# Parameterized Complexity of Streaming Diameter and Connectivity Problems

**Jelle J. Oostveen**
Department of Information and Computing Sciences, Utrecht University, The Netherlands

**Erik Jan van Leeuwen**
Department of Information and Computing Sciences, Utrecht University, The Netherlands

## Abstract

We initiate the investigation of the parameterized complexity of Diameter and Connectivity in the streaming paradigm. On the positive end, we show that knowing a vertex cover of size $k$ allows for algorithms in the Adjacency List (AL) streaming model whose number of passes is constant and memory is $\mathcal{O}(\log n)$ for any fixed $k$. Underlying these algorithms is a method to execute a breadth-first search in $\mathcal{O}(k)$ passes and $\mathcal{O}(k \log n)$ bits of memory. On the negative end, we show that many other parameters lead to lower bounds in the AL model, where $\Omega(n/p)$ bits of memory is needed for any $p$-pass algorithm even for constant parameter values. In particular, this holds for graphs with a known modulator (deletion set) of constant size to a graph that has no induced subgraph isomorphic to a fixed graph $H$, for most $H$. For some cases, we can also show one-pass, $\Omega(n \log n)$ bits of memory lower bounds. We also prove a much stronger $\Omega(n^2/p)$ lower bound for Diameter on bipartite graphs.

Finally, using the insights we developed into streaming parameterized graph exploration algorithms, we show a new streaming kernelization algorithm for computing a vertex cover of size $k$. This yields a kernel of $2k$ vertices (with $\mathcal{O}(k^2)$ edges) produced as a stream in poly$(k)$ passes and only $\mathcal{O}(k \log n)$ bits of memory.

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

**Keywords and phrases** Stream, Streaming, Graphs, Parameter, Complexity, Diameter, Connectivity, Vertex Cover, Disjointness, Permutation

**Digital Object Identifier** 10.4230/LIPIcs.IPEC.2022.24

**Related Version** *Full Version*: `https://arxiv.org/abs/2207.04872` [47]

**Funding** *Jelle J. Oostveen*: This author is partially supported by the NWO grant OCENW.KLEIN.114 (PACAN).

**Acknowledgements** We thank the reviewers for their helpful comments and feedback.

## 1 Introduction

Graph algorithms, such as to compute the diameter of an unweighted graph (Diameter) or to determine whether it is connected (Connectivity), often rely on keeping the entire graph in (random access) memory. However, very large networks might not fit in memory. Hence, graph streaming has been proposed as a paradigm where the graph is inspected through a so-called stream, in which its edges appear one by one [38]. To compensate for the assumption of limited memory, multiple passes may be made over the stream and computation time is assumed to be unlimited. The complexity theory question is which problems remain solvable and which problems are hard in such a model, taking into account trade-offs between the amount of memory and passes.


Many graph streaming problems require $\Omega(n)$ bits of memory [33, 34] for a constant number of passes on $n$-vertex graphs. Any $p$-pass algorithm for Connectivity needs $\Omega(n/p)$ bits of memory [38]. Single pass algorithms for Connectivity or Diameter need $\Omega(n \log n)$ bits of memory on sparse graphs [50]. A 2-approximation of Diameter requires $\Omega(n^{3/2})$ bits of memory on weighted graphs [34]. A naive streaming algorithm for Connectivity or Diameter stores the entire graph, using $\mathcal{O}(m \log n) = \mathcal{O}(n^2 \log n)$ bits and a single pass. For Connectivity, union-find yields a 1-pass, $\mathcal{O}(n \log n)$ bits of memory, algorithm [43].

An intriguing aspect on Diameter and Connectivity is that some classic algorithms for them rely on breadth-first search (BFS) or depth-first search (DFS). These seem difficult to execute efficiently in a streaming setting. It was a longstanding open problem to compute a DFS tree using $o(n)$ passes and $o(m \log n)$ bits of memory. This barrier was recently broken [42], through an algorithm that uses $\mathcal{O}(n/k)$ passes and $\mathcal{O}(nk \log n)$ bits of memory, for any $k$. The situation for computing single-source shortest paths seems similar [30], although good approximations exist even on weighted graphs (see e.g. [43, 31]). We do know that DFS algorithms cannot be executed in logarithmic space [48]. In streaming, any BFS algorithm that explores $k$ layers of the BFS tree must use at least $k/2$ passes or $\Omega(n^{1+1/k}/(\log n)^{1/k})$ space [34]. Hence, much remains unexplored when it comes to graph exploration- and graph distance-related streaming problems such as BFS/DFS, Diameter, and Connectivity. In particular, most lower bounds hold for general graphs. As such, a more fine-grained view of the complexity of these problems has so far been lacking.

In this paper, we seek to obtain this fine-grained view using parameterized complexity [25]. The idea of using parameterized complexity in the streaming setting was first introduced by Fafianie and Kratsch [32] and Chitnis et al. [21]. Many problems are hard in streaming parameterized by their solution size [32, 21, 18]. Crucially, however, deciding whether a graph has a vertex cover of size $k$ has a one-pass, $\tilde{\mathcal{O}}(k^2)$-memory kernelization algorithm by Chitnis et al. [19], and a $2^k$-passes, $\tilde{\mathcal{O}}(k)$-memory direct algorithm by Chitnis and Cormode [18].

Bishnu et al. [10] then showed that knowing a vertex cover of size $k$ is useful in solving other deletion problems using $p(k)$ passes and $f(k) \log n$ memory, notably $H$-free deletion; this approach was recently expanded on by Oostveen and van Leeuwen [46]. This leads to the more general question how knowing a (small) *$H$-free modulator*, that is, a set $X$ such that $G - X$ has no induced subgraph isomorphic to $H$ (note that $H = P_2$ in Vertex Cover $[k]$[1]), would affect the complexity of streaming problems and of BFS/DFS, Diameter, and Connectivity in particular. We are not aware of any investigations in this direction.

An important consideration is the streaming model (see [38, 36, 41, 44] or the survey by McGregor [43]). In the Edge Arrival (EA) model, each edge of the graph appears once in the stream, and the edges appear in some fixed but arbitrary order. Most aforementioned results use this model. In the Vertex Arrival (VA) model the edges arrive grouped per vertex, and an edge is revealed at its endpoint that arrives latest. In the Adjacency List (AL) model the edges also arrive grouped per vertex, but each edge is present for both its endpoints. This means we see each edge twice and when a vertex arrives we immediately see all its adjacencies (rather than some subset dependent on the arrival order, as in the VA model). This model is quite strong, but as we shall see, unavoidable for our positive results. We do not consider dynamic streaming models in this paper, although they do exist.

**Our Contributions.** The main takeaway from our work is that the vertex cover number likely sits right at the frontier of parameters that are helpful in computing Diameter and Connectivity. As our main positive result, we show the following.

[1] See Section 2 for the notation.

▶ **Theorem 1.** *Given a vertex cover of size $k$,* Diameter *$[k]$ and* Connectivity *$[k]$ can be solved using $\mathcal{O}(2^k k)$ passes and $\mathcal{O}(k \log n)$ bits of space or using one pass and $\mathcal{O}(2^k + k \log n)$ bits of space, in the AL model.*

The crux to our approach is to perform a BFS in an efficient manner, using $\mathcal{O}(k)$ passes and $\mathcal{O}(k \log n)$ space. Knowledge of a vertex cover is not a restricting assumption, as one may be computed using similar memory requirements [19, 18]. An extension allows the one-pass result to work without a vertex cover being given, at the cost of increasing the memory use to $\mathcal{O}(4^k + k \log n)$ bits of space.

