Streaming Deletion Problems Parameterized by Vertex Cover

Jelle J. Oostveen(✉) and Erik Jan van Leeuwen

Department of Information and Computing Sciences,
Utrecht University, Utrecht, The Netherlands
{j.j.oostveen,e.j.vanleeuwen}@uu.nl

Abstract. Streaming is a model where an input graph is provided one edge at a time, instead of being able to inspect it at will. In this work, we take a parameterized approach by assuming a vertex cover of the graph is given, building on work of Bishnu et al. [COCOON 2020]. We show the further potency of combining this parameter with the Adjacency List streaming model to obtain results for vertex deletion problems. This includes kernels, parameterized algorithms, and lower bounds for the problems of Π-free Deletion, H-free Deletion, and the more specific forms of Cluster Vertex Deletion and Odd Cycle Transversal. We focus on the complexity in terms of the number of passes over the input stream, and the memory used. This leads to a pass/memory trade-off, where a different algorithm might be favourable depending on the context and instance. We also discuss implications for parameterized complexity in the non-streaming setting.

1 Introduction

Streaming is an algorithmic paradigm to deal with data sets that are too large to fit into main memory [22]. Instead, elements of the data set are inspected in a fixed order[1] and aggregate data is maintained in a small amount of memory (much smaller than the total size of the data set). It is possible to make multiple passes over the data set. The goal is to design algorithms that analyze the data set while minimizing the combination of the number of passes and the required memory. We note that computation time is not measured in this paradigm. Streaming has proved very successful and is extensively studied in many diverse contexts [27,29]. In this work, we focus on the case where the data sets are graphs and the streamed elements are the edges of the graph.

[1] We consider insertion-only streams throughout this paper.

This work is based on the master thesis "Parameterized Algorithms in a Streaming Setting" by the first author. This work is partially supported by the NWO grant OCENW.KLEIN.114 (PACAN).

E. Bampis and A. Pagourtzis (Eds.): FCT 2021, LNCS 12867, pp. 413–426, 2021.
https://doi.org/10.1007/978-3-030-86593-1_29

A significant body of work on graph streaming works in the semi-streaming model, where $\tilde{\mathcal{O}}(n)$ memory[2] is allowed, with the aim of limiting the number of necessary passes to one or two. This memory requirement might still be too much for the largest of networks. Unfortunately, many basic problems in graphs require $\Omega(n)$ or even worse space [18,19] to compute in a constant number of passes. Therefore, Fafianie and Kratsch [17] and Chitnis et al. [13] introduced concepts and analysis from parameterized complexity [16] to the streaming paradigm. For example, it can be decided whether a graph has a vertex cover of size at most K using one pass and $\tilde{\mathcal{O}}(K^2)$ space, which is optimal. This led to various further works [4,6,11] and the first systematic study by Chitnis and Cormode [10].

Our work continues this line of research and follows up on recent work by Bishnu et al. [5,6][3]. They made two important conceptual contributions. First, they analyzed the complexity of parameterized streaming algorithms in three models that prescribe the order in which the edges arrive in the stream and that are commonly studied in the literature [6,14,27,28]. The *Edge Arrival* (EA) model prescribes some permutation of all the edges of the graph. The *Vertex Arrival* (VA) requires that the edges appear per vertex: if we have seen the vertices $V' \subseteq V$ already, and the next vertex is w, then the stream contains the edges between w and the vertices in V'. Finally, the *Adjacency List* (AL) gives the most information, as it requires the edges to arrive per vertex, but when vertex v appears in the stream, we also see all edges incident to v. This means we effectively see every edge twice in a single pass, once for both of its endpoints.

The second and more important contribution of Bishnu et al. [5] was to study the size K of a vertex cover in the graph as a parameter. This has been broadly studied in parameterized complexity (see e.g. the PhD thesis of Jansen [25]). They showed that the very general $\mathcal{F}$-Subgraph Deletion and $\mathcal{F}$-Minor Deletion problems all admit one pass, $\tilde{\mathcal{O}}(\Delta(\mathcal{F}) \cdot K^{\Delta(\mathcal{F})+1})$ space streaming algorithms in the AL model, by computing small kernels to which then a straightforward exhaustive algorithm is applied. On the other hand, such generic streaming algorithms are not possible in the EA and VA models, as then (super-) linear lower bounds exist even if the size of a smallest vertex cover is constant [5].

We focus on the induced subgraph version of the vertex deletion problem, parameterized by the size of a vertex cover. Here, Π is a collection of graphs.

Π-free Deletion [VC] *Input:* A graph G with a vertex cover X, and an integer $\ell \geq 1$. *Parameter:* The size $K := \lvert X \rvert$ of a vertex cover. *Question:* Is there a set $S \subseteq V(G)$ of size at most ℓ such that $G[V(G) \setminus S]$ does not contain a graph in Π as an induced subgraph?

[2] Throughout this paper, memory is measured in bits. The $\tilde{\mathcal{O}}$ notation hides factors polylogarithmic in n. Note that $\mathcal{O}(\log n)$ bits is the space required to store (the identifier of) a single vertex or edge.

