Polynomial Pass Semi-Streaming Lower Bounds for K-Cores and Degeneracy

Sepehr Assadi
Cheriton School of Computer Science, University of Waterloo, Canada

Prantar Ghosh
Department of Computer Science, Georgetown University, Washington, DC, USA

Bruno Loff
Department of Mathematics and LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

Parth Mittal
University of Waterloo, Canada

Sagnik Mukhopadhyay
University of Sheffield, UK

Abstract

The following question arises naturally in the study of graph streaming algorithms:

Is there any graph problem which is "not too hard", in that it can be solved efficiently with total communication (nearly) linear in the number n of vertices, and for which, nonetheless, any streaming algorithm with $\widetilde{O}(n)$ space (i.e., a semi-streaming algorithm) needs a polynomial $n^{\Omega(1)}$ number of passes?

Assadi, Chen, and Khanna [STOC 2019] were the first to prove that this is indeed the case. However, the lower bounds that they obtained are for rather non-standard graph problems.

Our first main contribution is to present the first polynomial-pass lower bounds for natural "not too hard" graph problems studied previously in the streaming model: ***k*-cores** and **degeneracy**. We devise a novel communication protocol for both problems with near-linear communication, thus showing that k-cores and degeneracy are natural examples of "not too hard" problems. Indeed, previous work have developed single-pass semi-streaming algorithms for approximating these problems. In contrast, we prove that any semi-streaming algorithm for *exactly* solving these problems requires (almost) $\Omega(n^{1/3})$ passes.

The lower bound follows by a reduction from a generalization of the **hidden pointer chasing (HPC)** problem of Assadi, Chen, and Khanna, which is also the basis of their earlier semi-streaming lower bounds.

Our second main contribution is improved **round-communication** lower bounds for the underlying communication problems at the basis of these reductions:

- We improve the previous lower bound of Assadi, Chen, and Khanna for HPC to achieve optimal bounds for this problem.
- We further observe that all current reductions from HPC can also work with a generalized version of this problem that we call **MultiHPC**, and prove an even stronger and optimal lower bound for this generalization.

These two results collectively allow us to improve the resulting pass lower bounds for semi-streaming algorithms by a polynomial factor, namely, from $n^{1/5}$ to $n^{1/3}$ passes.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Lower bounds and information complexity; Theory of computation → Graph algorithms analysis

Keywords and phrases Graph streaming, Lower bounds, Communication complexity, k-Cores and degeneracy

Digital Object Identifier 10.4230/LIPIcs.CCC.2024.7

Related Version *Full Version*: `http://arxiv.org/abs/2405.14835` [9]

Funding *Sepehr Assadi*: Supported in part by a Sloan Research Fellowship, an NSERC Discovery Grant, a University of Waterloo startup grant, and a Faculty of Math Research Chair grant.
Prantar Ghosh: Supported in part by NSF under award 1918989. Part of this work was done while he was at DIMACS, Rutgers University, supported in part by a grant (820931) to DIMACS from the Simons Foundation.
Bruno Loff: Funded by the European Union (ERC, HOFGA, 101041696). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. He was also supported by FCT through the LASIGE Research Unit, ref. UIDB/00408/2020 and ref. UIDP/00408/2020, and by CMAFcIO, FCT Project UIDB/04561/2020, `https://doi.org/10.54499/UIDB/04561/2020`.
Parth Mittal: Supported by a David R. Cheriton Graduate Scholarship and Sepehr Assadi's Sloan Research Fellowship, an NSERC Discovery grant, and a startup grant from University of Waterloo.
Sagnik Mukhopadhyay: Partially funded by the UKRI New Investigator Award, ref. EP/X03805X/1.

Acknowledgements The first named author would like to thank Yu Chen and Sanjeev Khanna for their collaboration in [7] that was the starting point of this project and Madhu Sudan for helpful conversations.

1 Introduction

Graph streaming algorithms process their inputs by making one or few passes over the edges of an input graph using limited memory. Algorithms that use space proportional to n, the number of vertices, are called *semi-streaming* algorithms. Since their introduction by [36], graph streaming algorithms have become one of the main theoretical research areas on processing massive graphs. We refer the interested reader to [53] for an introductory survey of earlier results on this topic.

In this work, we prove a polynomial-pass lower bound for any graph streaming algorithm that computes k-cores or degeneracy of a given graph. Our result is of interest from the point of view of proving strong lower bounds in the graph streaming model in addition to their direct implications for these two specific problems.

1.1 Polynomial Pass Lower Bounds in Graph Streams

Even though the study of multi-pass graph streaming algorithms started hand in hand with single-pass algorithms in [36], our understanding of powers and limitations of multi-pass algorithms, even for most basic problems, lags considerably behind. On one hand, for a problem like minimum cut, we have algorithms that in just $\widetilde{O}(n)$ space and two passes can solve the problem *exactly* [8][1] (see [7, Table 1] for a list of several such results). On the other hand, for some other basic problems such as undirected shortest path, directed reachability, and bipartite matching, the best known semi-streaming algorithms require $O(n^{1/2})$ [23], $n^{1/2+o(1)}$ [10, 51], and $n^{3/4+o(1)}$ [10] passes, respectively; yet, despite significant efforts, the best lower bound for any of these problems is still (even slightly below) $\Omega(\log n)$ passes [14, 22, 25, 40].

A key reason behind our weaker understanding of multi-pass streaming algorithms can be attributed to the lack of techniques for proving *super-logarithmic* pass lower bounds for semi-streaming algorithms. At this point, such lower bounds are only known for a handful

[1] See [61] for an implicit algorithm with the same bounds and [55] for the extension to weighted cuts in $O(\log n)$ passes.

of problems: clique and independent set [41], dominating set [1], Hamiltonian path [15], maximum cut [15, 46], vertex cover and coloring [3], exact Boolean CSPs [46], triangle detection [17, 58], and diameter computation [37]. Although, for all these problems, we can actually prove close-to-n pass lower bounds. Let us examine this dichotomy.

A quick glance at the list of problems above may suggest an intuitive difference between these problems and the ones like reachability or shortest path: the above list consists of problems that are computationally hard in a classical sense[2], suggesting that we are dealing with a "harder" class of problems in their case. While this intuition should not be taken as a formal evidence – as classical computational hardness does *not* imply streaming lower bounds (which are unconditional and information-theoretic) – [7] showed that one can also formally explain this dichotomy.

