Kernelization via Sampling with Applications to Dynamic Graph Streams

Rajesh Chitnis[*] Graham Cormode[†] Hossein Esfandiari[‡] MohammadTaghi Hajiaghayi[‡]
Andrew McGregor[§] Morteza Monemizadeh[¶] Sofya Vorotnikova[§]

Abstract

In this paper we present a simple but powerful subgraph sampling primitive that is applicable in a variety of computational models including dynamic graph streams (where the input graph is defined by a sequence of edge/hyperedge insertions and deletions) and distributed systems such as MapReduce. In the case of dynamic graph streams, we use this primitive to prove the following results:

- *Matching:* Our main result for matchings is that there exists an $\tilde{O}(k^2)$ space algorithm that returns the edges of a maximum matching on the assumption the cardinality is at most k. The best previous algorithm used $\tilde{O}(kn)$ space where n is the number of vertices in the graph and we prove our result is optimal up to logarithmic factors. Our algorithm has $\tilde{O}(1)$ update time. We also show that there exists an $\tilde{O}(n^2/\alpha^3)$ space algorithm that returns an α-approximation for matchings of arbitrary size. In independent work, Assadi et al. (arXiv 2015) proved this is optimal and provided an alternative algorithm. We generalize our exact and approximate algorithms to weighted matching. While there has been a substantial amount of work on approximate matching in insert-only graph streams, these are the first non-trivial results in the dynamic setting.
- *Vertex Cover and Hitting Set:* There exists an $\tilde{O}(k^d)$ space algorithm that solves the minimum hitting set problem where d is the cardinality of the input sets and k is an upper bound on the size of the minimum hitting set. We prove this is optimal up to logarithmic factors. Our algorithm has $\tilde{O}(1)$ update time. The case $d = 2$ corresponds to minimum vertex cover.

Finally, we consider a larger family of parameterized problems (including b-matching, disjoint paths, vertex coloring among others) for which our subgraph sampling primitive yields fast, small-space dynamic graph stream algorithms. We then show lower bounds for natural problems outside this family.

[*] The Weizmann Institute of Science, Rehovot, Israel. Supported by a postdoctoral fellowship from I-CORE ALGO. Email: rajesh.chitnis@weizmann.ac.il

[†] Department of Computer Science, University of Warwick, UK. Supported in part by European Research Council grant ERC-2014-CoG 647557, the Yahoo Faculty Research and Engagement Program and a Royal Society Wolfson Research Merit Award. Email: g.cormode@warwick.ac.uk.

[‡] Department of Computer Science, University of Maryland. Supported in part by NSF CAREER award 1053605, NSF Grant CCF-1161626, ONR YIP award N000141110662, DARPA/AFOSR grant FA9550-12-1-0423, and a Google Faculty Research award. Email: {hossein, hajiagha}@cs.umd.edu

[§] University of Massachusetts Amherst. Supported by NSF CAREER Award CCF-0953754 and CCF-1320719 and a Google Faculty Research Award. Email: {mcgregor,svorotni}@cs.umass.edu

[¶] Computer Science Institute of Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic. Partially supported by the project 14-10003S of GA ČR. Part of this work was done when the author was at Department of Computer Science, Goethe-Universität Frankfurt, Germany and supported in part by MO 2200/1-1. Email: monemi@iuuk.mff.cuni.cz

1 Introduction

Over the last decade, a growing body of work has considered solving graph problems in the data stream model. Most of the early work considered the insert-only variant of the model where the stream consists of edges being added to the graph and the goal is to compute properties of the graph using limited memory. Recently, however, there has been a significant amount of interest in being able to process dynamic graph streams where edges are both added and deleted from the graph [3, 6–8, 10, 27, 28, 33, 34, 40]. These algorithms are all based on the surprising efficacy of using random linear projections, aka linear sketching, for solving combinatorial problems. Results include testing edge connectivity [6] and vertex connectivity [28], constructing sparsifiers [7, 8, 33], approximating the densest subgraph [10, 20, 43], correlation clustering [3], and estimating the number of triangles [40]. For a recent survey of the area, see [42].