As a contrasting result, we observe that in the VA model, even a constant-size vertex cover does not help in computing Diameter and Connectivity. Moreover, the bound on the vertex cover seems necessary, as we can prove that any $p$-pass algorithm for Diameter requires $\Omega(n^2/p)$ bits of memory even on bipartite graphs and any $p$-pass algorithm for Connectivity requires $\Omega(n/p)$ bits of memory, both in the AL model. This indicates that both the permissive AL model and a low vertex cover number are truly needed.

In some cases, we are also able to prove that a single-pass algorithm requires $\Omega(n \log n)$ bits of memory (due to lack of space these proofs are fully deferred to the full version of the paper [47]).

More broadly, knowledge of being *$H$-free* (that is, not having a fixed graph $H$ as an induced subgraph) or having an $H$-free modulator does not help even in the AL model:

▶ **Theorem 2.** *For any fixed graph $H$ with $H \not\subseteq_i P_4$ ($H$ is not an induced subgraph of $P_4$) and $H \neq 3P_1, P_3 + P_1, P_2 + 2P_1$, any streaming algorithm for* Diameter *in the AL model that uses $p$ passes over the stream must use $\Omega(n/p)$ bits of memory on graphs $G$ even when $G$ is $H$-free.*

We note that these results hold for $H$-free graphs (without the need for a modulator). The case when $H \subseteq_i P_4$ is straightforward to solve with $\mathcal{O}(\log n)$ bits of memory, as the diameter is either 1 or 2 (an induced path of length 3 is a $P_4$). If the graph has diameter 1, it is a clique. This can be tested in a single pass by counting the number of edges.

▶ **Theorem 3.** *For any fixed graph $H$ with $H \neq P_2 + sP_1$ for $s \in \{0, 1, 2\}$ and $H \neq sP_1$ for $s \in \{1, 2, 3\}$, any streaming algorithm for* Diameter *in the AL model that uses $p$ passes over the stream must use $\Omega(n/p)$ bits of memory on graphs $G$ even when given a set $X \subseteq V(G)$ of constant size such that $G - X$ is $H$-free. If $G - X$ must be connected and $H$-free, then additionally $H \neq P_3$.*

We note that the case when $H = P_2$ or $H = P_1$ is covered by Theorem 1. Cobipartite graphs seem to be a bottleneck class. The cases when $H = 2P_1$ or when $H = P_3$ and $G - X$ must be connected lead to a surprising second positive result.

▶ **Theorem 4.** *Given a set $X$ of size $k$ such that $G - X$ is a disjoint union of $\ell$ cliques,* Diameter *$[k, \ell]$ and* Connectivity *$[k, \ell]$ can be solved using $\mathcal{O}(2^k k\ell)$ passes and $\mathcal{O}((k + \ell) \log n)$ bits of space or one pass and $\mathcal{O}(2^k \ell + (k + \ell) \log n)$ bits of space, in the AL model.*

The approach for this result is similar as for Theorem 1. Moreover, we show a complementary lower bound in the VA model, even for $\ell = 1$ and constant $k$.

Our results for Diameter are summarized in Table 1 and 2. In words, generalizing Theorem 1 using the perspective of an $H$-free modulator does not seem to lead to a positive result (Theorem 4). Instead, connectivity of the remaining graph after removing the modulator seems crucial. However, this perspective only helps for Theorem 4, while the problem remains

hard for most other $H$-free modulators and even for the seemingly simple case of a modulator to a path. While Theorem 4 would also hint at the possibility of using a modulator to a few components of small diameter, this also leads to hardness.

We emphasize that all instances of Diameter in our hardness reductions are connected graphs. Hence, the hardness of computing Diameter is separated from the hardness of computing Connectivity.

For Connectivity, we also give two broad theorems that knowledge of being $H$-free or having an $H$-free modulator does not help even in the AL model.

▶ **Theorem 5.** *For any fixed graph $H$ that is not a linear forest containing only paths of length at most 5, any streaming algorithm for Connectivity in the AL model that uses $p$ passes over the stream must use $\Omega(n/p)$ bits of memory on graphs $G$ even when $G$ is $H$-free.*

▶ **Theorem 6.** *For any fixed graph $H$ that is not a linear forest containing only paths of length at most 1, any streaming algorithm for Connectivity in the AL model that uses $p$ passes over the stream must use $\Omega(n/p)$ bits of memory on graphs $G$ even when given a set $X \subseteq V(G)$ of constant size such that $G - X$ is $H$-free.*

As a final result, we use our insights into graph exploration on graphs of bounded vertex cover to show a result on the Vertex Cover problem itself. In particular, a kernel on $2k$ vertices for Vertex Cover $[k]$ can be obtained as a stream in $\mathcal{O}(k^3)$ passes in the EA model using only $\tilde{\mathcal{O}}(k)$ bits of memory. In the AL model, the number of passes is only $\mathcal{O}(k^2)$. This kernel still may have $\mathcal{O}(k^2)$ edges, which means that saving it in memory would not give a better result than that of Chitnis et al. [19] (which uses $\tilde{\mathcal{O}}(k^2)$ bits of memory). Indeed, a better kernel seems unlikely to exist [24]. However, the important point is that storing the (partial) kernel in memory is not needed during its computation. Hence, it may be viewed as a possible first step towards a streaming algorithm for Vertex Cover $[k]$ using $\tilde{\mathcal{O}}(k)$ bits of memory and poly$(k)$ passes, which is an important open problem in the field, see [18]. Our kernel is constructed through a kernel by Buss and Goldsmith [14], and then finding a maximum matching in an auxiliary bipartite graph (following Chen et al. [16]) of bounded size through repeated DFS applications.

**Related work.** There has been substantial work on the complexity of graph-distance and reachability problems in the streaming setting. For example, Guruswami and Onak [37] showed that any $p$-pass algorithm needs $n^{1+\Omega(1/p)}/p^{\mathcal{O}(1)}$ memory when given vertices $s, t$ to test if $s, t$ are at distance at most $2p + 2$ in undirected graphs or to test $s$-$t$ reachability in directed graphs. Further work on directed $s$-$t$ reachability [6] recently led to a lower bound that any $o(\sqrt{\log n})$-pass algorithm needs $n^{2-o(1)}$ bits of memory [17]. Other recent work considers $p$-pass algorithms for $\epsilon$-property testing of connectivity [51, 39, 4], including strong memory lower bounds $n^{1-\mathcal{O}(\epsilon \cdot p)}$ on bounded-degree planar graphs [5]. Further problems in graph streaming are extensively discussed and referenced in these works; see also [3].

In the non-streaming setting, the Diameter problem can be solved in $\mathcal{O}(nm)$ time by BFS. There is a lower bound of $n^{2-\epsilon}$ for any $\epsilon > 0$ under the Strong Exponential Time Hypothesis (SETH) [49]. Parameterizations of Diameter have been studied with parameter vertex cover [13], treewidth [1, 40, 13], and other parameters [23, 8], leading to a $2^{\mathcal{O}(k)}n^{1+\epsilon}$ time algorithm on graphs of treewidth $k$ [13]. Running time $2^{o(k)}n^{2-\epsilon}$ for graphs of treewidth $k$ is not possible under SETH [1]. Subquadratic algorithms are known for various hereditary graph classes; see e.g. [15, 22, 26, 27, 28, 29, 35] and references in [22].

