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. 2002. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA.* 623–632.

[26] Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. 2010. How to compress interactive communication. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, 5-8 June 2010.* 67–76.

[27] MohammadHossein Bateni, Hossein Esfandiari, and Vahab S. Mirrokni. 2017. Almost Optimal Streaming Algorithms for Coverage Problems. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017.* 13–23.

[28] Ruben Becker, Andreas Karrenbauer, Sebastian Krinninger, and Christoph Lenzen. 2017. Near-Optimal Approximate Shortest Paths and Transshipment in Distributed and Streaming Models. In *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria.* 7:1–7:16.

[29] Suman K. Bera and Amit Chakrabarti. 2017. Towards Tighter Space Bounds for Counting Triangles and Other Substructures in Graph Streams. In *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany.* 11:1–11:14.

[30] Guy E. Blelloch, Jeremy T. Fineman, and Julian Shun. 2012. Greedy sequential maximal independent set and matching are parallel on average. In *24th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '12, Pittsburgh, PA, USA, June 25-27, 2012.* 308–317.

[31] Mark Braverman, Faith Ellen, Rotem Oshman, Toniann Pitassi, and Vinod Vaikuntanathan. 2013. A Tight Bound for Set Disjointness in the Message-Passing Model. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA.* 668–677.

[32] Mark Braverman, Ankit Garg, Denis Pankratov, and Omri Weinstein. 2013. From information to exact communication. In *Symposium on Theory of Computing Conference, STOC'13, June 1-4, 2013.* 151–160.

[33] Mark Braverman, Jieming Mao, and S. Matthew Weinberg. 2018. On Simultaneous Two-player Combinatorial Auctions. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, January 7-10, 2018.* 2256–2273.

[34] Mark Braverman and Ankur Moitra. 2013. An information complexity approach to extended formulations. In *Symposium on Theory of Computing Conference, STOC'13, June 1-4, 2013.* 161–170.

[35] Mark Braverman and Rotem Oshman. 2017. A Rounds vs. Communication Tradeoff for Multi-Party Set Disjointness. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017.* 144–155.

[36] Mark Braverman and Anup Rao. 2011. Information Equals Amortized Communication. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, October 22-25, 2011.* 748–757.

[37] Joshua Brody, Amit Chakrabarti, Ranganath Kondapally, David P. Woodruff, and Grigory Yaroslavtsev. 2014. Beyond set disjointness: the communication complexity of finding the intersection. In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014.* 106–113.

[38] Amit Chakrabarti, Graham Cormode, Ranganath Kondapally, and Andrew McGregor. 2010. Information Cost Tradeoffs for Augmented Index and Streaming Language Recognition. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA.* 387–396.

[39] Amit Chakrabarti, Graham Cormode, and Andrew McGregor. 2008. Robust lower bounds for communication and stream computation. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, May 17-20, 2008.* 641–650.

[40] Amit Chakrabarti and Sagar Kale. 2014. Submodular Maximization Meets Streaming: Matchings, Matroids, and More. In *Integer Programming and Combinatorial Optimization - 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings.* 210–221.

[41] Amit Chakrabarti and Anthony Wirth. 2016. Incidence Geometries and the Pass Complexity of Semi-Streaming Set Cover. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016.* 1365–1373.

[42] Deeparnab Chakrabarty, Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. 2017. Subquadratic submodular function minimization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017.* 1220–1231.

[43] Arkadev Chattopadhyay and Sagnik Mukhopadhyay. 2015. Tribes Is Hard in the Message Passing Model. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany.* 224–237.

[44] Stephen A. Cook. 1985. A Taxonomy of Problems with Fast Parallel Algorithms. *Information and Control* 64, 1-3 (1985), 2–21.

[45] Graham Cormode, Jacques Dark, and Christian Konrad. 2018. Approximating the Caro-Wei Bound for Independent Sets in Graph Streams. In *Combinatorial Optimization - 5th International Symposium, ISCO 2018, Marrakesh, Morocco, April 11-13, 2018, Revised Selected Papers.* 101–114.