The concept of *parameterized stream algorithms* was explored by Chitnis et al. [13] and Fafianie and Kratsch [22]. Their work investigated a natural connection between data streams and parameterized complexity. In parameterized complexity, the time cost of a problem is analyzed in terms of not only the input size but also other parameters of the input. For example, while the classic vertex cover problem is NP complete, it can be solved via a simple branching algorithm in time $2^k \cdot \mathrm{poly}(n)$ where k is the size of the optimal vertex cover. An important concept in parameterized complexity is *kernelization* in which the goal is to efficiently transform an instance of a problem into a smaller instance such that the smaller instance is a "yes" instance (e.g., has a solution of at least a certain size) iff the original instance was also a "yes" instance. For more background on parameterized complexity and kernelization, see [16, 24]. Parameterizing the *space* complexity of a problem in terms of the size of the output is a particularly appealing notion in the context of data stream computation. In particular, the space used by any algorithm that returns an actual solution (as opposed to an estimate of the size of the solution) is necessarily at least the size of the solution.

Our Results and Related Work. In this paper we present a simple but powerful subgraph sampling primitive that is applicable in a variety of computational models including dynamic graph streams (where the input graph is defined by a sequence of edge/hyperedge insertions and deletions) and distributed systems such as MapReduce. This primitive will be useful for both those parameterized problems whose output has bounded size and for those where the optimal solution need not be bounded. In the case where the output has bounded size, our results can be thought of as *kernelization via sampling*, i.e., we sample a relatively small set of edges according to a simple (but not uniform) sampling procedure and can show that the resulting graph has a solution of size at most k iff the original graph has an optimal solution of size at most k. We present the subgraph sampling primitive and implementation details in Section 2.

Graph Matchings. Finding a large matching is the most well-studied graph problem in the data stream model [4, 5, 15, 18, 23, 26, 31, 32, 37, 38, 41, 48]. However, all of the existing single-pass stream algorithms are restricted to the insert-only case, i.e., edges may be inserted but will never be deleted. This restriction is significant: for example, the simple greedy algorithm using $\tilde{O}(n)$ space returns a 2-approximation if there are no deletions. In contrast, prior to this paper no $o(n)$-approximation was known in the dynamic case when there are both insertions and deletions. Finding an algorithm for the dynamic case of this fundamental graph problem was posed as an open problem in the Bertinoro Data Streams Open Problem List [1].

We prove the following results for computing a matching in the dynamic model. Our first result is an $\tilde{O}(k^2)$ space algorithm that returns the edges of a maximum matching on the assumption that its cardinality is at most k. Our algorithm has $\tilde{O}(1)$ update time. The best previous algorithm [13] collects $\max(\deg(u), 2k)$ edges incident to each vertex u and finds the optimal matching amongst these edges. This algorithm can be implemented in $\tilde{O}(kn)$ space where n is the number of vertices in the graph. Indeed obtaining an algorithm with $f(k)$ space, for any function f, in the dynamic graph stream case was left as an important open problem [13]. We can also extend our approach to maximum weighted matching. Our second result is an optimal $\tilde{O}(n^2/\alpha^3)$ space algorithm that returns an α-approximation for matchings of arbitrary size. For example, this implies an $n^{1/3}$ approximation using $\tilde{O}(n)$ space, commonly known as the *semi-streaming* space restriction [23, 44]. We present our second result and an algorithm for graphs with bounded arboricity, along with a discussion of very recent related work [9, 11, 36], in Section 5.

Vertex Cover and Hitting Set. We next consider the problem of finding the *minimum vertex cover* and its generalization, *minimum hitting set*. The hitting set problem can be defined in terms of hypergraphs: given a set of hyperedges, select the minimum set of vertices such that every hyperedge contains at least one of the selected vertices. If all hyperedges have cardinality two, this is the vertex cover problem.

There is a growing body of work analyzing hypergraphs in the data stream model [17, 28, 35, 45–47]. For example, Emek and Rosén [17] studied the following set-cover problem which is closely related to the hitting set problem: given a stream of hyperedges (without deletions), find the minimum subset of these hyperedges such that every vertex is included in at least one of the hyperedges. They present an $O(\sqrt{n})$ approximation streaming algorithm using $\tilde{O}(n)$ space along with results for covering all but a small fraction of the vertices. Another related problem is independent set since the minimum vertex cover is the complement of the maximum independent set. Halldórsson et al. [29] presented streaming algorithms for finding large independent sets but these do not imply a result for vertex cover in either the insert-only or dynamic setting.