## 2 Preliminaries

We work on undirected, unweighted graphs. We denote a computational problem $A$ with A $[k]$, where [.] denotes the parameterization. The default parameter is solution size, if not mentioned otherwise. Diameter is to compute $\max_{s,t \in V} d(s,t)$ where $d(s,t)$ denotes the distance between $s$ and $t$. Connectivity asks to decide whether or not the graph is connected. A *twin class* consists of all vertices with the same open neighbourhood. In a graph with vertex cover size $k$, we have $\mathcal{O}(2^k)$ twin classes. For two graphs $G, H$, $G + H$ denotes their disjoint union. We also use $2G$ to denote $G + G$; $3G$ is $G + G + G$, etc. A *linear forest* is a disjoint union of paths. A path on $a$ vertices is denoted $P_a$ and has length $a - 1$.

We employ the following problem in communication complexity.

| |
|---|
| $\textsc{Disj}_n$ (Disjointness)<br>*Input:* Alice has a string $x \in \{0,1\}^n$ given by $x_1 x_2 \dots x_n$. Bob has a string $y \in \{0,1\}^n$ given by $y_1 y_2 \dots y_n$.<br>*Question:* Bob wants to check if $\exists 1 \leq i \leq n$ such that $x_i = y_i = 1$. (Formally, the answer is NO if this is the case.) |

The communication complexity necessary between Alice and Bob to solve this problem is well understood, and can be used to prove lower bounds on the memory use of streaming algorithms. This was first done by Henzinger et al. [38]. The following formulation by Bishnu et al. [9] comes in very useful.

▶ **Proposition 7** (Rephrasing of item $(ii)$ of [9], Proposition 5.6). *If we can show a reduction from $\textsc{Disj}_n$ to problem $\Pi$ in streaming model $\mathcal{M}$ such that in the reduction, Alice and Bob construct one model-$\mathcal{M}$ pass for a streaming algorithm for $\Pi$ by communicating the memory state of the algorithm only a constant number of times to each other, then any streaming algorithm working in the model $\mathcal{M}$ for $\Pi$ that uses $p$ passes requires $\Omega(n/p)$ bits of memory, for any $p \in \mathbb{N}$ [20, 11, 2].*

If we can show a reduction from Disjointness, we call a problem "hard", as it does not admit algorithms using only poly-logarithmic memory.

Any upper bound for the EA model holds for all models, and an upper bound for the VA model also holds for the AL model. On the other hand, a lower bound in the AL model holds for all models, and a lower bound for the VA model also holds for the EA model.

## 3 Upper Bounds for Diameter

We give an overview of our upper bound results for Diameter in Table 1. The memory-efficient results rely on executing a BFS on the graph, which is made possible by both the parameter and the use of the AL model. The one-pass results rely on the possibility to save the entire graph in a bounded fashion. Our upper bounds assume the deletion set related to the parameter is given, that is, it is in memory.

▶ **Lemma 8 (♣).** *In a graph with vertex cover size $k$, any simple path has length at most $2k$.*

(Further discussions and proofs for results marked with ♣ appear in the full online version of the paper [47].)

Lemma 8 is useful in that the diameter of such a graph can be at most $2k$ if the graph is connected. Our algorithm will simulate a BFS for $2k$ rounds, deciding on the distance of a vertex to all other vertices.

**Table 1** Overview of the algorithms and their complexity for DIAMETER and CONNECTIVITY. The results for the VERTEX COVER parameter are given in Theorems 10,11.

| Parameter ($k$) | Passes | Memory (bits) | Model |
|---|---|---|---|
| VERTEX COVER | $\mathcal{O}(2^k k)$ | $\mathcal{O}(k \log n)$ | AL |
| | 1 | $\mathcal{O}(2^k + k \log n)$ | AL |
| DISTANCE TO $\ell$ CLIQUES | $\mathcal{O}(2^k \ell k)$ | $\mathcal{O}((k+\ell) \log n)$ | AL |
| | 1 | $\mathcal{O}(2^k \ell + (k+\ell) \log n)$ | AL |

▶ **Lemma 9.** *Given a graph $G$ as an AL stream with a vertex cover $X$ of size $k$, we can compute the distance from a vertex $v$ to all others using $\mathcal{O}(k)$ passes and $\mathcal{O}(k \log n)$ bits of memory.*

**Proof.** We simulate a BFS originating at $v$ for at most $2k$ rounds on our graph, using a pass for each round. Contrary to a normal BFS, we only remember whether we visited the vertices in the vertex cover and their distances, to reduce memory complexity.

For every vertex $w \in X$, we save its tentative distance $d(w)$ from $v$; if this is not yet decided, this field has value $\infty$. Our claim will be that after round $i$, the value of $d(w)$ for vertices $w$ within distance $i$ from $v$ is correct. We initialize the distance of $v$ as $d(v) = 0$ (we store $d(v)$ regardless of whether $v \in X$).

Say we are in round $i \geq 1$. We execute a pass over the stream. Say we view a vertex $w \in X \cup \{v\}$ in the stream with its adjacencies. If $w$ has a distance of $d(w)$, we update the neighbours of $w$ in $X$ to have distance $d(u) = \min(d(u), d(w) + 1)$. If instead we view a vertex $w \notin X \cup \{v\}$ in the stream, we do the following. Locally save all the neighbours and look at their distances, and let $z$ be the neighbour with minimum $d(z)$ value. For every $u \in N(w)$ we update the distance as $d(u) = \min(d(z) + 2, d(u))$. This simulates the distance of a path passing through $w$ (note that this may not be the shortest path, but this may be resolved by other vertices). This completes the procedure for round $i$.

Notice that we use only $\mathcal{O}(k \log n)$ bits of memory during the procedure, and that the total number of passes is indeed $\mathcal{O}$(k) as we execute $2k$ rounds, using one pass each.

For the correctness, let us first argue the correctness of the claim *after round $i$, the value of $d(w)$ of vertices $w \in X$ within distance $i$ from $v$ is correct.* We proceed by induction, clearly the base case of 0 is correct. Now consider some vertex $w$ at distance $i$ from $v$. Consider a shortest path from $v$ to $w$. Look at the last vertex on the path before visiting $w$. If this vertex is in $X$, then by induction, this vertex has a correct distance after round $i - 1$, and so, in round $i$ this vertex will update the distance of $w$ to be $i$. If this vertex is not in $X$, then it has a neighbour with distance $i - 2$, which is correct after round $i - 2$ by induction, and so, the vertex not in $X$ will (have) update(d) the distance of $w$ to be $i$ in round $i$.

The correctness of the algorithm now follows from the claim, together with Lemma 8, and the fact that we can now output all distances using a single pass by either outputting the value of the field $d(w)$ for a vertex $w \in X$, or by looking at all neighbours of a vertex $w \notin X$ and outputting the smallest value $+1$. ◀

Related is a lower bound result by Feigenbaum et al. [34], which says that any BFS procedure that explores $k$ layers of the BFS tree must use at least $k/2$ passes or super-linear memory. This indicates that memory- and pass-efficient implementations of BFS, as in Lemma 9, are hard to come by.

We can now use Lemma 9 to construct an algorithm for finding the diameter of a graph parameterized by vertex cover, essentially by executing Lemma 9 for every twin class, which considers all options for vertices in the graph.

▶ **Theorem 10 (♣).** *Given a graph $G$ as an AL stream with vertex cover $X$ of size $k$, we can solve* Diameter *$[k]$ in $\mathcal{O}(2^k k)$ passes and $\mathcal{O}(k \log n)$ bits of memory.*

We show an alternative one-pass algorithm, by saving the graph as a representation by its twin classes, thereby completing the proof of Theorem 1.