In particular, [7] observed that these strong streaming lower bounds happen only when the communication complexity of the problem at hand is $\Omega(n^2)$. Such a high lower-bound on the communication complexity *immediately* gives an $\widetilde{\Omega}(n)$-pass lower bound for semi-streaming algorithms via standard reductions. Whereas, for almost all problems of interest in the semi-streaming model, including shortest path, reachability, and bipartite matching, we already know an $\widetilde{O}(n)$ communication upper bound[3] (the protocol for bipartite matching was only discovered in [20] after the work of [7], but $\widetilde{O}(n^{3/2})$ communication protocols were known already [31, 42]). We refer the reader to [7, Section 1.1] for more context regarding these observations and prior techniques for $o(\log n)$ pass lower bounds.

Toward Stronger Streaming Lower Bounds

A natural question in light of these observations, already posed in [7], is the following:

> **Motivating question.** *Is there any graph problem which is "not too hard", in that it can be solved efficiently with communication (nearly) linear in the number n of vertices, and for which, nonetheless, any semi-streaming algorithm needs a polynomial $n^{\Omega(1)}$ number of passes?*

To address this question, [7] introduced a new (four-player) communication problem called **Hidden Pointer Chasing (HPC)**, which acts as a cross between *Set-Intersection* and *Pointer Chasing* problems, which are the main problems for, respectively, proving $\Omega(n^2)$ communication lower bounds on graphs, and $o(\log n)$-pass lower bounds for semi-streaming algorithms.

Roughly speaking, the HPC problem is defined as follows. There are four players paired into two groups. Each pair of players inside a group shares m instances of the Set-Intersection problem on m elements (each of the two players holds a subset of $[m]$ and they need to identify the unique intersecting element). 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, follow these pointers for a fixed number of steps, and then return the last element reached. See (full version [9], Definition 3.3) for the formal description.

This problem admits an efficient communication protocol with no limit on its number of rounds, but [7] showed that any r-round protocol that aims to find the $(r+1)$-th pointer in HPC requires $\Omega(m^2/r^2)$ communication. This places HPC squarely in the middle of previous

[2] These are standard NP-hard problems or admit some fine-grained hardness (for the latter two) [47, 60].
[3] This perhaps can be seen as this: a problem whose (unbounded round) communication complexity is already high has almost no place in the streaming model, which is a much weaker model algorithmically.

techniques and quite suitable for performing reductions to prove streaming lower bounds even for not-too-hard graph problems. Using this, [7] proved the first set of polynomial-pass graph streaming lower bounds in this class of problems: computing *lexicographically-first maximal independent set (LFMIS)* and *s-t minimum cut* on graphs with exponential edge-capacities both require $\widetilde{\Omega}(n^{1/5})$ passes to be solved by semi-streaming algorithms.

Despite the significant advances on multi-pass streaming lower bounds in the last couple of years [2,11–14,22,25–27,46], there is still no other known (not-too-hard) problem that admits a polynomial-pass lower bounds beside those of [7]. In addition, it is worth mentioning that, strictly speaking, neither LFMIS nor the version of s-t minimum cut in [7] completely fit the premise of our original question: LFMIS is not purely a graph problem as it is not invariant under labeling of the vertices, and s-t minimum cut studied in [7] involves (i) making the non-standard assumption of exponential capacities, and (ii) even for unit-capacity graphs, is not known to admit an $\widetilde{O}(n)$ communication protocol (see [20]).

We prove polynomial-pass lower bounds for two natural graph problems, k-cores and degeneracy, by reduction from a harder variant of the HPC problem which we call MultiHPC. We also present novel $\widetilde{O}(n)$ communication protocols for these two problems. These two results together give us the first natural instances of a positive answer to our motivating question.

These results further demonstrate the power of reductions from the HPC problem, as a technique for proving strong lower bounds in the graph streaming model, which are beyond the reach of other techniques. With this in mind, we improve the lower bound of [7] for the HPC problem to an optimal bound of $\Omega(m^2/r)$ communication, i.e., we improve the known bound by a factor of r. This contribution alone results in a polynomial improvement in the number of passes, for all lower bounds that follow via reductions from HPC (for instance, it immediately improves the bounds of [7] for LFMIS and exponential-capacity s-t minimum cut to $\widetilde{\Omega}(n^{1/4})$ passes).

But, as it turns out, all the known lower-bounds that follow by reduction from HPC also follow by reduction from MultiHPC. For this variant, we can prove an $\Omega(m^2)$ lower-bound for r rounds (since the input size for MultiHPC is $r \cdot m^2$), and this translates to an improved semi-streaming lower-bound of $\widetilde{\Omega}(n^{1/3})$ passes for all of the above problems.

1.2 k-Cores and Degeneracy in Graph Streams

For any undirected graph $G = (V, E)$ and integer $k \geqslant 1$, a *k-core* in G is a maximal set S of vertices such that the induced subgraph of G on S, denoted by $G[S]$, has a minimum degree of at least k. In other words, any vertex in S has at least k other neighbors in S.

k-Cores provide a natural notion of well-connectedness in massive graphs, and as such, computing k-cores (and more generally k-core decompositions; see, e.g., [50]) has been widely studied in databases [21,28,49], social networks [29,30,44], machine learning [6,33,39], among others [38,48,62].

As a result, in recent years, there has been a rapidly growing body of work on computing k-cores on massive graphs in parallel and streaming models of computation [29,30,33,39,50,62]. In particular, [33] presented a single-pass algorithm that for any $\varepsilon > 0$, computes a $(1-\varepsilon)$-approximation of every k-core in G (i.e., obtains a $(1-\varepsilon)$-approximate k-core decomposition) in $\widetilde{O}(n/\varepsilon^2)$ space (see also [62] for an earlier streaming algorithm and [39] for a closely related parallel algorithm).