In Section 3.2, we present a $\tilde{O}(k^d)$ space algorithm that finds the minimum hitting set where d is the cardinality of the input sets and k is an upper bound on the cardinality of the minimum hitting set. We prove the space use is optimal and matches the space used by previous algorithms in the insert-only model [13, 22]. Our algorithms can be implemented with $\tilde{O}(1)$ update time. The only previous results in the dynamic model were by Chitnis et al. [13] and included a $\tilde{O}(kn)$ space algorithm and a $\tilde{O}(k^2)$ space algorithm under a much stronger "promise" that the vertex cover of the graph defined by any prefix of the stream may never exceed k. Relaxing this promise remained as the main open problem of Chitnis et al. [13]. In Section 3.2, we also generalize our exact matching result to hypergraphs. In Section D, we show our result is also optimal.

General Family of Results. We consider a larger family of parameterized problems for which our subgraph sampling primitive yields fast, small-space dynamic graph stream algorithms. This result is presented in Section 4, while lower bounds for various problems outside this family are proved in Section D.

2 Basic Subgraph Sampling Technique

Basic Approach and Intuition. The inspiration for our subgraph sampling primitive is the following simple procedure for *edge* sampling. Given a graph $G = (V, E)$ and probability $p \in [0, 1]$, let $\mu_{G,p}$ be the distribution $E \cup \{\perp\}$ defined by the following process:

1. Sample each vertex independently with probability p and let V' denote the set of sampled vertices.

2. Return an edge chosen uniformly at random from the edges in the induced graph on V'. If no such edge exists, return $\perp$.

The distribution $\mu_{G,p}$ has some surprisingly useful properties. For example, suppose that the optimal matching in a graph G has size at most k. It is possible to show that this matching has the same size as the optimal matching in the graph formed by taking $O(k^2)$ independent samples from $\mu_{G,1/k}$. It is not hard to show that such a result would not hold if the edges were sampled uniformly at random.[1] The intuition is that

[1] To see this, consider a layered graph on vertices $L_1 \cup L_2 \cup L_3 \cup L_4$ with edges forming a complete bipartite graph on $L_1 \times L_2$, a complete bipartite matching on $L_2 \times L_3$, and a perfect matching on $L_3 \times L_4$. If $|L_1| = n \gg k$ and $|L_2| = |L_3| = |L_4| = k/2$ then the maximum matching has size k and every matching includes all edges in the perfect matching on $L_3 \times L_4$. Since there are $\Omega(nk)$ edges in this graph we would need $\Omega(nk)$ edges sampled uniformly before we find the matching on $L_3 \times L_4$.

when we sample from $\mu_{G,p}$ we are less likely to sample an edge incident to a high degree vertex then if we sampled uniformly at random from the edge set. For a large family of problems including matching, it will be advantageous to avoid bias towards edges whose endpoints have high degree.

Our subgraph sampling primitive essentially parallelizes the process of sampling from $\mu_{G,p}$. This will lead to more efficient algorithms in the dynamic graph stream model. The basic idea is rather than select a subset of vertices V', we randomly partition V into $V_1 \cup V_2 \cup \ldots \cup V_{1/p}$. Selecting a random edge from the induced graph on any V_i results in an edge distributed as in $\mu_{G,p}$. Sampling an edge on each V_i results in $1/p$ samples from $\mu_{G,p}$ although note that the samples are no longer independent. This lack of independence will not be an issue and will sometimes be to our advantage. In many applications it will make sense to parallelize the sampling further and select a random edge between each pair, V_i and V_j, of vertex subsets. For applications involving hypergraphs we select random edges between larger subsets of $\{V_1, V_2, \ldots, V_{1/p}\}$.

Sampling Data Structure: We now present the subgraph sampling primitive formally. Given an unweighted (hyper)graph $G = (V, E)$, consider a "coloring" defined by a function $c : V \to [b]$. It will be convenient to introduce the notation: for each $i \in [b]$

$$V_i = \{v \in V : c(v) = i\}$$

and say that every vertex in V_i has color i. For a (hyper)edge $e \in E$, we define $c(e) = \{i \ : \ v \in e, c(v) = i\}$, i.e., $c(e)$ is exactly the set of colors seen on the vertices of e. For $S \subseteq [b]$, we say that an (hyper)edge e of G is *S-colored* if $c(e) = S$, i.e., each color from S is used to color the vertices in e and no other colors are used. Given a constant $q \geq 1$ which denotes the "size restriction", for each $S \subseteq [b]$ of cardinality at most q, E_S contains a single edge chosen uniformly at random from the set of all S-colored edges. If there are no S-colored edges, then $E_S = \emptyset$. The union of these sets defines the random graph $G' = (V, E')$, i.e.,

$$E' = \bigcup_{S \subseteq [b], |S| \leq q} E_S \ .$$

For example, given a simple graph, if we have $q = 1$ then for each color $i \in [b]$ we choose an edge whose endpoints are both colored i. If $q = 2$, then for every $1 \leq i \leq j \leq b$ we choose an edge whose one endpoint has color i and the other endpoint has color j: note that this includes the possibility that $i = j$. In the case of a weighted graph, for each distinct weight w we choose a single edge $E_{S,w}$ uniformly at random from the set of S-colored edges with weight w.