▶ **Theorem 11 (♣).** *Given a graph $G$ as an AL stream, we can solve* Diameter *$[k]$ in one pass and $\mathcal{O}(4^k + k \log n)$ bits of memory, or correctly report that a vertex cover of size $k$ does not exist. When a vertex cover of size $k$ is given, the memory use is $\mathcal{O}(2^k + k \log n)$.*

The ideas of Theorem 10 and Theorem 11 also work for a similar setting with a few adjustments. This is the setting of Theorem 4, that our problem is parameterized by Distance to $\ell$ Cliques, where both the deletion distance $k$ and the number of remaining cliques $\ell$ are bounded. The BFS idea works here as well, as shortest paths are of bounded length, and we can save information for every vertex in the deletion set, as well as some information for every clique. There is also a concept of twin classes in such an instance, where we also distinguish which clique a vertex belongs to, and this is useful for the BFS from "every" vertex, as well as a one-pass algorithm where we save the entire graph in a compressed representation. The details of the theorems and proofs that make up Theorem 4 are given in the full version (♣). When the number of cliques is not bounded, this setting admits a lower bound, which we will see in Section 4.

## 4 Lower Bounds for Diameter

We work with reductions from $\textsc{Disj}_n$, and we construct graphs where Alice controls some of the edges, and Bob controls some of the edges, depending on their respective input of the $\textsc{Disj}_n$ problem, and some parts of the graph are fixed. The aim is to create a gap in the diameter of the graph, that is, the answer to $\textsc{Disj}_n$ is YES if and only if the diameter is above or below a certain value. The lower bound then follows from Proposition 7. Here $n$ may be the number of vertices in the graph construction, but may also be different (possibly forming a different lower bound). Our lower bounds hold for connected graphs.

We start by proving simple lower bounds for the VA model when our problem is parameterized by the vertex cover number, and when our problem is parameterized by the distance to $\ell$ cliques. This shows that we actually need the AL model to achieve the upper bounds in Section 3. The constructions are illustrated in Figure 1 and Figure 2. Generally, $a$-vertices ($b$-vertices) and their incident edges are controlled by Alice (Bob). To give an idea of the reduction technique, we describe how a VA stream is constructed by Alice and Bob, in the construction of Figure 1. First, Alice reveals the middle vertices including the vertex $c$ and the fixed edges, then reveals the vertex $a$ with the edges dependent on her input. Then the memory state of the algorithm is given to Bob who can reveal his vertex $b$ with the edges dependent on his input. Notice that this is a valid VA stream, and Alice and Bob need no information about the input of the other.

▶ **Theorem 12 (♣).** *Any streaming algorithm for* Diameter *on graphs of vertex cover number at least 3 in the VA model that uses $p$ passes over the stream requires $\Omega(n/p)$ bits of memory.*

▶ **Theorem 13 (♣).** *Any streaming algorithm for* Diameter *on graphs of distance 2 to $\ell = 1$ clique in the VA model that uses $p$ passes over the stream requires $\Omega(n/p)$ bits of memory.*

**Table 2** An overview of the lower bounds for DIAMETER, with the parameter ($k$) on the left. These results hold for connected graphs. $(\mathcal{M}, m, p)$-hard means that any algorithm using $p$ passes in model $\mathcal{M}$ (or weaker) requires $\Omega(m)$ bits of memory. FVS stand for Feedback Vertex Set number, FEN for Feedback Edge Set number. Most proofs of the results in this table are deferred to the full paper (♣).

| Parameter ($k$) / Graph class | Size | Bound |
|---|---|---|
| General and Bipartite Graphs | | $(\text{AL}, n^2/p, p)$-hard |
| VERTEX COVER | $\geq 3$ | $(\text{VA}, n/p, p)$-hard |
| DISTANCE TO $\ell$ CLIQUES | $k \geq 2, \ell \geq 1$ | $(\text{VA}, n/p, p)$-hard |
| FVS, FES | $\geq 0$ | $(\text{AL}, n/p, p)$-hard |
| | $\geq 0$ | $(\text{AL}, n \log n, 1)$-hard |
| DISTANCE TO MATCHING | $\geq 3$ | $(\text{AL}, n/p, p)$-hard |
| DISTANCE TO PATH | $\geq 2$ | $(\text{AL}, n/p, p)$-hard |
| | $\geq 2$ | $(\text{AL}, n \log n, 1)$-hard |
| DISTANCE TO DEPTH $\ell$ TREE | $k \geq 3, \ell \geq 2$ | $(\text{AL}, n/p, p)$-hard |
| | $k \geq 0, \ell \geq 5$ | $(\text{AL}, n/p, p)$-hard |
| | $k \geq 0, \ell \geq 7$ | $(\text{AL}, n \log n, 1)$-hard |
| DIST. TO $\ell$ COMPS. OF DIAM. $x$ | $k, x \geq 2$ | $(\text{AL}, n/p, p)$-hard |
| DOMINATION NUMBER | $\geq 3$ | $(\text{AL}, n/p, p)$-hard |
| MAXIMUM DEGREE | $\geq 3$ | $(\text{AL}, n/p, p)$-hard |
| | $\geq 3$ | $(\text{AL}, n \log n, 1)$-hard |
| Split graphs | | $(\text{AL}, n/p, p)$-hard |

The lower bounds in Figure 1 and Figure 2 do not work for the AL model because there are vertices that may or may not be adjacent to both $a$ and $b$, so neither Alice nor Bob can produce the adjacency list of such a vertex alone. For the "Simple VA" construction, we can "fix" this by extending these vertices to edges but this is destructive to the small vertex cover number of the construction. This "fixed" construction is fully deferred to the full version of the paper (♣). It should be clear that AL reductions require care: no vertex may be incident to variable edges of both Alice and Bob.

The following theorem is a combination of several constructions, implying AL hardness on trees, splits graphs, and for many deletion-distance-to-$x$ parameters. An overview of all hardness results for DIAMETER is given in Table 2. See Figures 3, 4, 5 for illustrations of the constructions and an idea of the proof.

▶ **Theorem 14 (♣).** *Any streaming algorithm for DIAMETER that works on a family of graphs that includes the "Windmill", "Diamond", or "Split" construction in the AL model using $p$ passes over the stream requires $\Omega(n/p)$ bits of memory.*

We can now prove Theorem 2 and Theorem 3; full proofs are deferred (♣). Intuitively, if $H$ contains a cycle or a vertex of degree 3, a modification of "Windmill" is $H$-free; if $H$ is a linear forest, a modification of "Split" is (almost) $H$-free.

We can also prove a quadratic bound for general graphs; see Figure 6 for the construction.

▶ **Theorem 15 (♣).** *Any streaming algorithm for DIAMETER on general (dense) graphs in the AL model using $p$ passes over the stream requires $\Omega(n^2/p)$ bits of memory.*

Splitting up $u_A$ and $u_B$ into two vertices each, and making the tails from $t'_i$ to $t_i$ at least three edges longer for each $i$ makes the lower bound work for bipartite graphs (♣).

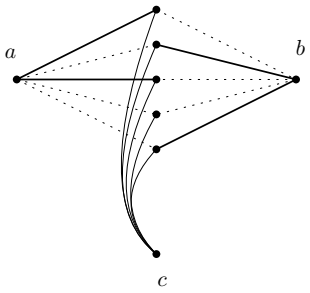


**Figure 1** VA lower bound for diameter with vertex cover size 3, called "Simple VA". The vertices in the middle are indexed $1, \ldots, n$. An edge incident to $a$ ($b$) is present when the entry of Alice (Bob) at the corresponding index is 1. The vertex $c$ ensures the graph is connected.