[3] As the Arxiv version contains more results, we refer to this version from here on.

To avoid triviality[4], we assume every graph in Π is edgeless or $K \geq \ell$. We assume the vertex cover is given as input; if only the size is given, we can use one pass and $\tilde{\mathcal{O}}(K^2)$ space or 2^K passes and $\tilde{\mathcal{O}}(K)$ space to obtain it [10,13] (this does not meaningfully impact our results). The unparameterized version of this problem is well known to be NP-hard [26] for any nontrivial and hereditary property Π. It has also been well studied in the parameterized setting (see e.g. [9, 20,31]). When parameterized by the vertex cover number, it has been studied from the perspective of kernelization: while a polynomial kernel cannot exist in general [8,21], polynomial kernels exist for broad classes of families Π [21,24]. As far as we are aware, parameterized algorithms for this parameterization have not been explicitly studied.

In the streaming setting, Chitnis et al. [11] showed for the unparameterized version of this problem in the EA model that any p-pass algorithm needs $\Omega(n/p)$ space if Π satisfies a mild technical constraint. For some Π-free Deletion [VC] problems, the results by Bishnu et al. [5] imply single-pass, poly(K) space streaming algorithms (through their kernel for $\mathcal{F}$-Subgraph Deletion [VC]) in the AL model and lower bounds in the EA/VA model. They also provide an explicit kernel for Cluster Vertex Deletion [VC] in the AL/EA/VA models. However, this still leaves the streaming complexity of many cases of the Π-free Deletion [VC] problem open.

Our Contributions. We determine the streaming complexity of the general Π-free Deletion [VC] problem. Our main positive result is a unified approach to a single-pass polynomial kernel for Π-free Deletion [VC] for a broad class of families Π. In particular, we show that the kernelization algorithms by Fomin et al. [21] and Jansen and Kroon [24] can be adapted to the streaming setting. The kernels of Fomin et al. [21] consider the case when Π can be characterized by few adjacencies, which intuitively means that for any vertex of any member of Π, adding or deleting edges between all but a few (say at most c_Π) distinguished other vertices does not change membership of Π. The exponent of the polynomial kernels depends on c_Π. Jansen and Kroon [24] considered even more general families Π. We show that these kernels can be computed in the AL model using a single pass and polynomial space (where the exponent depends on c_Π). This generalizes the previous results by Bishnu et al. [5] as well as their kernel for $\mathcal{F}$-Subgraph Deletion [VC].

To complement the kernels, we take a direct approach to find more memory-efficient algorithms, at the cost of using many passes. We show novel parameterized streaming algorithms that require $\tilde{\mathcal{O}}(K^2)$ space and $\mathcal{O}(K)^{\mathcal{O}(K)}$ passes. Here, all hidden constants depend on c_Π. Crucially, however, the exponent of the space usage of these algorithms does not, which provides an advantage over computing the kernel. We also provide explicit streaming algorithms for Cluster Vertex Deletion [VC] and Odd Cycle Transversal [VC] that require $\tilde{\mathcal{O}}(K)$ space (both) and $2^K K^2$ and 3^K passes respectively, as well as streaming algorithms for Π-free Deletion [VC,$|V(H)|$] when $\Pi = \{H\}$ and the problem is parameterized by K and $|V(H)|$. A crucial ingredient to these algorithms is a

[4] Otherwise, removing the entire vertex cover is a trivial solution.

streaming algorithm that finds induced subgraphs isomorphic to a given graph H. Further details are provided in Sect. 3.

The above results provide a trade-off in the number of passes and memory complexity of the algorithm used. However, we should justify using both the AL model and the parameter vertex cover. To this end, in Sect. 4, we investigate lower bounds for streaming algorithms for Π-free Deletion [VC]. The (unparameterized) linear lower bound of Chitnis et al. [11] in the EA model requires that Π contains a graph H for which $|E(H)| \geq 2$ and no proper subgraph is a member of Π. We prove that the lower bound extends to both the VA and AL models, with only small adjustments. Hence, parameterization is necessary to obtain sublinear passes and memory for most Π. Since Vertex Cover is one of the few natural graph parameters that has efficient parameterized streaming algorithms [13,17], this justifies the use of the vertex cover parameter. We also extend the reductions by Bishnu et al. [5] to general hardness results for Π-free Deletion in the VA and EA model when the size of the vertex cover is a constant (dependent on Π), justifying the use of the AL model for most Π.

We also consider the parameterized complexity of H-free Deletion [VC] in the non-streaming setting. While polynomial kernels were known in the non-streaming setting [21], we are unaware of any investigation into explicit parameterized algorithms for these problems. We give a general $2^{O(K^2)}\text{poly}(n, |V(H)|)$ time algorithm. This contrasts the situation for H-free Deletion parameterized by the treewidth t of the graph, where a $2^{o(t^{|V(H)|-2})}\text{poly}(n, |V(H)|)$ time lower bound is known under the Exponential Time Hypothesis (ETH) [31]. We also construct a graph property Π for which we provide a lower bound of $2^{o(K \log K)}\text{poly}(n, |V(H)|)$ for Π-free Deletion [VC] under ETH. Further details are provided in Sect. 3.