Definition 1. *We define* $\mathsf{Sample}_{b,q,1}$ *to be the distribution over subgraphs generated as above where c is chosen uniformly at random from a family of pairwise independent hash functions.* $\mathsf{Sample}_{b,q,r}$ *is the distribution over graphs formed by taking the union of r independent graphs sampled from* $\mathsf{Sample}_{b,q,1}$. *Algorithm 1 gives pseudocode for sampling from* $\mathsf{Sample}_{b,q,r}$.

Motivating Application. As a first application to motivate the subgraph sampling primitive we again consider the problem of estimating matchings. We will use the following simple lemma that will also be useful in subsequent sections (the proof, along with other omitted proofs, can be found in Appendix A).

Lemma 2. *Let $U \subseteq V$ be an arbitrary subset of $|U| = r$ vertices and let $c : V \to [4r\epsilon^{-1}]$ be a pairwise independent hash function. Then with probability at least $3/4$, at least $(1-\epsilon)r$ of the vertices in U are hashed to distinct values. Setting $\epsilon < 1/r$ ensures all vertices are hashed to distinct values with this probability.*

Suppose G is a graph with a matching $M = \{e_1, \ldots, e_k\}$ of size k. Let $G' \sim \mathsf{Sample}_{b,2,1}$. By the above lemma, there exists $b = O(k^2)$, such that all the $2k$ endpoints of edges in M are colored differently with

Algorithm 1 Algorithm for Sampling Subgraphs According to $\mathsf{Sample}_{b,q,r}$

```
Input: A (hyper)graph $G = (V, E)$ and natural numbers $b, q, r$.
Output: A subgraph $G' = (V, E')$ where $E' \subseteq E$
Choose $c_1, \dots, c_r$ u.a.r. from a family of pairwise independent hash functions mapping $V$ to $[b]$
Set $E' = \emptyset$
for $1 \leq j \leq r$ do
    for each $S \subseteq [b]$ such that $|S| \leq q$ do
        Select an edge $E^j_S$ u.a.r. from the set of $S$-colored edges $\{e \in E \ : \ \cup_{v \in e} c_j(v) = S\}$
        $E' \leftarrow E' \cup E^j_S$
Report the graph $G' = (V, E')$.
```

constant probability. Suppose the endpoints of edge e_i received the colors a_i and b_i. Then G' contains an edge in $E_{\{a_i,b_i\}}$ for each $i \in [k]$. Assuming all endpoints receive different colors, no edge in $E_{\{a_i,b_i\}}$ shares an endpoint with an edge in $E_{\{a_j,b_j\}}$ for $j \neq i$. Hence, we can conclude that G' also has a matching of size k. In Section 4, we show that a similar approach can be generalized to a range of problems. Using a similar argument there exists $b = O(k)$ such that G' contains a constant approximation to the optimum matching. However, in Section 3, we show that there exists $b = O(k)$ such that with high probability graphs sampled from $\mathsf{Sample}_{b,2,O(\log k)}$ preserve the size of the optimal matching exactly.

2.1 Application to Dynamic Data Streams and MapReduce

We now describe how the subgraph sampling primitive can be implemented in various computational models.

Dynamic Graph Streams. Let S be a stream of insertions and deletions of edges of an underlying graph $G(V, E)$. We assume that vertex set $V = \{1, 2, \dots, n\}$. We assume that the length of stream is polynomially related to n and hence $O(\log |S|) = O(\log n)$. We denote an undirected edge in E with two endpoints $u, v \in V$ by uv. For weighted graphs, we assume that the weight of an edge is specified when the edge is inserted and deleted and that the weight never changes. The following theorem (proved in the appendix) establishes that the sampling primitive can be efficiently implemented in dynamic graph streams.