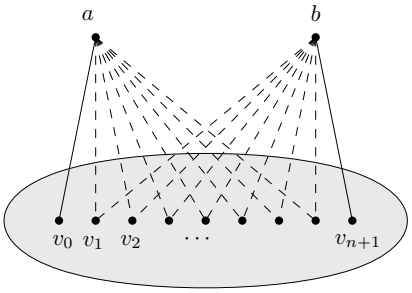


**Figure 2** VA lower bound for diameter with distance 2 to 1 clique, called "Clique VA". A dashed edge is present when the entry at the corresponding index is 1. The vertices inside the grey area form a clique. Hence, deletion distance to a clique is 2 (remove $a$ and $b$).

## 5 Connectivity

In this section, we show results for Connectivity. Connectivity is an easier problem than Diameter, that is, solving Diameter solves Connectivity as well, but not the other way around. Hence, lower bounds in this section also imply lower bounds for Diameter (in non-connected graphs). In general graphs, a single pass, $\mathcal{O}(n \log n)$ bits of memory algorithm exists by maintaining connected components in a Disjoint Set data structure [43], which is optimal in general graphs [50]. The interesting part about Connectivity is that some graph classes admit fairly trivial algorithms by a counting argument. For example, if the input is a forest, we can decide on Connectivity by counting the number of edges, which is a 1-pass, $\mathcal{O}(\log n)$ bits of memory, algorithm. An overview of the results in this section is given in Table 3. The following upper bounds follow from applications of the Disjoint Set data structure.

▶ **Observation 16 (♣).** *Given a graph $G$ as an AL stream with vertex cover number $k$, we can solve Connectivity $[k]$ in 1 pass and $\mathcal{O}(k \log n)$ bits of memory.*

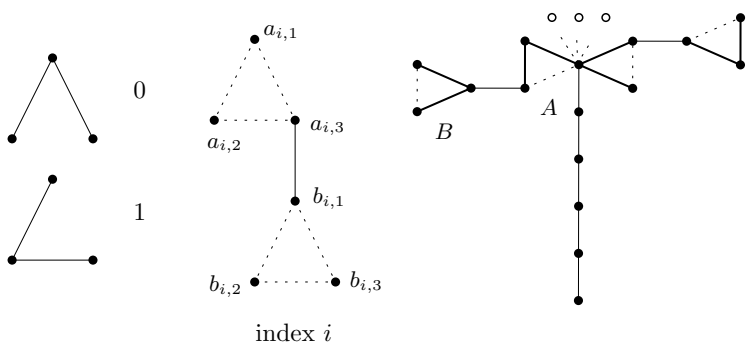


**Figure 3** AL lower bound for diameter consisting of a tree, called "Windmill". The difference in an entry 1 or 0 is shown on the left. The gadget for index $i$ combines a 0/1-gadget for Alice and a 0/1-gadget for Bob. It makes two 1 entries at this index a path of length 5, and a tree structure of depth at most 4 otherwise. These $n$ gadgets are then identified at $a_{i,1}$ and a tail is added.

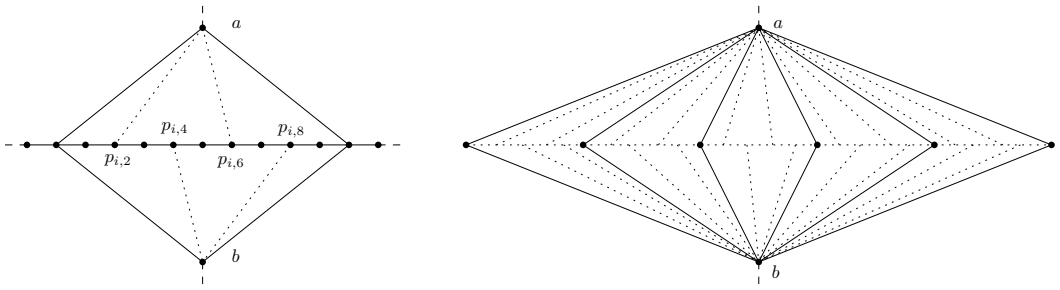


**Figure 4** AL lower bound for diameter consisting of a path and 2 vertices, called "Diamond". Note that $a$ is connected to $b$ with an edge (indicated with a dashed line here). On the left the gadget for a single index $i$ is shown, where the dotted edges are present when the entry at index $i$ is 0 (for Alice incident on $a$, for Bob incident on $b$). On the right, the construction is sketched in full.

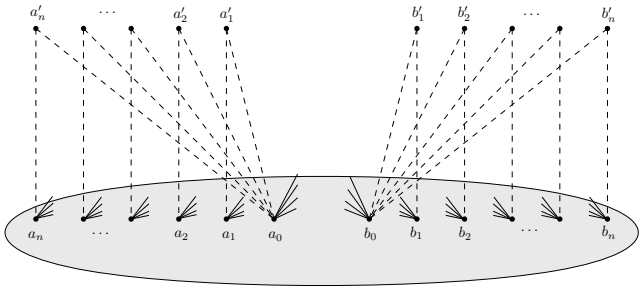


**Figure 5** AL lower bound for diameter on split graphs, called "Split". Depending on the input, some $a'_i$ either has an edge to $a_0$ or $a_i$ when $x_i = 0$ or 1. The same holds for $b'_i$ with $y_i$. The grey area forms a clique, and each $a_i$ is connected to all $b'_j$ where $i \neq j$, and the same holds for $b_i$ and $a'_j$.

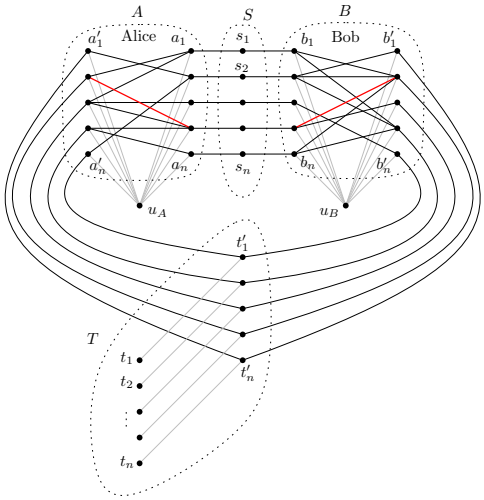


**Figure 6** AL lower bound for diameter where Alice and Bob have $n^2$ bits but the graph has $\mathcal{O}(n)$ vertices. The bits are seen as an adjacency matrix in the bipartite graphs $A$ and $B$, identically: the red edge $a'_i$ to $a_j$ in $A$ is the same index as the red edge $b_j$ to $b'_i$ in $B$. Edges are present when the entry is a 0. Then, each $s_i, t_j$ pair can discern whether or not at least one of the edges $a_i$ to $a'_j$ or $b_i$ to $b'_j$ is present, hence deciding whether or not both Alice and Bob have a 1 at that entry.