Preliminaries. We work on undirected graphs $G = (V, E)$, where $|V| = n, |E| = m$. We denote an edge $e \in E$ between $v \in V$ and $u \in V$ with $uv \in E$. For a set of vertices $V' \subseteq V$, denote the subgraph induced by V' as $G[V']$. Denote the neighbourhood of a vertex v with $N(v)$ and for a set S denote $N(S)$ as $\bigcup_{v \in S} N(v)$. We write $N[v]$ for $N(v)$ including v, so $N[v] = N(v) \cup \{v\}$.

We denote the parameters of a problem in $[\cdot]$ brackets, a problem A parameterized by vertex cover number and solution size is denoted by A [VC, ℓ].

2 Adapting Existing Kernels

We first show that very general kernels for vertex cover parameterization admit straightforward adaptations to the AL streaming model. The kernels considered are those by Fomin et al. [21] and by Jansen and Kroon [24]. Fomin et al. [21] provide general kernelization theorems that make extensive use of a single property, namely that some graph properties can be characterized by few adjacencies.

Definition 1. ([21, Definition 3]) *A graph property Π is* characterized by $c_\Pi \in \mathbb{N}$ adjacencies *if for all graphs $G \in \Pi$, for every vertex $v \in V(G)$, there is a set $D \subseteq V(G) \setminus \{v\}$ of size at most c_Π such that all graphs G' that are obtained from G by adding or removing edges between v and vertices in $V(G) \setminus D$, are also contained in Π.*

Fomin et al. show that graph problems such as Π-free Deletion [VC], can be solved efficiently through kernelization when Π is characterized by few adjacencies (and meets some other demands), by making heavy use of the Reduce algorithm they provide. The idea behind the Reduce algorithm is to save *enough* vertices with specific adjacencies in the vertex cover, and those vertices that we forget have equivalent vertices saved to replace them. The sets of adjacencies we have to consider can be reduced by making use of the characterization by few adjacencies, as more than c_Π adjacencies are not relevant. The number of vertices we retain is ultimately dependent on $\ell \leq K$.

In the AL streaming model, we have enough information to compute this kernel, by careful memory management in counting adjacencies towards specific subsets of the vertex cover. The following theorem then shows how this algorithm leads to streaming kernels for Π-free Deletion [VC] as an adaptation of [21, Theorem 2]. We call a graph G *vertex-minimal* with respect to Π if $G \in \Pi$ and for all $S \subsetneq V(G)$, $G[S] \notin \Pi$.

Theorem 1 (♣[5]). *If Π is a graph property such that:*

(i) Π is characterized by c_Π adjacencies,
(ii) every graph in Π contains at least one edge, and
(iii) there is a non-decreasing polynomial $p : \mathbb{N} \to \mathbb{N}$ such that all graphs G that are vertex-minimal with respect to Π satisfy $|V(G)| \leq p(K)$,

then Π-free Deletion [VC] admits a kernel on $\mathcal{O}((K + p(K))K^{c_\Pi})$ vertices in the AL streaming model using one pass and $\mathcal{O}((K + K^{c_\Pi}) \log(n))$ space.

We note that the theorem applies to $\mathcal{F}$-Subgraph Deletion [VC] when $\Delta(\mathcal{F})$ (the maximum degree) is bounded as well as to Cluster Vertex Deletion [VC]. As such, our streaming kernels generalize the kernels of Bishnu et al. [5] for these problems, while the memory requirements and kernel sizes are fairly comparable. A discussion and further implications for several general problems, following Fomin et al. [21], appear in the full version of the paper.

We also give an adaptation (♣) of a more recent kernel by Jansen and Kroon [24], which has another broad range of implications for streaming kernels. This kernel uses a different characterization of the graph family, however, the adaptation to the AL streaming model is very similar. We observe that the adaptation of this kernel leads to a streaming algorithms for problems like Perfect Deletion [VC], AT-free Deletion [VC], Interval Deletion [VC], and Wheel-free Deletion [VC].

3 A Direct FPT Approach

In this section, we give direct FPT streaming algorithms for Π-free Deletion [VC] for the same cases as Theorem 1. This is motivated by the fact that Chitnis and Cormode [10] found a direct FPT algorithm for Vertex Cover

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

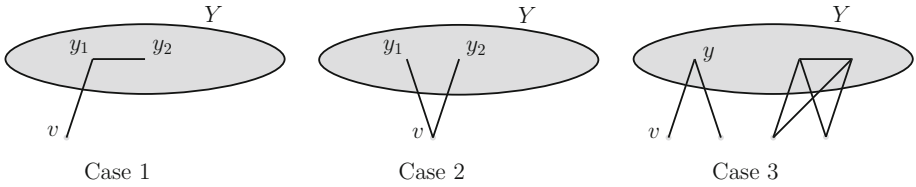

Fig. 1. The different cases how a P_3 can exists with respect to Y, part of the vertex cover. Notice that the case where the entire P_3 is contained in Y is not included here. Case 3 assumes there are no Case 1 or Case 2 P_3's in the graph anymore.

using $\mathcal{O}(2^k)$ passes and only $\tilde{\mathcal{O}}(k)$ space in contrast to the kernel of Chitnis et al. [11] using one pass and $\tilde{\mathcal{O}}(k^2)$ space. Therefore, we aim to explore the pass/memory trade-off for Π-free Deletion [VC] as well.