Theorem 3. *Suppose G is a graph with w_0 distinct weights. It is possible to sample from $\mathsf{Sample}_{b,q,r}$ with probability at least $1 - \delta$ in the dynamic graph stream model using $\tilde{O}(b^q r w_0)$ space and $\tilde{O}(r)$ update time.*

MapReduce and Distributed Models. The sampling distribution is naturally parallel, making it straightforward to implement in a variety of popular models. In MapReduce, the r hash functions can be shared state among all machines, allowing Map function to output each edge keyed by its color under each hash function. Then, these can be sampled from on the Reduce side to generate the graph G'. Optimizations can do some data reduction on the Map side, so that only one edge per color class is emitted, reducing the communication cost. A similar outline holds for other parallel graph models such as Pregel.

3 Parameterized Matching, Vertex Cover, and Hitting Set

3.1 Finding Maximum Matchings and Minimum Vertex Covers Exactly

In this section, we present results on finding edges in a maximum matching and the vertices in a minimum vertex cover of a graph G. We use $\mathrm{match}(G)$ to denote the size of the maximum (weighted or unweighted as appropriate) matching in G and use $\mathrm{vc}(G)$ to denote the size of minimum vertex cover. The main theorem we prove in this section is that a set of edges is a maximum matching in the sampled graph iff it is a maximum matching in the original graph:

Theorem 4 (Finding Exact Solutions)**.** *Suppose* $\mathrm{match}(G) \le k$*. Then, with probability* $1 - 1/\mathrm{poly}(k)$*,*

$$\mathrm{match}(G') = \mathrm{match}(G) \ \textit{and} \ \mathrm{vc}(G') = \mathrm{vc}(G) \ ,$$

where $G' = (V, E') \sim \mathsf{Sample}_{1000k,2,O(\log k)}$*.*

Intuition and Preliminaries. To argue that G' has a matching of the optimal size, it suffices to show that for every edge $uv \in G$ that is not in G', there are a large number of edges incident to one or both of u and v that is in G'. If this is the case, then it will still be possible to match at least one of these vertices in G'.

To make this precise, let U be the subset of vertices with degree at least $10k$. Let F be the set of edges in the induced subgraph on $V \setminus U$, i.e., the set of edges whose endpoints both have small degree. We will prove that with high probability,

$$\big(F \subseteq E'\big) \quad \text{and} \quad (\forall u \in U \ , \ \deg_{G'}(u) \ge 5k) \ , \tag{1}$$

where E' is the set of edges in G'. Note that any sampled graph G' that satisfies (1) has the property that for all edges $uv \in G$ that are not in G' we have $\deg_{G'}(u) \ge 5k$ or $\deg_{G'}(v) \ge 5k$.

Analysis. The first lemma establishes that it is sufficient to prove that (1) holds with high probability.

Lemma 5. *If* $\mathrm{match}(G) \le k$ *then* (1) *implies* $\mathrm{match}(G') = \mathrm{match}(G)$ *and* $\mathrm{vc}(G') = \mathrm{vc}(G)$*.*

The next lemma establishes that (1) holds with the required probability.

Lemma 6. *Property* (1) *holds with probability at least* $1 - 1/\mathrm{poly}(k)$*.*

Proof. Let $\mathrm{VC}(G)$ be a minimum vertex cover of G. Note that $\mathrm{match}(G) \le k$ implies that $\mathrm{vc}(G) = |\mathrm{VC}(G)| \le 2k$ because the endpoints of the edges in a maximum matching form a vertex cover. Next consider $H \sim \mathsf{Sample}_{1000k,2,1}$. We will show that for any $e \in F$ and $u \in U$,

$$\mathbb{P}\left[e \in H\right] > 1/2 \quad \text{and} \quad \mathbb{P}\left[\deg_H(u) \ge 5k\right] \ge 1/2 \ .$$

It follows that if $r = O(\log k)$ and $G' \sim \mathsf{Sample}_{1000k,2,r}$ then

$$\mathbb{P}\left[e \in G' \text{ and } \deg_{G'}(u) \ge 5k\right] \ge 1 - 1/\mathrm{poly}(k) \ .$$

We then take the union bound over the $O(k^2)$ edges in F and the $O(k)$ vertices in U. The fact that $|F| = O(k^2)$ and $|U| = O(k)$ follows from the promises $\mathrm{match}(G) \le k$ and $\mathrm{vc}(G) \le 2k$. In particular, the induced graph on $V \setminus U$ has a matching of size $\Omega(|F|/k)$ since the maximum degree is $O(k)$ and this size is at most k. Since all vertices in U must be in the minimum vertex cover, $|U| \le 2k$.