**Table 3** Overview of the results for Connectivity. All hardness results listed here are through reductions from Disjointness. $(\mathcal{M}, m, p)$-hard means that any algorithm using $p$ passes in model $\mathcal{M}$ (or weaker) requires $\Omega(m)$ bits of memory. $(\mathcal{M}, m, p)$-str. means that there is an algorithm that uses $p$ passes in model $\mathcal{M}$ (or stronger) using $\mathcal{O}(m)$ bits of memory. FVS stand for Feedback Vertex Set number, FEN for Feedback Edge Set number. We state most upper bounds only as observations, and most proofs of the results in this table are deferred to the full paper (♣).

| Parameter ($k$) / Graph class | Size | Bound |
|---|---|---|
| General Graphs | | $(\text{EA}, n \log n, 1)$-str. via Disjoint Set [43] |
| | | $(\text{EA}, n \log n, 1)$-hard by Sun and Woodruff [50] |
| Vertex Cover Number | $\geq 0$ | $(\text{AL}, k \log n, 1)$-str. Disjoint Set on Vertex Cover |
| | $\geq 2$ | $(\text{VA}, n/p, p)$-hard by Henzinger et al. [38] |
| Distance to $\ell$ cliques | $\geq 0$ | $(\text{AL}, (k+\ell) \log n, 1)$-str. via Disjoint Set |
| FVS | $= 0$ | $(\text{EA}, \log n, 1)$-str. by counting. |
| | $\geq 1$ | $(\text{AL}, n/p, p)$-hard |
| FES | $\geq 0$ | $(\text{EA}, \log n, 1)$-str. by counting. |
| Distance to matching | $\geq 2$ | $(\text{AL}, n/p, p)$-hard |
| Distance to path | $\geq 0$ | $(\text{EA}, k \log n, 1)$-str. by checking connection to path |
| Distance to depth $\ell$ tree | $\geq 0$ | $(\text{EA}, k \log n, 1)$-str. by checking connection to tree |
| Domination Number | $\geq 2$ | $(\text{AL}, n/p, p)$-hard |
| Distance to Chordal | $\geq 1$ | $(\text{AL}, n/p, p)$-hard |
| Maximum Degree | $\geq 2$ | $(\text{AL}, n/p, p)$-hard, $(\text{AL}, n \log n, 1)$-hard |
| Bipartite Graphs | | $(\text{AL}, n/p, p)$-hard, $(\text{AL}, n \log n, 1)$-hard |
| Interval Graphs | | $(\text{VA}, n/p, p)$-hard |
| Split graphs | | $(\text{EA}, n/p, p)$-str. by finding degree 0 vertex |
| | | $(\text{VA}, n/p, p)$-hard |

▶ **Observation 17 (♣).** *Given a graph $G$ as an AL stream with a deletion set $X$ of size $k$ to $\ell$ cliques, we can solve Connectivity $[k, \ell]$ in 1 pass and $\mathcal{O}((k+\ell) \log n)$ bits of memory.*

We fully defer a simple lower bound construction for the AL model to the full version (♣).

An interesting lower bound is for a unique case: graphs of maximum degree 2. We mentioned that for a forest we have a simple counting algorithm for Connectivity, so the hardness must be for some graph which consists of one or more cycles. Although we show (♣) that Connectivity is hard for graphs with a Feedback Vertex Set of constant size, we now show that in the specific case of maximum degree 2-graphs, the problem is still hard, see Figure 7 for an illustration of the construction. We note that this reduction is similar to the problem tackled by Verbin and Yu [51] and Assadi et al. [4], but our result is slightly stronger in this setting, as it concerns a distinction between 1 or 2 disjoint cycles.

▶ **Theorem 18 (♣).** *Any streaming algorithm for Connectivity that works on a family of graphs that includes graphs of maximum degree 2 in the AL model using $p$ passes over the stream requires $\Omega(n/p)$ bits of memory.*

We note that we can make the result of Theorem 18 hold for bipartite graphs of maximum degree 2 by subdividing every edge, making the graph odd cycle-free, and thus bipartite. The proofs of Theorems 5 and 6 follow, and are deferred the the full version (♣).

Interval and split graphs are hard in the VA model, see Figures 8 and 9.

▶ **Theorem 19 (♣).** *Any streaming algorithm for Connectivity that works on a family of graphs that includes interval graphs or split graphs in the VA model using $p$ passes over the stream requires $\Omega(n/p)$ bits of memory.*

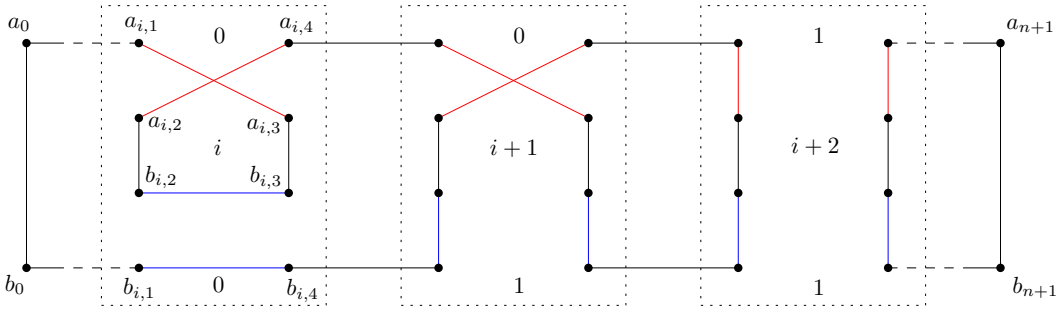


**Figure 7** AL lower bound for connectivity, called "Cycles". The graph consists of one or multiple cycles depending on the output of $\textsc{Disj}_n$. The black edges are always present. The red (blue) edges are controlled by Alice (Bob) and are in a crossing (horizontal) or vertical configuration depending on whether the $i$-th entry of Alice (Bob) is 0 or 1.

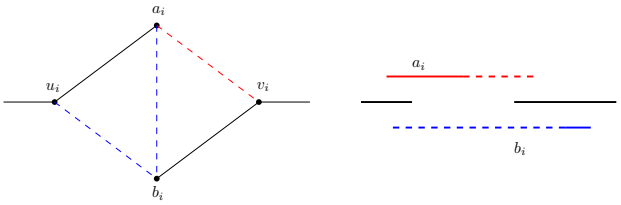


**Figure 8** VA lower bound for connectivity on interval graphs, called "Interval". We see the gadget for index $i$, where the dotted lines are present when the corresponding value is 0. The black edges are always present, and the red (blue) edges correspond to the input of Alice (Bob). The $n$ gadgets are placed consecutively.

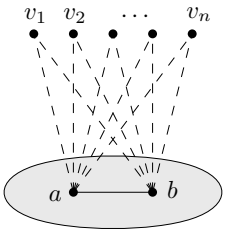


**Figure 9** VA lower bound for connectivity on split graphs, called "Split-Conn". The dashed edges towards $v_i$ are present when there is a 0 at index $i$.

For split graphs, in any model, Connectivity admits a one-pass, $\mathcal{O}(n)$ bits of memory algorithm by counting if there is a vertex of degree 0 (and so also for any $p$ a $p$-pass algorithm using $\mathcal{O}(n/p)$ bits by splitting up the work in $p$ parts)[2]. If there can be no isolated vertices, then a split graph is always connected.

## 6 Vertex Cover kernelization

We now show how our insights into parameterized, streaming graph exploration can aid in producing a new kernelization algorithm for Vertex Cover $[k]$[3]. The basis for our result is a well-known kernel for the Vertex Cover $[k]$ problem of Buss and Goldsmith [14], consisting of $\mathcal{O}(k^2)$ edges. Constructing this kernel is simple: find all vertices with degree bigger than $k$, and remove them from the graph, and decrease the parameter with the number of vertices removed, say to $k'$. Then, there is no solution if there are more than $k \cdot k'$ edges. Therefore, we have a kernel consisting of $\mathcal{O}(k^2)$ edges. We are able to achieve this same kernel in the AL model, as counting the degree of a vertex is possible in this model. Interestingly, we do not require $\tilde{\mathcal{O}}(k^2)$ bits of memory to produce a stream corresponding to the kernel of $\mathcal{O}(k^2)$ edges. This result is also possible in the EA model, by allowing vertices up to degree $2k$.

[2] This assumes the vertices are labelled $1 \ldots n$ and do not have arbitrary labels.
[3] This section is based on the master thesis "Parameterized Algorithms in a Streaming Setting" by the first author.