[46] Graham Cormode, Jacques Dark, and Christian Konrad. 2018. Independent Sets in Vertex-Arrival Streams. *CoRR* abs/1807.08331 (2018).

[47] Graham Cormode and Hossein Jowhari. 2017. A second look at counting triangles in graph streams (corrected). *Theor. Comput. Sci.* 683 (2017), 22–30.

[48] William H. Cunningham. 1985. On submodular function minimization. *Combinatorica* 5, 3 (1985), 185–192.

[49] Erik D. Demaine, Piotr Indyk, Sepideh Mahabadi, and Ali Vakilian. 2014. On Streaming and Communication Complexity of the Set Cover Problem. In *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings.* 484–498.

[50] Shahar Dobzinski, Noam Nisan, and Sigal Oren. 2014. Economic efficiency requires interaction. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014.* 233–242.

[51] Pavol Duris, Zvi Galil, and Georg Schnitger. 1984. Lower Bounds on Communication Complexity. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA.* 81–91.

[52] Sebastian Eggert, Lasse Kliemann, and Anand Srivastav. 2009. Bipartite Graph Matchings in the Semi-streaming Model. In *Algorithms - ESA 2009, 17th Annual European Symposium, September 7-9, 2009. Proceedings.* 492–503.

[53] Yuval Emek and Adi Rosén. 2014. Semi-Streaming Set Cover - (Extended Abstract). In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I.* 453–464.

[54] Alina Ene and Huy L. Nguyen. 2018. Submodular Maximization with Nearly-optimal Approximation and Adaptivity in Nearly-linear Time. *CoRR* abs/1804.05379. To appear in SODA 2019. (2018).

[55] Alina Ene, Huy L. Nguyen, and Adrian Vladu. 2018. Submodular Maximization with Packing Constraints in Parallel. *CoRR* abs/1808.09987 (2018).

[56] Matthew Fahrbach, Vahab S. Mirrokni, and Morteza Zadimoghaddam. 2018. Non-monotone Submodular Maximization with Nearly Optimal Adaptivity Complexity. *CoRR* abs/1808.06932 (2018).

[57] Matthew Fahrbach, Vahab S. Mirrokni, and Morteza Zadimoghaddam. 2018. Submodular Maximization with Optimal Approximation, Adaptivity and Query Complexity. *CoRR* abs/1807.07889. To appear in SODA 2019. (2018).

[58] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. 2005. On graph problems in a semi-streaming model. *Theor. Comput. Sci.* 348, 2-3 (2005), 207–216. https://doi.org/10.1016/j.tcs.2005.09.013

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

[60] Dmitry Gavinsky, Julia Kempe, Iordanis Kerenidis, Ran Raz, and Ronald de Wolf. 2007. Exponential separations for one-way quantum communication complexity, with applications to cryptography. *STOC* (2007), 516–525.

[61] Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrovic, and Ronitt Rubinfeld. 2018. Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018.* 129–138.

[62] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. 2012. On the Communication and Streaming Complexity of Maximum Bipartite Matching. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '12).* SIAM, 468–485. http://dl.acm.org/citation.cfm?id=2095116.2095157

[63] Martin Grötschel, László Lovász, and Alexander Schrijver. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1, 2 (1981), 169–197.

[64] Sudipto Guha and Andrew McGregor. 2007. Lower Bounds for Quantile Estimation in Random-Order and Multi-pass Streaming. In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings.* 704–715.

[65] Sudipto Guha and Andrew McGregor. 2008. Tight Lower Bounds for Multi-pass Stream Computation Via Pass Elimination. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games.* 760–772.

[66] Venkatesan Guruswami and Krzysztof Onak. 2013. Superlinear Lower Bounds for Multipass Graph Processing. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013.* 287–298.

[67] Bjarni V. Halldórsson, Magnús M. Halldórsson, Elena Losievskaja, and Mario Szegedy. 2010. Streaming Algorithms for Independent Sets. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I.* 641–652.