To prove $\mathbb{P}\left[\boldsymbol{e \in H}\right] \ge \boldsymbol{1/2}$**.** Let the endpoints of e be x and y. We define a set of vertices A such that e is the unique edge that remains if all vertices in A are removed from the graph:

$$A = (\mathrm{VC}(G) \cup \Gamma(x) \cup \Gamma(y)) \setminus \{x, y\} \quad \text{where } \Gamma(\cdot) \text{ denotes the set of neighbors of a vertex} \ .$$

The removal of $\mathrm{VC}(G) \setminus \{x, y\}$ ensures all remaining edges are incident to either x or y. The subsequent removal of $(\Gamma(x) \cup \Gamma(y)) \setminus \{x, y\}$ ensures the unique remaining edge is xy as claimed.

Consider the hash function $c : [n] \to [b]$ that defined H where $b = 1000k$. Observe that if all the vertices in A receive colors that are different than $c(x)$ and $c(y)$, then xy is the unique $\{c(x), c(y)\}$-colored edge and hence is definitely sampled. Since $b = 1000k$ and $|A| \leq 2k + 10k + 10k = 22k$,

$$\mathbb{P}\left[xy \in H\right] \geq 1 - \mathbb{P}\left[\exists a \in A : c(a) = c(x)\right] - \mathbb{P}\left[\exists a \in A : c(a) = c(y)\right] \geq 1 - 2|A|/b > 1/2 \ .$$

To prove $\mathbb{P}\left[\deg_{H}(u) \geq 5k\right] \geq 1/2$. Let N_u be an arbitrary set of $10k$ neighbors of u and $A = \mathrm{VC}(G) \setminus \{u\}$. If $c(u) \notin c(A)$ and there exist different colors $c_1, \ldots, c_{5k}$ such that each $c_i \in c(N_u) \setminus c(A)$ then the algorithm returns at least $5k$ edges incident to u in H. This follows since every edge not incident to u has at least one vertex in A. Hence, every $\{c_i, c(u)\}$-colored edge is incident to u and is distinct from every $\{c_j, c(u)\}$-colored edge.

Observe that $\mathbb{P}\left[c(u) \in c(A)\right] \leq 2k/b$. By appealing to Lemma 2, with probability at least $3/4$, there are at least $6k$ colors used to color the vertices N_u. Of these colors, at least $5k$ are colored differently from vertices in A. Hence we find $5k$ edges incident to u with probability at least $3/4 - 2k/b \geq 1/2$. □

Extension to Weighted Matching. We now extend the result of the previous section to the weighted case. The following lemma shows that it is possible to remove an edge uv from a graph without changing the weight of the maximum weighted matching, if u and v satisfy certain properties.

Lemma 7. *Let $G = (V, E)$ be a weighted graph and let $G' = (V, E')$ be a subgraph with the property:*

$$\forall uv \in E \setminus E' \ , \ \deg_{G'}^{w(uv)}(u) \geq 5k \ \textit{ or } \ \deg_{G'}^{w(uv)}(v) \geq 5k \ ,$$

where $\deg_G^w(u)$ is the number of edges incident to u in G with weight w. Then, $\mathrm{match}(G) = \mathrm{match}(G')$.

Consider a weighted graph G and let $G' \sim \mathsf{Sample}_{1000k,2,O(\log k)}$. For each weight w, let G_w and G'_w denote the subgraphs consisting of edges with weight exactly w. By applying the analysis of the previous section to G_w and G'_w we may conclude that G' satisfies the properties of the above lemma. Hence, $\mathrm{match}(G) = \mathrm{match}(G')$. To reduce the dependence on the number of distinct weights in Theorem 3, we may first round each weight to the nearest power of $(1 + \epsilon)$ at the cost of incurring a $(1 + \epsilon)$ factor error. If W is the ratio of the max weight to min weight, there are $O(\epsilon^{-1} \log W)$ distinct weights after the rounding.

3.2 Finding Minimum Hitting Set Exactly

In this section we present exact results for hitting set and hypergraph matching. Let $\mathrm{hs}(G)$ denote the cardinality of the minimum hitting set of G. Throughout the section, let G be a hypergraph where each edge has size exactly d and $\mathrm{hs}(G) \leq k$. Throughout this section we assume d is a constant.

Intuition and Preliminaries. Given that the hitting set problem is a generalization of the vertex cover problem, naturally some of the ideas in this section build upon ideas from the previous section. However, the combinatorial structure we need to analyze for our sampling result goes beyond what is typically needed when extending vertex cover kernalization results to hitting sets. We first need to review a basic definition and result about “sunflower” set systems.