▶ **Theorem 20 (♣).** *Given a graph $G$ as an AL stream, we can make an AL stream corresponding to an $\mathcal{O}(k^2)$-edge kernel for the* Vertex Cover *$[k]$ problem using two passes and $\tilde{\mathcal{O}}(k)$ bits of memory. When we work with an EA stream, we can make an EA stream corresponding to an $\mathcal{O}(k^2)$-edge kernel using four passes and $\tilde{\mathcal{O}}(k)$ bits of memory.*

Next, we show how to use Theorem 20 to produce a kernel of even smaller size, using only $\tilde{\mathcal{O}}(k)$ bits of memory. This requires Theorem 20 to convert the original graph stream into the kernel input for the next theorem, which only increases the number of passes by a factor 2 or 4 (we have to apply Theorem 20 every time the other procedure uses a pass).

Interestingly, Chen et al. [16] show a way to convert the kernel of Buss and Goldsmith into a $2k$-vertex kernel for Vertex Cover $[k]$, using a theorem by Nemhauser and Trotter [45]. We will adapt this method in the streaming setting. The kernel conversion is done by converting the $\mathcal{O}(k^2)$ edges kernel into a bipartite graph (two copies of all vertices $V, V'$, and an edge $(x, y)$ translates to the edges $(x, y')$, $(x', y)$), in which we find a minimum vertex cover using a maximum matching (see for example [12, Page 74, Theorem 5.3]). The minimum vertex cover we find gives us the sets stated in the theorem by Nemhauser and Trotter [45], as indicated by the constructive proof of the same theorem by Bar-Yehuda and Even [7]. Lastly, we use these sets to give the $2k$ kernel in the streaming setting as indicated by Chen et al. [16]. This also works for the EA model, because we only require the input kernel to consist of $\mathcal{O}(k^2)$ edges, not that it specifically is the kernel by Buss and Goldsmith.

▶ **Lemma 21 (♣).** *Given a graph $G$ as a stream in model AL or EA, we can produce a stream in the same model corresponding to the Phase 1 bipartite graph of [7, Algorithm NT] using two passes and $\tilde{\mathcal{O}}(1)$ bits of memory.*

By making some observations on the conversions by Chen et al. [16], we can conclude that the maximum matching we need to find in the bipartite graph consists of at most $4k = \mathcal{O}(k)$ edges, and otherwise we can return NO. For more details, see ♣. To find the maximum matching we execute a DFS procedure, which can be done with surprising efficiency in this restricted bipartite setting.

▶ **Theorem 22 (♣).** *Given a bipartite graph $B$ as an AL stream with $\mathcal{O}(k)$ vertices, we can find a maximum matching of size at most $\mathcal{O}(k)$ using $\mathcal{O}(k^2)$ passes and $\tilde{\mathcal{O}}(k)$ bits of memory. For the EA model this can be done in $\mathcal{O}(k^3)$ passes.*

Using this maximum matching, we can find a vertex cover kernel of size $2k$. The final result is as follows, which consists of putting the original stream through each step for every time we require a pass, i.e. the number of passes of each of the parts of this theorem combine in a multiplicative fashion.

▶ **Theorem 23 (♣).** *Given a graph $G$ as an AL stream, we can produce a kernel of size $2k$ for the* Vertex Cover *$[k]$ problem using $\mathcal{O}(k^2)$ passes and $\tilde{\mathcal{O}}(k)$ bits of memory. In the EA model, this procedure takes $\mathcal{O}(k^3)$ passes.*

## 7 Conclusion

We studied the complexity of Diameter and Connectivity in the streaming model, from a parameterized point of view. In particular, we considered the viewpoint of an $H$-free modulator, showing that a vertex cover or a modulator to the disjoint union of $\ell$ cliques effectively forms the frontier of memory- and pass-efficient streaming algorithms. Both problems remain hard for almost all other $H$-free modulators of constant size (often even of

size 0). We believe that this forms an interesting starting point for further investigations into which other graph classes or parameters might be useful when computing Diameter and Connectivity in the streaming model.

On the basis of our work, we propose four concrete open questions:

- What is the streaming complexity of computing Distance to $\ell$ Cliques? On the converse of Vertex Cover $[k]$, we are not aware of any algorithms to compute this parameter, even though it is helpful in computing Diameter and Connectivity.
- Are there algorithms or lower bounds for Diameter or Connectivity in the AL model for interval graphs?
- Assuming isolated vertices are allowed in the graph, can we solve Connectivity in the AL model on split graphs using $O(\log n)$ bits of memory?
- Is there a streaming algorithm for Vertex Cover $[k]$ using $\mathcal{O}(\text{poly}(k))$ passes and $\mathcal{O}(\text{poly}(k, \log n))$ bits of memory, or can it be shown that one cannot exist? This result would be relevant in combination with our kernel.

## References


1 Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proc. SODA 2016*, pages 377–391. SIAM, 2016. doi:10.1137/1.9781611974331.ch28.

2 Deepak Agarwal, Andrew McGregor, Jeff M. Phillips, Suresh Venkatasubramanian, and Zhengyuan Zhu. Spatial scan statistics: approximations and performance study. In *Proc. SIGKDD 2006*, pages 24–33. ACM, 2006. doi:10.1145/1150402.1150410.

3 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Polynomial pass lower bounds for graph streaming algorithms. *CoRR*, abs/1904.04720, 2019. arXiv:1904.04720.

4 Sepehr Assadi, Gillat Kol, Raghuvansh R. Saxena, and Huacheng Yu. Multi-pass graph streaming lower bounds for cycle counting, max-cut, matching size, and other problems. In *Proc. FOCS 2020*, pages 354–364. IEEE, 2020. doi:10.1109/FOCS46700.2020.00041.

5 Sepehr Assadi and Vishvajeet N. Graph streaming lower bounds for parameter estimation and property testing via a streaming XOR lemma. In *Proc. STOC 2021*, pages 612–625. ACM, 2021. doi:10.1145/3406325.3451110.

6 Sepehr Assadi and Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In *Proc. FOCS 2020*, pages 342–353. IEEE, 2020. doi:10.1109/FOCS46700.2020.00040.

7 Reuven Bar-Yehuda and Shimon Even. A local-ratio theorem for approximating the weighted vertex cover problem. In *Proc. WG 1983*, pages 17–28. Universitätsverlag Rudolf Trauner, Linz, 1983. URL: http://www.gbv.de/dms/tib-ub-hannover/022054669.pdf.

8 Matthias Bentert and André Nichterlein. Parameterized complexity of diameter. In *Proc. CIAC 2019*, volume 11485 of *LNCS*, pages 50–61. Springer, 2019. doi:10.1007/978-3-030-17402-6_5.

9 Arijit Bishnu, Arijit Ghosh, Sudeshna Kolay, Gopinath Mishra, and Saket Saurabh. Fixed-parameter tractability of graph deletion problems over data streams. *CoRR*, abs/1906.05458, 2019. arXiv:1906.05458.

10 Arijit Bishnu, Arijit Ghosh, Sudeshna Kolay, Gopinath Mishra, and Saket Saurabh. Fixed parameter tractability of graph deletion problems over data streams. In *Proc. COCOON 2020*, volume 12273 of *LNCS*, pages 652–663. Springer, 2020. doi:10.1007/978-3-030-58150-3_53.

11 Arijit Bishnu, Arijit Ghosh, Gopinath Mishra, and Sandeep Sen. On the streaming complexity of fundamental geometric problems. *CoRR*, abs/1803.06875, 2018. arXiv:1803.06875.

12 J. Adrian Bondy and Uppaluri S. R. Murty. *Graph Theory with Applications*. Macmillan Education UK, 1976. doi:10.1007/978-1-349-03521-2.