The *degeneracy* of a graph $G = (V, E)$, denoted by $\kappa(G)$, is the largest integer $k \geqslant 0$ such that G contains a non-empty k-core. The simple greedy algorithm that at every step peels off the smallest degree vertex results in the so-called *degeneracy ordering* of G and $\kappa(G)$

is equal to the largest degree of a vertex removed in this peeling process [52]. Degeneracy is a standard measure of uniform sparsity and is closely related to other such notions like arboricity (which is within a factor 2 of degeneracy). Moreover, computing degeneracy is a subroutine for approximating various other problems such as arboricity [5], densest subgraph [24], $(\kappa+1)$ coloring [32].

The degeneracy problem, and closely related uniform-sparsity measures such as densest subgraph, have also been studied extensively in the graph streaming literature [4, 16, 18, 19, 34, 35, 54]. In particular, [34] provided an $O(\log n)$-pass semi-streaming algorithm that outputs a constant factor approximation to degeneracy, and [35] subsequently improved this to a single-pass $(1-\varepsilon)$-approximation in $\widetilde{O}(n/\varepsilon^2)$ space (see also [54] for densest subgraph and [4, 18] for degeneracy coloring).

In terms of lower bounds, [18] prove that any *single-pass* streaming algorithm that computes the exact value of degeneracy or approximates it within an additive factor of λ requires $\Omega(n^2)$ space or $\Omega(n^2/\lambda^2)$ space respectively. Our *polynomial-pass* lower bounds for k-cores and degeneracy, now in a very strong sense, rule out the possibility of extending any prior semi-streaming algorithms computing near-optimal solutions to these problems, to compute *exactly* optimal solutions.

1.3 Our Results

We give an informal presentation of our results here. The details can be found in the full version [9]. The first main result is our polynomial-pass lower bound for k-core computation and degeneracy.

▶ **Result 1** (Formalized in full version [9], Theorem 5.1)**.** For any integer $p \geqslant 1$, any p-pass streaming algorithm for computing the degeneracy of an input n-vertex graph requires $\widetilde{\Omega}(n^2/p^3)$ space. In particular, any semi-streaming algorithm for the problem requires $\widetilde{\Omega}(n^{1/3})$ passes.
Moreover, the same lower bounds also apply to the algorithms that given any integer $k \geqslant 1$, can check whether or not the input graph contains a non-empty k-core.

Result 1 provides a strongly negative answer to the question of obtaining semi-streaming algorithms for exact computation of degeneracy and k-cores in a small number of passes. We obtain Result 1 via a detailed and technical reduction, presented in Section 5 of the full version [9], from a variant of the Hidden Pointer Chasing (HPC) problem of [7], which we call Boolean Multilayer Hidden Pointer Chasing (BMHPC).

In a standard HPC problem, we are given m instances of m-bit Set-Intersection ($O(m^2)$ bits in total) and interpret the intersection point of each instance as pointing to a different instance among these m. We then wish to know the position we end up in after following $r+1$ pointers. In a Boolean variant, we only care to know the *parity* of the position we end up in. In a Multilayer variant, we are given r different layers, each layer with its own m instances (rm^2 bits in total), where we think of the intersection points at each layer as pointing to some instance in the next layer, and wish to know where we end up in the last layer by following these pointers.

In addition to the reduction from BMHPC, in Section 6 of the full version [9], we present a novel and non-trivial communication protocol that finds the degeneracy ordering (and thus degeneracy itself) and non-empty k-cores for any given k, using only $\widetilde{O}(n)$ communication. This communication upper bound thus places the k-core and degeneracy problems as perfect illustrations of a positive answer to our and [7]'s motivating question outlined earlier: namely, problems that prior techniques could not have proven any lower bound beyond $\log n$ passes. Result 1 thus constitutes the first set of natural graph problems with polynomial pass lower bounds for semi-streaming algorithms.

Our second main contribution is providing optimal lower bounds for the HPC problem and all its variants (Boolean, Multilayer, and Boolean Multilayer). A communication lower-bound of $\Omega(\frac{m^2}{r^2})$ was previously known for $(r-1)$-round protocols computing the r-th pointer in a (single layer) HPC problem. We prove the following.

▶ **Result 2** (Formalized in full version [9], Theorem 4.2)**.** For any integer $1 \leqslant r = O(\sqrt{m})$, any $(r-1)$-round protocol for computing the r-th pointer in the HPC problem on a universe of size m requires $\tilde{\Omega}(m^2/r)$ communication.

For any integer $r \geqslant 1$, any $(r-1)$-round protocol for computing the r-th pointer in the Multilayer HPC problem on a universe of size m requires $\tilde{\Omega}(m^2)$ communication.

Moreover, the same lower bounds hold for the Boolean versions of the above.

Result 2, by strengthening the lower bound of [7], allows us to prove polynomially stronger bounds on the number of passes of semi-streaming algorithms via reductions from HPC. As it turns out, every known reduction from HPC [7] can be easily converted to a reduction from MHPC. Our results thus imply improved pass lower bounds (from $\widetilde{\Omega}(n^{1/5})$ to $\widetilde{\Omega}(n^{1/3})$) for semi-streaming algorithms solving these problems. We capture this in the next corollary.

▶ **Corollary 1.** *For any integer $p \geqslant 1$, any p-pass streaming algorithm for the following problems on n-vertex graphs requires $\widetilde{\Omega}(n^2/p^3)$ space. In particular, any semi-streaming algorithm for these problems require $\widetilde{\Omega}(n^{1/3})$ passes.*

- *Computing the minimum s-t cut value in a weighted graph (with exponential edge capacities)*
- *Computing the lexicographically-first maximal independent set (LFMIS) of an undirected graph*

We obtain Result 2 by following the elegant analysis of pointer chasing problems due to [63] via the *triangular discrimination distance* between distributions, as opposed to more standard measures such as KL-divergence and total variation distance typically used in this context. This in turn requires extending the notion of "almost solving" for the Set-Intersection problem introduced by [7] (and further refined in [14]), to the triangular discrimination distance: roughly speaking, this corresponds to proving a lower bound for communication protocols that, instead of finding the intersecting element, change its distribution slightly from uniform distribution. The analysis in [7] measured this change by total variation distance, but now we need to do so by triangular discrimination distance instead. Finally, we prove a nearly-optimal lower bound on the communication-distance tradeoff for almost solving Set-Intersection in terms of the triangular discrimination distance.

2 Overview

In this version, we only present a high-level and informal overview of our techniques and proofs. All the technical details and formal proofs are available in the full version [9].