Lemma 8 (Sunflower Lemma [19]). *Let $\mathcal{F}$ be a collection of subsets of* $[n]$. *Then $A_1, \ldots, A_s \in \mathcal{F}$ is an* s-sunflower *if $A_i \cap A_j = C$ for all $1 \leq i < j \leq s$. We refer to C as the* core *of the sunflower and $A_i \setminus C$ as the* petals. *If each set in $\mathcal{F}$ has size at most d and $|\mathcal{F}| > d!k^d$, then $\mathcal{F}$ contains a $(k + 1)$-sunflower.*

Let $s_G(C)$ denote the number of petals in a maximum sunflower in the graph G with core C. We say a core is *large* if $s_G(C) > ak$ for some large constant a and *significant* if $s_G(C) > k$. Define the sets:

- $U = \{C \subseteq V \mid s_G(C) > ak\}$ is the set of large cores.
- $F = \{D \in E \mid \forall C \in U, C \not\subseteq D\}$ is the set of edges that do not include a large core.
- $U' = \{C \in U \mid \forall C' \subset C, s_G(C') \leq k\}$ is the set of large cores that do not contain significant cores.

The sets U and F play a similar role to the sets of the same name in the previous section. For example, if $d = 2$, then a large core corresponds to a high degree vertex. However, the set U' has no corresponding notion when $d = 2$ because a high degree vertex cannot contain another high degree vertex. The following (rather technical) lemma will play a crucial role when dealing with cores that are subsets of other cores in U' or of edges in F. It shows that if a core C is contained in a set D, then the set of edges that intersect D at C has a hitting set that a) does not include vertices in C and b) has small size assuming $s_G(C)$ is small.

Lemma 9. *For any two sets of vertices C, D, where $C \subseteq D$, define*

$$M_{C,D} = \{D' \setminus C \mid D' \in E, D \cap D' = C\} .$$

Then $\operatorname{hs}(M_{C,D}) \leq s_G(C)d$. See Figure 1 in the Appendix for an example.

Hitting Set. For the rest of this section we let $G' = (V, E') \sim \mathsf{Sample}_{b,d,r}(G)$ where $b = O(k)$, d is the cardinality of the hyperedges, and $r = O(\log k)$. It will also be convenient to use the notation $\operatorname{HS}(\mathcal{S})$ to denote a minimum hitting set of a collection of sets $\mathcal{S}$, i.e., $\operatorname{hs}(\mathcal{S}) = |\operatorname{HS}(\mathcal{S})|$.

Theorem 10. *Suppose $\operatorname{hs}(G) \leq k$. With probability $1 - 1/\operatorname{poly}(k)$, $\operatorname{hs}(G') = \operatorname{hs}(G)$.*

Proof. For each significant core C there has to be at least one vertex from the hitting set in C. Since all large cores are significant, $\operatorname{hs}(G) = \operatorname{hs}(U \cup F)$. If $C \in U$ has a subset C' such that $s_G(C') > k$, then there is at least one vertex from the hitting set in C' and this vertex also hits C. Thus, $\operatorname{hs}(G) = \operatorname{hs}(U' \cup F)$. By Lemma 11, the set of significant cores in G' is a superset of U' with high probability. By Lemma 12, every edge in F is in G' with high probability. □

Lemma 11. $\mathbb{P}[s_{G'}(C) > k$ *for all* $C \in U'] \geq 1 - 1/\operatorname{poly}(k)$.

Proof. Fix an arbitrary core $C \in U'$. Consider $H \sim \mathsf{Sample}_{b,d,1}$ and let $c : [n] \to [b]$ be the coloring that defined H. We need to identify $k + 1$ sets of colors $S_1, S_2, \ldots S_{k+1} \subset [b]$ each of size d, such that any set of edges $D_1, D_2, \ldots, D_{k+1}$ where D_i is S_i-colored forms a sunflower of size $k + 1$ on core C. In order for this to hold, the color sets have to satisfy the following three properties:

1. All edges that are S_i-colored contain C.
2. There is at least one S_i-colored edge.
3. If D is S_i-colored and D' is S_j-colored then $(D \setminus C) \cap (D' \setminus C) = \emptyset$.

In what follows, we first define a set $\mathcal{F} = \{S_1, S_2, \ldots\}$ that satisfies the above properties. We then argue that $|\mathcal{F}| \geq k + 1$ with probability at least $1/2$. By repeating the process $O(\log k)$ times will ensure that such a family exists with high probability. The lemma follows by taking the union bound over all $C \in U'$ since $|U'| = O(k^{d-1})$ by Lemma 18.