3.1 P_3-free Deletion

We start with the scenario where $\Pi = \{P_3\}$, which means we consider the problem Cluster Vertex Deletion [VC]. The general idea of the algorithm is to branch on what part of the given vertex cover should be in the solution. For managing the branching correctly, we use a black box enumeration technique also used by Chitnis and Cormode [10]. In a branch, we first check whether the 'deletion-free' part of the vertex cover (Y) contains a P_3, which invalidates a branch. Otherwise, what remains is some case analysis where either one or two vertices of a P_3 lie outside the vertex cover, for which we deterministically know which vertices have to be removed to make the graph P_3-free. We illustrate this step in Fig. 1. Case 1 and 2 have only one option for removal of a vertex. After Case 1 and 2 no longer occur, we can find Case 3 occurrences and show that we can delete all but one of the vertices in such an occurrence. So, if this process can be executed in a limited number of passes, the algorithm works correctly.

To limit the number of passes, the use of the AL model is crucial. Notice that for every pair of vertices y_1, y_2 in the vertex cover, we can identify a Case 1 or 2 P_3 of Fig. 1, or these cases but with v in the vertex cover as well, in a constant number of passes. This is because we can first use a pass to check the presence of an edge between y_1 and y_2, and afterwards use a pass to check the edges of every other vertex towards y_1 and y_2 (which are given together because of the AL model). This means we can find P_3's contained in the vertex cover or corresponding to Case 1 or 2 P_3's in $\mathcal{O}(K^2)$ passes total. The remaining Case 3 can be handled in $\mathcal{O}(K)$ passes from the viewpoint of each $y \in Y$. So this algorithm takes $\mathcal{O}(2^K K^2)$ passes (including branching).

Theorem 2 (♣). *We can solve* Cluster Vertex Deletion [VC] *in the* AL *streaming model using* $\mathcal{O}(2^K K^2)$ *passes and* $\mathcal{O}(K \log n)$ *space.*

Let us stress some details. The use of the AL model is crucial, as it allows us to locally inspect the neighbourhood of a vertex when it appears in the stream. The same approach would require more memory or more passes in other models

Algorithm 1. The procedure FindH.

```
1: function FindH(solution set S, forbidden set Y ⊆ X, integer i)
2:     for each Set O of i vertices of H that can be outside X do ▷ Check non-edges
3:         Denote H' = H \ O
4:         for each Occurrence of H' in Y do   ▷ Check O(binom(|X|, |H|-i)(|H|-i)!) options
5:             S' ← ∅, O' ← O
6:             for each Vertex v ∈ V \ (S ∪ X) do
7:                 Check the edges/non-edges towards H' ∈ Y
8:                 if v is equivalent to some w ∈ O' for H' then
9:                     S' ← S' ∪ {v}, O' ← O' \ {w}
10:                    if O' = ∅ then return S'        ▷ We found an occurrence of H
11:    return ∅                                        ▷ No occurrence of H found
```

to accomplish this result. Also note that we could implement this algorithm in a normal setting (the graph is in memory, and not a stream) to get an algorithm for Cluster Vertex Deletion [VC] with a running time of $\mathcal{O}(2^K \cdot K^2 \cdot (n+m))$.

3.2 H-free Deletion

We now consider a more generalized form of Π-free Deletion [VC], where $\Pi = \{H\}$, a single graph. Unfortunately, the approach when $H = P_3$ does not seem to carry over to this case, because the structure of a P_3 is simple and local.

Theorem 3 (♣). *We can solve H-free Deletion [VC] in $2^{O(K^2)}\,\mathrm{poly}(n, |V(H)|)$ time, where H contains at least one edge and K is the size of the vertex cover.*

In the proof, we rely on the assumption that $\ell < K$ and use that the vertices outside the vertex cover can be partitioned into at most 2^K equivalence classes. Moreover, we use the algorithm implied by the work of Abu-Khzam [1] to find occurrences of H in G.

In order to analyze the complexity with respect to H more precisely and to obtain a streaming algorithm, we present a different algorithm that works off a simple idea. We branch on the vertex cover, and then try to find occurrences of H of which we have to remove a vertex outside the vertex cover. We branch on these removals as well, and repeat this find-and-branch procedure. In an attempt to keep the second branching complexity low, we start by searching for occurrences of H such that only one vertex lies outside the vertex cover, and increase this number as we find no occurrences. For briefness, we only present the occurrence detection part of the algorithm here, a procedure we call FindH. Note that this is not (yet) a streaming algorithm.