2.1 Hidden Pointer Chasing

The Multilayer Hidden Pointer Chasing (MHPC) problem, the starting point of our reductions, is defined as follows. The problem operates on two disjoint universes $\mathcal{X} = \{x_1, \ldots, x_m\}$ and $\mathcal{Y} = \{y_1, \ldots, y_m\}$. There are four players P_A, P_B, P_C, P_D, out of which P_A and P_B each hold rm subsets of $\mathcal{Y}$, called A_x^j and B_x^j for $x \in \mathcal{X}$ and $j \in [r]$, and P_C and P_D each hold rm subsets of $\mathcal{X}$, called C_y^j and D_y^j for $y \in \mathcal{Y}$ and $j \in [r]$, with the promise $|A_x^j \cap B_x^j| = 1$ and $|C_y^j \cap D_y^j| = 1$ for every $j \in [r]$, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. This means that each pair of sets of two of the players, e.g. A_x^j and B_x^j defines a pointer $\{y\} = A_x^j \cap B_x^j$, which we think of as pointing to the pair of sets in the next layer, C_y^{j+1} and D_y^{j+1}, belonging to the other two players. Following these pointers, and writing a singleton set as the element it contains, we define a sequence $z_0 = x_1$, $z_1 = A_{z_0}^1 \cap B_{z_0}^1, z_2 = C_{z_1}^2 \cap D_{z_1}^2, z_3 = A_{z_2}^3 \cap B_{z_2}^3$, *etc.* In the $\mathsf{MHPC}_{m,r}$ problem, the players wish to learn z_r. In the $\mathsf{BMHPC}_{m,r}$ problem, the players only need to learn one bit about z_r, that is, $b(z_r) := i \bmod 2$ where i is the index of z_r in $\mathcal{X}$ or $\mathcal{Y}$. Now, there is a very obvious way of doing this in r rounds, if the correct pair of players start: the players just follow the pointers, solving the necessary Set-Intersection instances. This costs them r rounds with $O(m)$ bits of communication per round, for a total of rm bits. However, we will show:

▶ **Theorem 2.** *Any randomized protocol with less than r rounds, or even any randomized protocol with r rounds which is misaligned, in that the "wrong" pair of players starts to speak, cannot solve $\mathsf{BMHPC}_{m,r}$ correctly with fewer than $\Omega(m^2)$ bits of communication.*

This theorem is proven by combining ideas from three different previous works: [7], [14], and [63]. But first, let us give an overall intuition for why it should be expected to hold.

In a misaligned r-round protocol for $\mathsf{BMHPC}_{m,r}$, it is players P_C and P_D who begin the protocol by talking with each other. This means that the "wrong" pair of players begin to speak, in the sense that they wish to compute the value $\{z_1\} = A_1^1 \cap B_1^1$, but this instance is with P_A and P_B, so they have no way to do this. So the first round cannot say anything about z_1: the best P_C and P_D can do is send some information about all of their Set-Intersection instances, without knowing which one is important. This means that each bit that P_C and P_D communicate with P_A and P_B in the first round can only reveal $\frac{1}{m}$ bits of information about the average instance. But now in the next round, P_A and P_B, although they know z_1, cannot have learned much information about $C_{z_1}^2$ or $D_{z_1}^2$. But then, how can they say anything about $\{z_2\} = C_{z_1}^2 \cap D_{z_1}^2$? The difficult situation is now reversed! This "always one step behind" situation is similar to what happens for pointer chasing [57, 59, 63], except now the pointers are "hidden" behind set intersection instances.

A previous paper of Assadi, Chen and Khanna [7] showed a lower-bound of $\Omega(\frac{m^2}{r^2})$ for the (single layer) Hidden Pointer Chasing problem $\mathsf{HPC}_{m,r}$, which is a version of $\mathsf{MHPC}_{m,r}$ where all the layers are identical ($A_i^j, B_i^j, C_i^j, D_i^j$ is the same for all $j \in [r]$). The lower-bound was proven via an information-theoretic argument. They first show that any low-round protocol for HPC must be "almost solving" a set intersection instance on one of the rounds. They then show that this is impossible via an information complexity argument, akin to the lower-bound on the information complexity for set disjointness. However, a later paper by Assadi and Raz [14] directly showed that any protocol that "almost solves" set intersection can be used to obtain a protocol that exactly solves set intersection (hence the term "almost solving"). This would allow us to replace the *ad hoc* information complexity argument in [7], and instead appeal, in a black-box fashion, to a previously known lower-bound on the information complexity of Set-Intersection [43].

One could take these previous lower-bounds for $\mathsf{HPC}_{m,r}$, and prove a lower-bound of $\Omega(\frac{m^2}{r})$ for $\mathsf{MHPC}_{m,r}$, but not the lower-bound of $\Omega(m^2)$ which we obtain here. The insufficiency of these previous proofs comes from the notion of "almost solving" that is used. There, a protocol is said to "almost solve" Set-Intersection if the distribution of the intersection point is sufficiently changed by the knowledge gained from the protocol's execution. More precisely, if the distribution of the intersection point $\mu(A \cap B \mid \Pi)$, conditioned on knowing the transcript Π, is sufficiently far away, in total variation distance (TVD), from the distribution of the intersection point $\mu(A \cap B)$, as it is known before the protocol begins. The quadratic margin in terms of r is ultimately a result of the quadratic loss between TVD and Shannon information, in the use of Pinsker's inequality.

This same issue was the cause of a decades-long open problem on the complexity of (non-hidden) pointer chasing. Nisan and Wigderson proved in 1991 [56] that any r-round protocol for pointer chasing, where the wrong player starts, needs to communicate $\omega(\frac{m}{r^2})$ bits. But there is a simple upper bound of $O(\frac{m}{r})$. In 2000, Klauck [45] gave a non-constructive proof of a matching lower-bound. That is, he showed that the randomized communication complexity is indeed $\Omega(\frac{m}{r})$, but without providing a hard distribution, which must exist via Yao's Principle. This problem remained open until 2019, when Yehudayoff [63] showed that the distributional complexity of pointer chasing is $\Omega(\frac{m}{r})$ under the uniform distribution, whenever $r \ll \sqrt{m}$. The proof used a measure of information called *triangular discrimination*, which had never before been used in the lower-bounds literature.