[68] Bjarni V. Halldórsson, Magnús M. Halldórsson, Elena Losievskaja, and Mario Szegedy. 2016. Streaming Algorithms for Independent Sets in Sparse Hypergraphs. *Algorithmica* 76, 2 (2016), 490–501.

[69] Magnús M. Halldórsson, Xiaoming Sun, Mario Szegedy, and Chengu Wang. 2012. Streaming and Communication Complexity of Clique Approximation. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I.* 449–460.

[70] Sariel Har-Peled, Piotr Indyk, Sepideh Mahabadi, and Ali Vakilian. 2016. Towards Tight Bounds for the Streaming Set Cover Problem. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016.* 371–383.

[71] Nicholas James Alexander Harvey. 2008. *Matchings, matroids and submodular functions.* Ph.D. Dissertation. Massachusetts Institute of Technology.

[72] Nicholas J. A. Harvey. 2008. Matroid intersection, pointer chasing, and Young's seminormal representation of S_n. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008.* 542–549.

[73] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. 2016. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016.* 489–498.

[74] Gábor Ivanyos, Hartmut Klauck, Troy Lee, Miklos Santha, and Ronald de Wolf. 2012. New bounds on the classical and quantum communication complexity of some graph properties. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India.* 148–159.

[75] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. 2000. A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA.* 97–106.

[76] Satoru Iwata and James B. Orlin. 2009. A simple combinatorial algorithm for submodular function minimization. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009.* 1230–1237.

[77] Rahul Jain, Jaikumar Radhakrishnan, and Pranab Sen. 2003. A Direct Sum Theorem in Communication Complexity via Message Compression. In *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, June 30 - July 4, 2003. Proceedings.* 300–315.

[78] T. S. Jayram, Ravi Kumar, and D. Sivakumar. 2003. Two applications of information complexity. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA.* 673–682.

[79] Hossein Jowhari and Mohammad Ghodsi. 2005. New Streaming Algorithms for Counting Triangles in Graphs. In *Computing and Combinatorics, 11th Annual International Conference, COCOON 2005, Kunming, China, August 16-29, 2005, Proceedings.* 710–716.

[80] Sagar Kale and Sumedh Tirodkar. 2017. Maximum Matching in Two, Three, and a Few More Passes Over Graph Streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA.* 15:1–15:21.

[81] Bala Kalyanasundaram and Georg Schnitger. 1992. The Probabilistic Communication Complexity of Set Intersection. *SIAM J. Discrete Math.* 5, 4 (1992), 545–557.

[82] Michael Kapralov. 2013. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013.* 1679–1697. https://doi.org/10.1137/1.9781611973105.121

[83] Michael Kapralov and David P. Woodruff. 2014. Spanners and sparsifiers in dynamic streams. In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014.* 272–281.

[84] Christian Konrad, Frédéric Magniez, and Claire Mathieu. 2012. Maximum Matching in Semi-streaming with Few Passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings.* 231–242.

[85] Ilan Kremer, Noam Nisan, and Dana Ron. 1995. On randomized one-round communication complexity. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA.* 596–605.

[86] Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. 2013. Fast greedy algorithms in mapreduce and streaming. In *25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '13, Montreal, QC, Canada - July 23 - 25, 2013.* 1–10.

[87] Konstantin Kutzkov and Rasmus Pagh. 2014. Triangle Counting in Dynamic Graph Streams. In *Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings.* 306–318.

[88] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. 2015. A Faster Cutting Plane Method and its Implications for Combinatorial and Convex Optimization. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015.* 1049–1065.

[89] List of Open Problems in Sublinear Algorithms: Problem 14 [n. d.]. List of Open Problems in Sublinear Algorithms: Problem 14. https://sublinear.info/14.

[90] List of Open Problems in Sublinear Algorithms: Problem 22 [n. d.]. List of Open Problems in Sublinear Algorithms: Problem 22. https://sublinear.info/22.