Lemma 1 (♣). *Given a graph G with vertex cover X, graph H with at least one edge, and sets $S, Y \subseteq X$, and integer i, Algorithm 1 finds an occurrence of H in G that contains no vertices in S and $X \setminus Y$ and contains $|V(H)| - i$ vertices in Y. It runs in $\mathcal{O}\left(\binom{h}{i}[i^2 + \binom{K}{h-i}(h-i)!((h-i)^2 + Kn + (h-i)in)]\right)$ time, where $|V(H)| = h$ and $|X| = K$.*

FindH is adaptable to the streaming setting, as is the complete algorithm. All the actions FindH takes are local inspection of edges, and many enumeration actions, which lend itself well to usage of the AL streaming model. The number of passes of the streaming version is closely related to the running time of the non-streaming algorithm. This then leads to the full find-and-branch procedure.

Theorem 4 (♣). *We can solve H-free Deletion [VC] in the AL model, where H contains at least one edge, using $\mathcal{O}(2^K h^{K+2} K^h h!)$ or alternatively $\mathcal{O}(2^K h^{K+2} K! h!)$ passes and $\mathcal{O}((K+h^2)\log n)$ space, where $|V(H)| = h$.*

3.3 Towards Π-free Deletion

An issue with extending the previous approach to the general Π-free Deletion problem is the dependence on the maximum size h of the graphs $H \in \Pi$. Without further analysis, we have no bound on h. However, we can look to the preconditions used by Fomin et al. [21] on Π in e.g. Theorem 1 to remove this dependence.

The first precondition is that the set $\Pi' \subseteq \Pi$ of graphs that are vertex-minimal with respect to Π have size bounded by a function in K, the size of the vertex cover. That is, for these graphs $H \in \Pi'$ we have that $|V(H)| \leq p(K)$, where $p(K)$ is some function. We can prove (♣) that it suffices to only remove vertex-minimal elements of Π to solve Π-free Deletion. Note that Fomin et al. [21] require that this is a polynomial, we have no need to demand this. If we also assume that we know the set Π', we obtain the following result.

Theorem 5 (♣). *If Π is a graph property such that:*

(i) we have explicit knowledge of $\Pi' \subseteq \Pi$, which is the subset of q graphs that are vertex-minimal with respect to Π, and
(ii) there is a non-decreasing function $p : \mathbb{N} \to \mathbb{N}$ such that all graphs $G \in \Pi'$ satisfy $|V(G)| \leq p(K)$, and
(iii) every graph in Π contains at least one edge,

then Π-free Deletion [VC] can be solved using $\mathcal{O}(q \cdot 2^K \cdot p(K)^K \cdot K! \cdot K \cdot p(K)! \cdot p(K)^2 \cdot n)$ time.

We argue this algorithm is essentially tight, under the Exponential Time Hypothesis (ETH) [23], by augmenting a reduction by Abu-Khzam et al. [2].

Theorem 6 (♣). *There is a graph property Π for which we cannot solve Π-free Deletion [VC] in $2^{o(K \log K)}\text{poly}(n)$ time, unless ETH fails, where K is the vertex cover number of G, even if each graph that has property Π has size quadratic in its vertex cover number.*

Next, we look to further improve the bound of Theorem 5. Note that so far, we have made no use at all of the characterization by few adjacencies of Π, as in Theorem 1. We now argue that there may be graphs in Π that cannot occur in G simply because it would not fit with the vertex cover.

Lemma 2 (♣). *If Π is a graph property such that*

(i) every graph in Π is connected and contains at least one edge, and
(ii) Π is characterized by c_Π adjacencies,

and G is some graph with vertex cover X, $|X| = K$, and $S \subseteq V(G)$ some vertex set. Then $G[V(G) \setminus S]$ is Π-free if and only if $G[V(G) \setminus S]$ is Π'-free, where $\Pi' \subseteq \Pi$ contains only those graphs in Π with $\leq (c_\Pi + 1)K$ vertices.

The precondition that every graph in Π is connected is necessary to obtain this result. We can use Lemma 2 in combination with Theorem 5 to obtain a new result. Alternatively, using a streaming version of the algorithm instead of the non-streaming one, immediately also provides a streaming result.

Theorem 7 (♣). *Given a graph G with vertex cover X, $|X| = K$, if Π is a graph property such that*

(i) every graph in Π is connected and contains at least one edge, and
(ii) Π is characterized by c_Π adjacencies, and
(iii) we have explicit knowledge of $\Pi' \subseteq \Pi$, which is the subset of q graphs of at most size $(c_\Pi + 1)K$ that are vertex-minimal with respect to Π,

then Π-free Deletion [VC] can be solved using $\mathcal{O}(q \cdot 2^K \cdot ((c_\Pi + 1)K)^K \cdot K! \cdot K \cdot ((c_\Pi + 1)K)! \cdot ((c_\Pi + 1)K)^2 \cdot n)$ time. Assuming $c_\Pi \geq 1$ this can be simplified to $\mathcal{O}(q \cdot 2^K \cdot {c_\Pi}^K \cdot K^{K+3} \cdot K! \cdot (c_\Pi K)! \cdot n)$ time. In the streaming setting, Π-free Deletion [VC] can be solved using $\mathcal{O}(q \cdot 2^K \cdot c_\Pi^K \cdot K^{K+2} \cdot K! \cdot (c_\Pi K)!)$ passes in the AL streaming model, using $\tilde{\mathcal{O}}((c_\Pi K)^2 + q \cdot (c_\Pi + 1)K)$ space.