Thus, being simultaneously aware of the three works of [7], [14], and [63], one is naturally led to ask if they can be combined in such a way as to improve the $\frac{m^2}{r^2}$ lower bound for HPC, to $\frac{m^2}{r}$? And could we then prove a lower bound of $\Omega(m^2)$ for Multilayer HPC?

This turns out to be the case. We are not only able to prove Theorem 2, but we also improve the lower bound for (single layer) HPC:

▶ **Theorem 3.** *Let $r = O(\sqrt{m})$. Then, any randomized protocol with less than r rounds, or even any misaligned randomized protocol with r rounds, cannot solve $\mathsf{BHPC}_{m,r}$ correctly with fewer than $\Omega(\frac{m^2}{r})$ bits of communication.*

The key insight in the new lower bounds is that the notion of "almost solving" an instance of Set-Intersection can be adapted to use triangular discrimination instead of TVD. Two issues then need to be addressed.

First, we must show that a low-round protocol for HPC or MHPC must be "almost solving" (in the new sense) an instance of Set-Intersection in one of the rounds. The proofs follows the general outline of [7], but need to be adapted to use triangular discrimination instead of TVD. To see that it works, one must first understand that triangular discrimination obeys a property analogous to TVD, saying that the expected value of $f(x)$, when x is sampled by some distribution μ, is not too far from the expected value of $f(x)$ when x is sampled by a different distribution ν, if μ and ν are close with respect to triangular discrimination. This is obvious for TVD, but not as obvious for triangular discrimination. It is also not obvious how to adapt the proof to Multilayer HPC, in a way that works for any number of rounds $r \leqslant m$.

Second, we must show that a low-information protocol that "almost solves" (in the new sense) Set-Intersection can still be used to obtain a low-information protocol that exactly solves Set-Intersection. The proof is similar to [14]. In that paper, a reduction is given which solves a given Set-Intersection instance by sampling $O(1)$ runs of a protocol that "almost solves" Set-Intersection in terms of TVD. We reinterpret their reduction as using the almost-solving protocol to assign scores to elements (predicting how likely they are to be the intersecting element), and come up with a new scoring function which allows a reduction from set intersection to almost-solving with respect to positive triangular discrimination.

2.2 Reduction to Degeneracy

We give a high-level overview of the key idea behind the reduction from BMHPC to the streaming problem of finding the graph degeneracy. First, suppose that we want to show a streaming lower bound for the harder problem of finding a degeneracy ordering. In the classical offline setting, we can obtain such an ordering by the peeling algorithm that recursively removes the min-degree node from the graph and appends it to the end of the ordering. But naively implementing this algorithm in the semi-streaming setting seems difficult since it is inherently sequential. We can store the degree of each node in semi-streaming space and find the min-degree node v in the graph. After we remove v, we need to find a min-degree node v' in the new graph $G \setminus \{v\}$. But at the beginning of the stream, we did not know which node v is, and hence might not have stored enough of its neighbors so as to update their degrees and find v'. Hence, naively, we need to make a new pass for each peeled vertex, which takes $\Theta(n)$ passes in total for an n-node graph. One might wonder whether *any* semi-streaming algorithm for degeneracy ordering would need close to these many passes. If so, how do we prove it?

Consider just the basic problem of finding a min-degree node in an n-node graph, which is the primitive for finding a degeneracy ordering. It can be shown via a simple reduction that a streaming algorithm for this problem can be used to solve SetInt_n, the Set-Intersection communication problem with universe size n. As noted above, finding the degeneracy ordering translates to finding a sequence of nodes that have smallest degree in the remaining graph. This means we can use it to basically solve a sequence of $\mathsf{SetInt}_{\Theta(n)}$ instances. These instances are, however, not independent. The solution to the first instance gives a min-degree node in the original graph, whose removal leads to the second instance; solving this instance reveals the third instance, and so on and so forth. This gives a flavor of a combination of SetInt and pointer chasing, where each pointer is revealed by solving a SetInt instance corresponding to the previous pointer. This is precisely the concept behind HPC (or MHPC for that matter)! Hence, it is plausible that the degeneracy ordering problem can be reduced from MHPC, and we embark on the journey to find such a reduction.

Recall the definition of MHPC from Section 2.1. Given an instance of MHPC, we construct the following layered graph with $r + 1$ layers $L_0, \ldots, L_r$. Each layer has m nodes: the nodes in the even layers correspond to x_i's and the ones in the odd layers correspond to y_i's. The edges of the graph are always between two consecutive layers. The players P_A and P_B encode the sets $A^1_{x_i}$ and $B^1_{x_i}$ by adding edges between L_0 and L_1. Consider the following encoding: for each $i, j \in [m]$, if $y_j \in A^1_{x_i}$, then P_A adds an edge from the ith node in L_0 to the jth node in L_1. P_B does the analogous construction for the elements in $B^1_{x_i}$ (note that this can lead to parallel edges). P_C and P_D encode the sets $C^2_{y_i}$ and $D^2_{y_i}$ by adding edges between L_1 and L_2 in the analogous way. Again, P_A and P_B encode $A^3_{y_i}$ and $B^3_{y_i}$ with edges between L_2 and L_3, and this proceeds alternately until the relevant players add the edges between L_r and L_{r+1}.

Let v_0 be the first node in L_0; recall that it corresponds to $x_1 = z_0$. Assume that v_0 is the min-degree node in the graph with $\deg(v_0) = d - 1$ and all other nodes have the same degree d. Again, recall that $z_1 = A^1_{z_0} \cap B^1_{z_0}$. By construction and by the unique-intersection promise of the SetInt instances of MHPC, v_0 has two parallel edges to the node representing z_1 in L_1; call this node v_1. To all other nodes in L_1, v_0 has at most one edge. Hence, when the peeling algorithm deletes v_0 from the graph, only the degree of v_1 drops by 2, i.e., $\deg(v_1)$ becomes $d - 2$; all other nodes have at most a drop of 1 in degree, i.e., have degree $\geqslant d - 1$. Thus, v_1 becomes the new min-degree node in the graph. Now, when v_1 is deleted, by similar logic, the node v_2 in L_2, corresponding to the element $z_2 = C^2_{z_1} \cap D^2_{z_1}$, becomes a min-degree node in the remaining graph with $\deg(v_2) = d - 2$. However, now some node in L_0 might also have degree $d - 2$; this is the case when $z_1 = A^1_{x_i} \cap B^1_{x_i}$ for some $i \neq 1$ as well. Now assume that the peeling algorithm breaks ties by choosing a node in the highest layer among all min-degree nodes (and arbitrarily within the highest layer). Then, indeed it chooses v_2 as the next node to peel (since it is the unique min-degree node in L_2, again by the SetInt promise). Thus, it follows inductively that the ith iteration of the peeling algorithm removes the node corresponding to z_{i-1} in L_{i-1}. Hence, the $(r+1)$th node in the degeneracy ordering can be used to identify z_r.