**13** Karl Bringmann, Thore Husfeldt, and Måns Magnusson. Multivariate analysis of orthogonal range searching and graph distances. *Algorithmica*, 82(8):2292–2315, 2020. doi:10.1007/s00453-020-00680-z.

**14** Jonathan F. Buss and Judy Goldsmith. Nondeterminism within P. *SIAM J. Comput.*, 22(3):560–572, 1993. doi:10.1137/0222038.

**15** Sergio Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. *ACM Trans. Algorithms*, 15(2):21:1–21:38, 2019. doi:10.1145/3218821.

**16** Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: Further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001. doi:10.1006/jagm.2001.1186.

**17** 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 *Proc. STOC 2021*, pages 570–583. ACM, 2021. doi:10.1145/3406325.3451038.

**18** Rajesh Chitnis and Graham Cormode. Towards a theory of parameterized streaming algorithms. In *Proc. IPEC 2019*, volume 148 of *LIPIcs*, pages 7:1–7:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.IPEC.2019.7.

**19** Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, Mohammad Taghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *Proc. SODA 2016*, pages 1326–1344. SIAM, 2016. doi:10.1137/1.9781611974331.ch92.

**20** Rajesh Hemant Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Brief announcement: New streaming algorithms for parameterized maximal matching & beyond. In *Proc. SPAA 2015*, pages 56–58. ACM, 2015. doi:10.1145/2755573.2755618.

**21** Rajesh Hemant Chitnis, Graham Cormode, Mohammad Taghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *Proc. SODA 2015*, pages 1234–1251. SIAM, 2015. doi:10.1137/1.9781611973730.82.

**22** Derek G. Corneil, Feodor F. Dragan, Michel Habib, and Christophe Paul. Diameter determination on restricted graph families. *Discret. Appl. Math.*, 113(2-3):143–166, 2001. doi:10.1016/S0166-218X(00)00281-X.

**23** David Coudert, Guillaume Ducoffe, and Alexandru Popa. Fully polynomial FPT algorithms for some classes of bounded clique-width graphs. *ACM Trans. Algorithms*, 15(3):33:1–33:57, 2019. doi:10.1145/3310228.

**24** Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *J. ACM*, 61(4):23:1–23:27, 2014. doi:10.1145/2629620.

**25** Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999. doi:10.1007/978-1-4612-0515-9.

**26** Guillaume Ducoffe. Beyond helly graphs: The diameter problem on absolute retracts. In *Proc. WG 2021*, volume 12911 of *LNCS*, pages 321–335. Springer, 2021. doi:10.1007/978-3-030-86838-3_25.

**27** Guillaume Ducoffe and Feodor F. Dragan. A story of diameter, radius, and (almost) helly property. *Networks*, 77(3):435–453, 2021. doi:10.1002/net.21998.

**28** Guillaume Ducoffe, Michel Habib, and Laurent Viennot. Fast diameter computation within split graphs. In *Proc. COCOA 2019*, volume 11949 of *LNCS*, pages 155–167. Springer, 2019. doi:10.1007/978-3-030-36412-0_13.

**29** Guillaume Ducoffe, Michel Habib, and Laurent Viennot. Diameter computation on $H$-minor free graphs and graphs of bounded (distance) vc-dimension. In *Proc. SODA 2020*, pages 1905–1922. SIAM, 2020. doi:10.1137/1.9781611975994.117.

**30** Michael Elkin. Distributed exact shortest paths in sublinear time. *J. ACM*, 67(3):15:1–15:36, 2020. doi:10.1145/3387161.

**31** Michael Elkin and Chhaya Trehan. $(1+\epsilon)$-approximate shortest paths in dynamic streams. *CoRR*, abs/2107.13309, 2021. arXiv:2107.13309.

**32** Stefan Fafianie and Stefan Kratsch. Streaming kernelization. In *Proc. MFCS 2014*, volume 8635 of *LNCS*, pages 275–286. Springer, 2014.

**33** 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. doi:10.1016/j.tcs.2005.09.013.

**34** Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the data-stream model. *SIAM J. Comput.*, 38(5):1709–1727, 2008. doi:10.1137/070683155.

**35** Pawel Gawrychowski, Haim Kaplan, Shay Mozes, Micha Sharir, and Oren Weimann. Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\tilde{o}(n^{5/3})$ time. *SIAM J. Comput.*, 50(2):509–554, 2021. doi:10.1137/18M1193402.

**36** Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In Yuval Rabani, editor, *Proc. SODA 2012*, pages 468–485. SIAM, 2012. doi:10.1137/1.9781611973099.41.

**37** Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. *Algorithmica*, 76(3):654–683, 2016. doi:10.1007/s00453-016-0138-7.

**38** Monika Rauch Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. Computing on data streams. In *Proc. DIMACS 1998*, volume 50 of *DIMACS*, pages 107–118. DIMACS/AMS, 1998. doi:10.1090/dimacs/050/05.

**39** Zengfeng Huang and Pan Peng. Dynamic graph stream algorithms in o(n) space. *Algorithmica*, 81(5):1965–1987, 2019. doi:10.1007/s00453-018-0520-8.

**40** Thore Husfeldt. Computing graph distances parameterized by treewidth and diameter. In *Proc. IPEC 2016*, volume 63 of *LIPIcs*, pages 16:1–16:11. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.IPEC.2016.16.

**41** Michael Kapralov. Better bounds for matchings in the streaming model. In Sanjeev Khanna, editor, *Proc. SODA 2013*, pages 1679–1697. SIAM, 2013. doi:10.1137/1.9781611973105.121.

**42** Shahbaz Khan and Shashank K. Mehta. Depth first search in the semi-streaming model. In Rolf Niedermeier and Christophe Paul, editors, *Proc. STACS 2019*, volume 126 of *LIPIcs*, pages 42:1–42:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.STACS.2019.42.

**43** Andrew McGregor. Graph stream algorithms: a survey. *SIGMOD Rec.*, 43(1):9–20, 2014. doi:10.1145/2627692.2627694.

**44** Andrew McGregor, Sofya Vorotnikova, and Hoa T. Vu. Better algorithms for counting triangles in data streams. In Tova Milo and Wang-Chiew Tan, editors, *Proc. PODS 2016*, pages 401–411. ACM, 2016. doi:10.1145/2902251.2902283.

**45** George L. Nemhauser and Leslie E. Trotter Jr. Vertex packings: Structural properties and algorithms. *Math. Program.*, 8(1):232–248, 1975. doi:10.1007/BF01580444.

**46** Jelle J. Oostveen and Erik Jan van Leeuwen. Streaming deletion problems parameterized by vertex cover. In *Proc. FCT 2021*, volume 12867 of *LNCS*, pages 413–426. Springer, 2021. doi:10.1007/978-3-030-86593-1_29.

**47** Jelle J. Oostveen and Erik Jan van Leeuwen. Parameterized complexity of streaming diameter and connectivity problems. *CoRR*, abs/2207.04872, 2022. doi:10.48550/arXiv.2207.04872.

**48** John H. Reif. Depth-first search is inherently sequential. *Inf. Process. Lett.*, 20(5):229–234, 1985. doi:10.1016/0020-0190(85)90024-9.

**49** Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proc. STOC 2013*, pages 515–524. ACM, 2013. doi:10.1145/2488608.2488673.

**50** Xiaoming Sun and David P. Woodruff. Tight bounds for graph problems in insertion streams. In *Proc. APPROX/RANDOM 2015*, volume 40 of *LIPIcs*, pages 435–448. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.435.

**51** Elad Verbin and Wei Yu. The streaming complexity of cycle counting, sorting by reversals, and other problems. In Dana Randall, editor, *Proc. SODA 2011*, pages 11–25. SIAM, 2011. doi:10.1137/1.9781611973082.2.