[91] List of Open Problems in Sublinear Algorithms: Problem 29 [n. d.]. List of Open Problems in Sublinear Algorithms: Problem 29. https://sublinear.info/29.

[92] Michael Luby. 1986. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM J. Comput.* 15, 4 (1986), 1036–1053.

[93] Andrew McGregor. 2005. Finding Graph Matchings in Data Streams. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings.* 170–181. https://doi.org/10.1007/11538462_15

[94] Andrew McGregor. 2014. Graph stream algorithms: a survey. *SIGMOD Record* 43, 1 (2014), 9–20. http://doi.acm.org/10.1145/2627692.2627694

[95] Andrew McGregor, Sofya Vorotnikova, and Hoa T. Vu. 2016. Better Algorithms for Counting Triangles in Data Streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016.* 401–411.

[96] Andrew McGregor and Hoa T. Vu. 2017. Better Streaming Algorithms for the Maximum Coverage Problem. In *20th International Conference on Database Theory, ICDT 2017, March 21-24, 2017, Venice, Italy.* 22:1–22:18.

[97] Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. 1995. On data structures and asymmetric communication complexity. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA.* 103–111.

[98] J. Ian Munro and Mike Paterson. 1978. Selection and Sorting with Limited Storage. In *19th Annual Symposium on Foundations of Computer Science, Ann Arbor, Michigan, USA, 16-18 October 1978.* 253–258.

[99] Arkadi Nemirovski. 1994. On Parallel Complexity of Nonsmooth Convex Optimization. *J. Complexity* 10, 4 (1994), 451–463.

[100] Noam Nisan and Avi Wigderson. 1991. Rounds in Communication Complexity Revisited. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA.* 419–429.

[101] Christos H. Papadimitriou and Michael Sipser. 1984. Communication Complexity. *J. Comput. Syst. Sci.* 28, 2 (1984), 260–269.

[102] Stephen Ponzio, Jaikumar Radhakrishnan, and Srinivasan Venkatesh. 1999. The Communication Complexity of Pointer Chasing: Applications of Entropy and Sampling. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA.* 602–611.

[103] Alexander A. Razborov. 1992. On the Distributional Complexity of Disjointness. *Theor. Comput. Sci.* 106, 2 (1992), 385–390.

[104] Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. 2018. Computing Exact Minimum Cuts Without Knowing the Graph. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA.* 39:1–39:16.

[105] Barna Saha and Lise Getoor. 2009. On Maximum Coverage in the Streaming Model & Application to Multi-topic Blog-Watch. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, Sparks, Nevada, USA.* 697–708.

[106] Atish Das Sarma, Sreenivas Gollapudi, and Rina Panigrahy. 2011. Estimating PageRank on graph streams. *J. ACM* 58, 3 (2011), 13:1–13:19.

[107] Alexander Schrijver. 2000. A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time. *J. Comb. Theory, Ser. B* 80, 2 (2000), 346–355.

[108] Elad Verbin and Wei Yu. 2011. The Streaming Complexity of Cycle Counting, Sorting by Reversals, and Other Problems. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, January 23-25, 2011.* 11–25.

[109] Omri Weinstein and David P. Woodruff. 2015. The Simultaneous Communication of Disjointness with Applications to Data Streams. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, July 6-10, 2015, Proceedings, Part I.* 1082–1093.

[110] Andrew Chi-Chih Yao. 1979. Some Complexity Questions Related to Distributive Computing (Preliminary Report). In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA.* 209–213.

[111] Andrew Chi-Chih Yao. 1983. Lower Bounds by Probabilistic Arguments (Extended Abstract). In *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983.* 420–428.

[112] Amir Yehudayoff. 2016. Pointer chasing via triangular discrimination. *Electronic Colloquium on Computational Complexity (ECCC)* 23 (2016), 151.

[113] Mariano Zelke. 2011. Intractability of min- and max-cut in streaming graphs. *Inf. Process. Lett.* 111, 3 (2011), 145–150.