The above high-level idea has quite a few strong assumptions. The challenge is now to get rid of them. We list these challenges and then describe how we overcome them.

(i) The constructed graph has parallel edges. Then the reduction would only prove a lower bound against algorithms that can handle multigraphs, which is much weaker than a lower bound against algorithms that work on simple graphs.

(ii) We assume that the tie-breaking is done by the peeling algorithm so as to pick a vertex in the highest layer. It is not at all clear how to get rid of this assumption in a straightforward way.

(iii) We also assume that we can set the initial degrees in such a way that v_0 has degree $d-1$ and all other nodes have degree d. It is not clear that we can do this while preserving the relevant properties of the construction.

(iv) Even if we can overcome the above challenges and the reduction goes through, then we prove a lower bound for finding degeneracy ordering, which is (at least formally) harder than the problem of finding the degeneracy value. Ideally we would like to show the lower bound for the simplest variant of the problem: checking whether degeneracy of the graph is smaller than a given value k or not.

To get around (i), we modify the construction to have a pair of nodes represent each element. The edge construction is done in the following way. Suppose the pair (u_1, u_2) represents an element x_i in layer $\ell - 1$, and (w_1, w_2) represents y_j in layer ℓ. If $y_j \in A^\ell_{x_i}$, then we add edges from u_1 to both w_1 and w_2. Similarly, if $y_j \in B^\ell_{x_i}$, then we add edges from u_2 to both w_1 and w_2. Note that if $y_j \in A^\ell_{x_i} \cap B^\ell_{x_i}$, then we have all 4 cross edges between the two pairs, and otherwise we have only 2 edges between them, one on each w_i. Hence, when u_1 and u_2 are removed, both w_1 and w_2 lose degree by 2 if y_j is the intersecting element, and otherwise they only lose degree by 1. This captures the property of the reduction that we want, without constructing parallel edges.

For (ii), we do something more elaborate. On a high level, we duplicate each of the layers $L_1, \ldots, L_r$ to provide a "padding" between two initially-consecutive layers. This padding has additional nodes that create an asymmetry between the layer preceding it and the one succeeding it. This asymmetry ensures that the degrees of the nodes in the higher layer drop more than those in the lower layer. Then, we can proceed with the peeling algorithm as planned.

To handle (iii), we show that once we are done with the construction based on the MHPC instance, we can consider each node, look at its degree, and add edges from it to some auxiliary vertices so as to reach its "target degree". We need to be careful about two things: one, we preserve the properties of the construction so that the reduction goes through, and two, we do not add too many new nodes that might make the bound obtained from the reduction weak. We succeed in achieving a construction without violating the above.

Finally, for (iv), we observe that while we gave the above outline for a reduction from MHPC, the "easier" boolean version BMHPC has a similar lower bound. We then succeed in extending the ideas to reduce the boolean problem of "checking whether degeneracy $\leqslant k$"

from the BMHPC problem, thus obtaining the desired lower bound for this simple variant. For reduction from BMHPC, where the goal is to output just the bit $b(z_r)$ (see definition in Section 2.1) rather than z_r, we need to make non-trivial modifications in the graph: we join the nodes which represent the bit-1 elements in the last layer, to some "special nodes" S. The other nodes in the last layer are not joined to them. The special nodes are also adjacent to all nodes in the other layers. We show that if $b(z_r) = 1$, then after the peeling algorithm removes the nodes corresponding to z_r in the last layer, the degrees of the special nodes drop enough such that all the remaining vertices get peeled one by one, while having degree at most some value k during deletion. This implies that the graph has degeneracy $\leqslant k$. Otherwise, if $b(z_r) = 0$, we show that after peeling off the nodes representing z_r in the last layer, the minimum vertex-degree in the remaining subgraph is at least $k + 1$, implying that the degeneracy of the graph must be at least $k + 1$. We give the detailed reduction and proof in Section 5 of the full version [9].

2.3 Communication Upper Bounds for Degeneracy

We give a short overview of the $\widetilde{O}(n)$ communication protocol for computing the degeneracy of a graph. In the two player communication model, the edges of the input graph G are split into two disjoint sets E_A and E_B given to the players Alice and Bob respectively, and they wish to find the degeneracy of G. Note that the search problem (finding the degeneracy) reduces to its decision counterpart (is the degeneracy $\leqslant k$?) by a binary search, costing only a $\log n$ multiplicative factor in the communication cost. Hence, we focus on the version where Alice and Bob are additionally given an integer k, and wish to decide if the degeneracy of G is at most k.

To solve this decision problem, we implement the following version of the peeling algorithm in a communication protocol: while there is a vertex of degree $\leqslant k$, remove it. If the graph is non-empty at the end, reject, otherwise accept. The main challenge in adapting this algorithm is that in the worst case, it seems to update the degree of almost all vertices in G after each deletion, and there is no way to do that without a lot of communication.

However, we observe that if a vertex has degree at least $k + \sqrt{n}$, then it cannot be deleted for the next $\sqrt{n}$ iterations (since each iteration can reduce its degree by at most one). This observation alone gives us the following $\widetilde{O}(n\sqrt{n})$ communication protocol:

1. Compute the degree of each vertex in G.
2. Ignore all vertices of degree $\geqslant k + \sqrt{n}$ while performing $\sqrt{n}$ rounds of the trivial peeling algorithm.
3. Go to Step 1.