The required explicit knowledge of Π' might give memory problems. That is, we have to store Π' somewhere to make this algorithm work, which takes $\tilde{\mathcal{O}}(q \cdot (c_\Pi + 1)K)$ space. Note that q can range up to $K^{\mathcal{O}(K)}$. We adapt the streaming algorithm to the case when we have oracle access to Π in ♣.

3.4 Odd Cycle Transversal

Specific forms of Π-free Deletion [VC] allow for improvement over Theorem 7, which we illustrate for the problem of Odd Cycle Transversal [VC]. Note that odd cycle-free and induced odd cycle-free are equivalent.

Odd Cycle Transversal [VC]
Input: A graph G with a vertex cover X, and an integer ℓ.
Parameter: The size $K := \|X\|$ of the vertex cover.
Question: Is there a set $S \subseteq V(G)$ of size at most ℓ such that $G[V(G) \setminus S]$ contains no induced odd cycles?

The interest in this problem comes from the FPT algorithm using iterative compression provided in [15, Section 4.4], based on work by Reed et al. [30]. Although Chitnis and Cormode [10] have shown how iterative compression can

be used in the streaming setting, adapting the algorithm out of Reed et al. seems difficult. The main cause for this is the use of a maximum-flow algorithm, which does not seem to lend itself well to the streaming setting because of its memory requirements. Instead, we present the following approach.

It is well known that a graph without odd cycles is a bipartite graph (and thus 2-colourable) and vice versa. In the algorithm, we guess what part of the vertex cover is in the solution, and then we guess the colouring of the remaining part. Then vertices outside the vertex cover for which not all neighbours have the same colour must be deleted. This step can be done in one pass if we use the AL streaming model. In the same pass, we can also check if the colouring is valid within the vertex cover. If the number of deletions does not exceed the solution size and the colouring is valid within the vertex cover, then the resulting graph is bipartite and thus odd cycle free.

The total number of guesses comes down to $\mathcal{O}(3^K)$ options, as any vertex in the vertex cover is either in the solution, coloured with colour 1 or coloured with colour 2. This directly corresponds to the number of passes, as only one pass is needed per guessed colouring.

Theorem 8 (♣). *Given a graph G given as an AL stream with vertex cover X, $|X| = K$, we can solve* Odd Cycle Transversal [VC] *using $\mathcal{O}(3^K)$ passes and $\mathcal{O}(K \log n)$ space.*

If we think about this algorithm, we can notice that often the colouring we guess on the vertex cover is invalid. An alternative approach (♣) follows by noting that a connected component within the vertex cover can only have two possible valid colourings. We can exploit this to decrease the number of passes when the number of connected components in the vertex cover is low. This comes at a price: to easily find components of the vertex cover, we store it in memory, which increases the memory complexity. Alternatively, we can use $\mathcal{O}(K)$ passes to find the connected components of the vertex cover in every branch.

4 Lower Bounds

We show lower bounds for Π-free Deletion. To prove lower bounds for streaming, we can show reductions from problems in communication complexity, as first shown by Henzinger et al. [22]. An example of such a problem is Disjointness.

Disjointness
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 $y_1 y_2 \dots y_n$.
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 following proposition is given and used by Bishnu et al. [5], and gives us one important consequence of reductions from a problem in communication complexity to a problem for streaming algorithms.

Proposition 1. *(Rephrasing of item (ii) of [5, Proposition 5.6]) If we can show a reduction from* Disjointness *to problem Π in streaming model $\mathcal{M}$ such that the reduction uses a 1-pass streaming algorithm of Π as a subroutine, then any streaming algorithm working in the model $\mathcal{M}$ for Π that uses p passes requires $\Omega(n/p)$ bits of memory, for any $p \in \mathbb{N}$ [3,7,12].*

The structure of these reductions is relatively simple: have Alice and Bob construct the input for a streaming algorithm depending on their input to Disjointness. If we do this in such a manner that the solution the streaming algorithm outputs gives us exactly the answer to Disjointness, we can conclude that the streaming algorithm must abide the lower bound of Disjointness.

Chitnis et al. [11, Theorem 6.3] prove hardness for many Π, those that obide to a small precondition. However, Chitnis et al. do not describe in their reduction how Alice and Bob give their 'input' as a stream to the algorithm for Π-free Deletion, and thus it would apply only to the EA streaming model. However, if we observe the proof closely, we can see it extends to the VA model.

We would also like it to extend to the AL model. However, this requires a slightly stronger precondition on the graph class Π.

Theorem 9. *If Π is a set of graphs such that each graph in Π is connected, and there is a graph $H \in \Pi$ such that*

- *H is a minimal element of Π under the operation of taking subgraphs, i.e., no proper subgraph of H is in Π, and*
- *H has at least two **disjoint** edges,*

then any p-pass (randomized) streaming algorithm working on the AL *streaming model for Π-free Deletion [ℓ] needs $\Omega(n/p)$ bits of space.*

Proof. We add onto the proof of [11, Theorem 6.3], by specifying how Alice and Bob provide the input to the p-pass streaming algorithm.