Note that the communication in Step 2 comes from Alice and Bob sending each other the low degree ($< k + \sqrt{n}$) neighbors of the vertex deleted in each iteration of the peeling algorithm. We observe that while a vertex has degree $\geqslant k + \sqrt{n}$, it is not listed in Step 2, and once its degree falls below the threshold of $k + \sqrt{n}$, it is listed at most $\sqrt{n}$ times due to Step 2. Thus, the total communication due to Step 2 over the course of the entire protocol is bounded by $\widetilde{O}(n\sqrt{n})$. Also, we recompute the degrees of *all* vertices (which costs $O(n \log n)$ communication each time) at most $\sqrt{n}$ times; these two facts combined give us the desired bound.

To get an improved $\widetilde{O}(n)$ communication protocol, we extend the idea above to partition the vertices into $\log n$ sets, where the i-th set contains vertices of degree between $k + 2^{i-1}$ and $k + 2^i$. While the global approach (of simply ignoring the high-degree vertices for $\sqrt{n}$ steps) does not work any more, we are able to make a more local argument as follows: for a vertex of degree $k + \ell$ to be deleted, it must lose at least ℓ neighbors, which means it must lose $\ell/2$ neighbors in either Alice's or Bob's edge set. But now the players can just track this "private" loss of degree of each vertex, and communicate to update the degree of a vertex in the i-th set only when either private degree falls by at least 2^{i-2}. We are able to show that the degree of each vertex is updated $O(\log n)$ times over the entire course of this new protocol, and hence the total communication is $\widetilde{O}(n)$. We further show that this can be extended to finding a k-core of the graph with $\widetilde{O}(n)$ communication. Thus, we establish finding degeneracy and k-core as not-too-hard problems.

References

1 Sepehr Assadi A. 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*, pages 321–335, 2017.

2 Sepehr Assadi A. A two-pass (conditional) lower bound for semi-streaming maximum matching. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9–12, 2022*, pages 708–742. SIAM, 2022.

3 Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Ami Paz. Smaller cuts, higher lower bounds. *ACM Trans. Algorithms*, 17(4):30:1–30:40, 2021.

4 Noga Alon and Sepehr Assadi. Palette sparsification beyond (Δ+1) vertex coloring. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 6:1–6:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

5 Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.

6 J. Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. Large scale networks fingerprinting and visualization using the k-core decomposition. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 41–50, 2005.

7 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Polynomial pass lower bounds for graph streaming algorithms. In *STOC*, pages 265–276. ACM, 2019.

8 Sepehr Assadi and Aditi Dudeja. A simple semi-streaming algorithm for global minimum cuts. In *SOSA*, pages 172–180. SIAM, 2021.

9 Sepehr Assadi, Prantar Ghosh, Bruno Loff, Parth Mittal, and Sagnik Mukhopadhyay. Polynomial pass semi-streaming lower bounds for k-cores and degeneracy. *arXiv preprint*, 2024. arXiv:2405.14835.

10 Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. In *SODA*, pages 627–669. SIAM, 2022.

11 Sepehr Assadi, Gillat Kol, Raghuvansh Saxena, and Huacheng Yu. Multi-pass graph streaming lower bounds for cycle counting, max-cut, matching size, and other problems. In *61st Annual IEEE Symposium on Foundations of Computer Science, FOCS (to appear)*, 2020.

12 Sepehr Assadi, Gillat Kol, and Zhijun Zhang. Rounds vs communication tradeoffs for maximal independent sets. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 – November 3, 2022*, pages 1193–1204. IEEE, 2022.

13 Sepehr Assadi and Vishvajeet N. Graph streaming lower bounds for parameter estimation and property testing via a streaming XOR lemma. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 612–625. ACM, 2021.

14 Sepehr Assadi and Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In *FOCS*, pages 342–353. IEEE, 2020.

15 Nir Bachrach, Keren Censor-Hillel, Michal Dory, Yuval Efron, Dean Leitersdorf, and Ami Paz. Hardness of distributed optimization. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 – August 2, 2019*, pages 238–247. ACM, 2019.

16 Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. Densest subgraph in streaming and mapreduce. *PVLDB*, 5(5):454–465, 2012.

17 Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. 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.*, pages 623–632, 2002.

18 Suman K. Bera, Amit Chakrabarti, and Prantar Ghosh. Graph coloring via degeneracy in streaming and other space-conscious models. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, pages 11:1–11:21, 2020.

19 Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos E. Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 173–182, 2015.

20 Joakim Blikstad, Jan van den Brand, Yuval Efron, Sagnik Mukhopadhyay, and Danupon Nanongkai. Nearly optimal communication and query complexity of bipartite matching. In *FOCS*, pages 1174–1185. IEEE, 2022. `doi:10.1109/FOCS54457.2022.00113`.

21 Francesco Bonchi, Francesco Gullo, Andreas Kaltenbrunner, and Yana Volkovich. Core decomposition of uncertain graphs. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani, editors, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA – August 24–27, 2014*, pages 1316–1325. ACM, 2014.

22 Amit Chakrabarti, Prantar Ghosh, Andrew McGregor, and Sofya Vorotnikova. Vertex ordering problems in directed graph streams. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1786–1802, 2020.

23 Yi-Jun Chang, Martin Farach-Colton, Tsan-sheng Hsu, and Meng-Tsung Tsai. Streaming complexity of spanning tree computation. In *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, pages 34:1–34:19, 2020.

24 Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In Klaus Jansen and Samir Khuller, editors, *Approximation Algorithms for Combinatorial Optimization, Third International Workshop, APPROX 2000, Saarbrücken, Germany, September 5-8, 2000, Proceedings*, volume 1913 of *Lecture Notes in Computer Science*, pages 84–95. Springer, 2000.

25 Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh R. Saxena, Zhao Song, and Huacheng Yu. Almost optimal super-constant-pass streaming lower bounds for reachability. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 570–583. ACM, 2021.

26 Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh R. Saxena, Zhao Song, and Huacheng Yu. Near-optimal two-pass streaming algorithm for sampling random walks over directed graphs. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 52:1–52:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

27 Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh R. Saxena, Zhao Song, and Huacheng Yu. Towards multi-pass streaming lower bounds for optimal approximation of max-cut. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 878–924. SIAM, 2023.

28 Deming Chu, Fan Zhang, Xuemin Lin, Wenjie Zhang, Ying Zhang, Yinglong Xia, and Chenyi Zhang. Finding the best k in core decomposition: A time and space optimal solution. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*, pages 685–696. IEEE, 2020.

29 Laxman Dhulipala, Guy E. Blelloch, and Julian Shun. Julienne: A framework for parallel graph algorithms using work-efficient bucketing. In Christian Scheideler and Mohammad Taghi Hajiaghayi, editors, *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, pages 293–304. ACM, 2017.

30 Laxman Dhulipala, Guy E. Blelloch, and Julian Shun. Theoretically efficient parallel graph algorithms can be fast and scalable. In Christian Scheideler and Jeremy T. Fineman, editors, *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018*, pages 393–404. ACM, 2018.

31 Shahar Dobzinski, Noam Nisan, and Sigal Oren. Economic efficiency requires interaction. *Games Econ. Behav.*, 118:589–608, 2019.

32 Paul Erdős and András Hajnal. On chromatic number of graphs and set-systems. *Acta Math. Acad. Sci. Hungar*, 17(61-99):1, 1966.

33 Hossein Esfandiari, Silvio Lattanzi, and Vahab S. Mirrokni. Parallel and streaming algorithms for k-core decomposition. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1396–1405. PMLR, 2018.

34 Martin Farach-Colton and Meng-Tsung Tsai. Computing the degeneracy of large graphs. In Alberto Pardo and Alfredo Viola, editors, *LATIN 2014: Theoretical Informatics – 11th Latin American Symposium, Montevideo, Uruguay, March 31 – April 4, 2014. Proceedings*, volume 8392 of *Lecture Notes in Computer Science*, pages 250–260. Springer, 2014.

35 Martin Farach-Colton and Meng-Tsung Tsai. Tight approximations of degeneracy in large graphs. In *LATIN 2016: Theoretical Informatics – 12th Latin American Symposium, Ensenada, Mexico, April 11-15, 2016, Proceedings*, pages 429–440, 2016.

36 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.

37 Silvio Frischknecht, Stephan Holzer, and Roger Wattenhofer. Networks cannot compute their diameter in sublinear time. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1150–1162. SIAM, 2012.

38 Edoardo Galimberti, Francesco Bonchi, Francesco Gullo, and Tommaso Lanciano. Core decomposition in multilayer networks: Theory, algorithms, and applications. *ACM Trans. Knowl. Discov. Data*, 14(1):11:1–11:40, 2020.

39 Mohsen Ghaffari, Silvio Lattanzi, and Slobodan Mitrovic. Improved parallel algorithms for density-based network clustering. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2201–2210. PMLR, 2019.

40 Venkatesan Guruswami and Krzysztof Onak. 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*, pages 287–298, 2013.

41 Magnús M. Halldórsson, Xiaoming Sun, Mario Szegedy, and Chengu Wang. 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*, pages 449–460, 2012.

42 Gábor Ivanyos, Hartmut Klauck, Troy Lee, Miklos Santha, and Ronald de Wolf. New bounds on the classical and quantum communication complexity of some graph properties. In *FSTTCS*, volume 18 of *LIPIcs*, pages 148–159. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012.

43 T. S. Jayram, Ravi Kumar, and D. Sivakumar. Two applications of information complexity. In *STOC*, pages 673–682. ACM, 2003.

44 Wissam Khaouid, Marina Barsky, S. Venkatesh, and Alex Thomo. K-core decomposition of large networks on a single PC. *Proc. VLDB Endow.*, 9(1):13–23, 2015.

45 Hartmut Klauck. On quantum and probabilistic communication: Las vegas and one-way protocols. In *32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 644–651, 2000.

46 Gillat Kol, Dmitry Paramonov, Raghuvansh R. Saxena, and Huacheng Yu. Characterizing the multi-pass streaming complexity for solving boolean csps exactly. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 80:1–80:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

47 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1272–1287. SIAM, 2016.

48 Chao Li, Li Wang, Shiwen Sun, and Chengyi Xia. Identification of influential spreaders based on classified neighbors in real-world complex networks. *Appl. Math. Comput.*, 320:512–523, 2018.

49 Conggai Li, Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. Efficient progressive minimum k-core search. *Proc. VLDB Endow.*, 13(3):362–375, 2019.

50 Quanquan C. Liu, Jessica Shi, Shangdi Yu, Laxman Dhulipala, and Julian Shun. Parallel batch-dynamic algorithms for k-core decomposition and related graph problems. In Kunal Agrawal and I-Ting Angelina Lee, editors, *SPAA '22: 34th ACM Symposium on Parallelism in Algorithms and Architectures, Philadelphia, PA, USA, July 11–14, 2022*, pages 191–204. ACM, 2022.

51 Yang P. Liu, Arun Jambulapati, and Aaron Sidford. Parallel reachability in almost linear work and square root depth. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1664–1686. IEEE Computer Society, 2019.

52 David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, 30(3):417–427, 1983.

53 Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Rec.*, 43(1):9–20, 2014.

54 Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa T. Vu. Densest subgraph in dynamic graph streams. In *Mathematical Foundations of Computer Science 2015 – 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II*, pages 472–482, 2015.

55 Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted min-cut: sequential, cut-query, and streaming algorithms. In *STOC*, pages 496–509. ACM, 2020. doi:10.1145/3357713.3384334.

56 Noam Nisan and Avi Wigderson. 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*, pages 419–429, 1991.

57 Noam Nisan and Avi Wigderson. Rounds in communication complexity revisited. *SIAM J. Comput.*, 22(1):211–219, 1993.

58 Christos H. Papadimitriou and Michael Sipser. Communication complexity. *J. Comput. Syst. Sci.*, 28(2):260–269, 1984.

59 Stephen Ponzio, Jaikumar Radhakrishnan, and Srinivasan Venkatesh. 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*, pages 602–611, 1999.

60 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 515–524. ACM, 2013.

61 Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. Computing exact minimum cuts without knowing the graph. In *ITCS*, volume 94 of *LIPIcs*, pages 39:1–39:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.39.

62 Ahmet Erdem Sariyüce, Bugra Gedik, Gabriela Jacques-Silva, Kun-Lung Wu, and Ümit V. Çatalyürek. Streaming algorithms for k-core decomposition. *Proc. VLDB Endow.*, 6(6):433–444, 2013.

63 Amir Yehudayoff. Pointer chasing via triangular discrimination. *Comb. Probab. Comput.*, 29(4):485–494, 2020.