Let H be a minimal graph in Π which has at least two disjoint edges, say e_1 and e_2. Let $H' \coloneqq H \setminus \{e_1, e_2\}$. Create as an input for the streaming algorithm n copies of H', where in copy i we add the edges e_1 and e_2 if and only if the input of Disjointness has a 1 for index i for Alice and Bob respectively.

As e_1 and e_2 are disjoint, e_2 is incident on two vertices v, w which are not incident to e_1. For every pass the algorithm requires, we do the following. We provide all the copies of H as input to the streaming algorithm by letting Alice input all vertices $V(H) \setminus \{v, w\}$ as an AL stream. Note that Alice has enough information to do this, as the vertices incident on the edge e_2 in each copy of H is never included in this part of the stream. Then Alice passes the memory of the streaming algorithm to Bob, who inputs the edges incident to the vertices v, w for each copy of H (which includes e_2 if and only if the respective bit in the input of Disjointness is 1). This ends a pass of the stream.

Note that Alice and Bob have input the exact specification of a graph as described by Chitnis et al. [11, Theorem 6.3], but now as a AL stream. Hence, the correctness follows. □

Theorem 9 provides a lower bound for, for example, Even Cycle Transversal $[\ell]$ (where Π is the set of all graphs that contain a $C_4, C_6, \ldots$), and similarly Odd Cycle Transversal $[\ell]$ and Feedback Vertex Set $[\ell]$. Theorem 9 does not hold for the scenario where Π contains only stars.

Notice that the lower bound proof makes a construction with a vertex cover size linear in n. Therefore, these bounds do not hold when the vertex cover size is bounded. We can prove lower bounds with constant vertex cover size for H-free Deletion with specific requirements on H, for the EA and VA models. These results follow by adapting the known lower bound construction by Bishnu et al. [5]. Here, we give a summarizing theorem for these lower bounds.

Theorem 10. (♣). *If H is such that either:*

1. *H is a connected graph with at least 3 edges and a vertex of degree 2, or,*
2. *H is a graph with a vertex of degree at least 2 for which every neighbour has an equal or larger degree,*

then any algorithm for solving H-free Deletion [VC] on a graph G with $K \geq |\mathrm{VC}(H)| + 1$ requires $\Omega(n/p)$ bits when using p passes in the VA/EA models, even when the solution size $\ell = 0$.

Theorem 10 proves lower bounds for Odd Cycle Transversal [VC], Even Cycle Transversal [VC], Feedback Vertex Set [VC], and Cograph Deletion [VC]. Examples for which Theorem 10 does not give a lower bound include Cluster Vertex Deletion [VC] (indeed, then a kernel is known [5]), or more generally, H-free Deletion [VC] when H is a star.

5 Conclusion

We have seen different streaming algorithms and lower bounds for Π-free Deletion and its more specific forms, making use of the minimum vertex cover as a parameter. We have seen the potency of the AL streaming model in combination with the vertex cover, where in other streaming models lower bounds arise. It is interesting that for very local structures like a P_3, this combination works effortlessly, giving a very efficient memory-optimal algorithm. For more general structures troubles arise, but nonetheless, we can solve the more general problems with a many-pass, low-memory approach. Alternatively, the adaptations of kernels gives rise to a few-pass, high-memory algorithm, which provides a possible trade-off when choosing an algorithm.

We also propose the following open problems. Can lower bounds be found expressing a pass/memory trade-off in the vertex cover size for the Π-free Deletion [VC] problem? Or alternatively, can we find an upper bound for Π-free Deletion [VC] using $\mathcal{O}(K \log n)$ bits of memory but only a polynomial in K number of passes? Essentially, here we ask whether or not our algorithm is reasonably tight, or can be improved to only use a polynomial number of passes in K. A lower bound expressing a trade-off in terms of the vertex cover

size is a standalone interesting question, as most lower bound statements about streaming algorithms express a trade-off in terms of n.

We also ask about the unparameterized streaming complexity of Cluster Vertex Deletion in the AL model. While lower bounds for most other Π-free Deletion problems in the AL model follow from our work (Theorem 9) and earlier work of Bishnu et al. [5], this appears an intriguing open case.

Finally, we ask if there is a $2^{o(K \log K)}$ lower bound for Π-free Deletion [VC] when Π is characterized by few adjacencies?

References

1. Abu-Khzam, F.N.: Maximum common induced subgraph parameterized by vertex cover. Inf. Process. Lett. **114**(3), 99–103 (2014)
2. Abu-Khzam, F.N., Bonnet, É., Sikora, F.: On the complexity of various parameterizations of common induced subgraph isomorphism. Theor. Comput. Sci. **697**, 69–78 (2017)
3. Agarwal, D., McGregor, A., Phillips, J.M., Venkatasubramanian, S., Zhu, Z.: Spatial scan statistics: approximations and performance study. In: Proceedings of SIGKDD 2006, pp. 24–33. ACM (2006)
4. Agrawal, A., et al.: Parameterized streaming algorithms for min-ones d-sat. In: Proceedings of FSTTCS 2019. LIPIcs, vol. 150, pp. 8:1–8:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)
5. Bishnu, A., Ghosh, A., Kolay, S., Mishra, G., Saurabh, S.: Fixed-parameter tractability of graph deletion problems over data streams. CoRR abs/1906.05458 (2019)
6. Bishnu, A., Ghosh, A., Kolay, S., Mishra, G., Saurabh, S.: Fixed parameter tractability of graph deletion problems over data streams. In: Kim, D., Uma, R.N., Cai, Z., Lee, D.H. (eds.) COCOON 2020. LNCS, vol. 12273, pp. 652–663. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58150-3_53
7. Bishnu, A., Ghosh, A., Mishra, G., Sen, S.: On the streaming complexity of fundamental geometric problems. CoRR abs/1803.06875 (2018)
8. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernelization lower bounds by cross-composition. SIAM J. Discret. Math. **28**(1), 277–305 (2014)
9. Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. Inf. Process. Lett. **58**(4), 171–176 (1996)
10. Chitnis, R., Cormode, G.: Towards a theory of parameterized streaming algorithms. In: Proc. IPEC 2019. LIPIcs, vol. 148, pp. 7:1–7:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)
11. Chitnis, R., et al.: Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In: Proceedings of SODA 2016, pp. 1326–1344. SIAM (2016)
12. Chitnis, R.H., Cormode, G., Esfandiari, H., Hajiaghayi, M., Monemizadeh, M.: Brief announcement: New streaming algorithms for parameterized maximal matching & beyond. In: Proceedings of SPAA 2015, pp. 56–58. ACM (2015)
13. Chitnis, R.H., Cormode, G., Hajiaghayi, M.T., Monemizadeh, M.: Parameterized streaming: Maximal matching and vertex cover. In: Proceedings of SODA 2015, pp. 1234–1251. SIAM (2015)

14. Cormode, G., Dark, J., Konrad, C.: Independent sets in vertex-arrival streams. In: Proceedings of ICALP 2019. LIPIcs, vol. 132, pp. 45:1–45:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)
15. Cygan, M., et al.: Parameterized Algorithms. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-319-21275-3
16. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science, Springer, Heidelberg (1999). https://doi.org/10.1007/978-1-4612-0515-9
17. Fafianie, S., Kratsch, S.: Streaming kernelization. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS 2014. LNCS, vol. 8635, pp. 275–286. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44465-8_24
18. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: Graph distances in the streaming model: the value of space. In: Proceedings of SODA 2005, pp. 745–754. SIAM (2005)
19. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. Theor. Comput. Sci. **348**(2–3), 207–216 (2005)
20. Fomin, F.V., Golovach, P.A., Thilikos, D.M.: On the parameterized complexity of graph modification to first-order logic properties. Theory Comput. Syst. **64**(2), 251–271 (2020). https://doi.org/10.1007/s00224-019-09938-8
21. Fomin, F.V., Jansen, B.M.P., Pilipczuk, M.: Preprocessing subgraph and minor problems: when does a small vertex cover help? J. Comput. Syst. Sci. **80**(2), 468–495 (2014)
22. Henzinger, M.R., Raghavan, P., Rajagopalan, S.: Computing on data streams. In: Abello, J.M., Vitter, J.S. (eds.) Proceedings of DIMACS 1998. DIMACS, vol. 50, pp. 107–118. DIMACS/AMS (1998)
23. Impagliazzo, R., Paturi, R.: On the complexity of k-SAT. J. Comput. Syst. Sci. **62**(2), 367–375 (2001)
24. Jansen, B.M.P., de Kroon, J.J.H.: Preprocessing vertex-deletion problems: characterizing graph properties by low-rank adjacencies. In: Proceedings of SWAT 2020, LIPIcs, vol. 162, pp. 27:1–27:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)
25. Jansen, B.M.: The power of data reduction: Kernels for fundamental graph problems. Ph.D. thesis, Utrecht University (2013)
26. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. J. Comput. Syst. Sci. **20**(2), 219–230 (1980)
27. McGregor, A.: Graph stream algorithms: a survey. SIGMOD Rec. **43**(1), 9–20 (2014)
28. McGregor, A., Vorotnikova, S., Vu, H.T.: Better algorithms for counting triangles in data streams. In: Proceedings of PODS 2016, pp. 401–411. ACM (2016)
29. Muthukrishnan, S.: Data streams: algorithms and applications. Found. Trends Theor. Comput. Sci. **1**(2) (2005)
30. Reed, B.A., Smith, K., Vetta, A.: Finding odd cycle transversals. Oper. Res. Lett. **32**(4), 299–301 (2004)
31. Sau, I., dos Santos Souza, U.: Hitting forbidden induced subgraphs on bounded treewidth graphs. In: Proceedings of MFCS 2020. LIPIcs, vol. 170, pp. 82:1–82:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)