Property 1. We first define a set of vertices A such that any edge that does not intersect A must include C. Then, for any $S \subset [b]$ that is disjoint from $c(A)$, we may infer that all S-colored edges contain C. This follows since if $S = c(D)$ for some edge D, then $c(D) \cap c(A) = \emptyset$ which implies that $D \cap A = \emptyset$, and so $C \subseteq D$. Let

$$A = (\mathrm{HS}(G) \setminus C) \cup \left(\bigcup_{C' \subset C} \mathrm{HS}(M_{C',C}) \right) .$$

All edges that do not intersect $\mathrm{HS}(G) \setminus C$ must intersect with C. But all edges that intersect with only a subset of C, say C', must intersect with $\mathrm{HS}(M_{C',C})$. Hence A has the claimed property. We will say that C is a *good core* if $c(C) \cap c(A) = \emptyset$ and $|c(C)| = |C|$.

Property 2. Next, let $\mathcal{P}$ be a set of petals in a sunflower with core C that do not intersect with A. We may choose a set of $|\mathcal{P}| = ak - |A|$ such petals. For each $P \in \mathcal{P}$, define the set:

$$A_P = A \cup C \cup \left(\bigcup_{Q \in \mathcal{P} \setminus P} Q \right) .$$

If C is a good core, let $\mathcal{P}'$ contain all $P \in \mathcal{P}$ such that $c(P) \cap c(A_P) = \emptyset$ and $|c(P)| = |P|$. If C is not a good core, let $\mathcal{P}' = \emptyset$. Then the family $\mathcal{F} = \{c(P \cup C)\}_{P \in \mathcal{P}'}$ satisfies Properties 1 and 2. Note that no two petals in $\mathcal{P}'$ share the same color and hence $|\mathcal{F}| = |\mathcal{P}'|$ assuming C is a good core.

Property 3. Assume C is a good core since otherwise $\mathcal{F} = \emptyset$ and Property 3 is trivially satisisfied. Let $S_1, S_2 \in \mathcal{F}$ and suppose $S_1 = c(C \cup P_1)$ and $S_2 = c(C \cup P_2)$ for some $P_1, P_2 \in \mathcal{P}'$. Suppse edges $C \cup Q_1$ and $C \cup Q_2$ are S_1-colored and S_2-colored respectively. Then $c(Q_1) = c(P_1)$ and $c(Q_2) = c(P_2)$ because $|c(C)| = |C|$, $|c(P_1)| = |P_1|$, $|c(P_2)| = |P_2|$, and all edges have the same cardinality. But $c(P_1) \cap c(P_2) = \emptyset$ implies $c(Q_1) \cap c(Q_2) = \emptyset$ and so $Q_1 \cap Q_2 = \emptyset$ as required.

Size of $\mathcal{F}$. We need to show that $|\mathcal{P}'| \geq (k+1)$ with probability $1/2$. Recall that $c : V \to [b]$ is chosen randomly from a family of pairwise independent hash functions and suppose $b = 8 \max(d|A| + d^2, d|A_P| + d^2)$. Note that $b = O(k)$ since, by appealing to Lemma 9,

$$|A| \leq |A_P| \leq |A| + |C| + d|\mathcal{P}| \leq \mathrm{hs}(G) + \sum_{C' \subset C} \mathrm{hs}(M_{C,C'}) + |C| + d|\mathcal{P}| \leq k + 2^d dk + d + dak = O(k) .$$

Then, $\mathbb{P}\left[C \text{ is not a good core}\right] = \mathbb{P}\left[c(C) \cap c(A) \neq \emptyset \text{ or } |c(C)| \neq |C|\right] \leq (d|A| + d^2)/b \leq 1/8$. For each $P \in \mathcal{P}$, let $X_P = 1$ if $P \notin \mathcal{P}'$ or C is not a good core. Let $X_P = 0$ otherwise. Then

$$\mathbb{E}\left[\sum X_P\right] \leq |\mathcal{P}| \left(d|A_P| + d^2)/b + 1/8\right) \leq |\mathcal{P}|/4 ,$$

and so $\mathbb{P}\left[\sum X_P \geq |\mathcal{P}|/2\right] \leq 1/2$ by the Markov inequality. Hence, $|\mathcal{P}'| = |\mathcal{P}| - \sum X_P \geq |\mathcal{P}|/2 = ak/2 - |A|/2 \geq k+1$ for sufficiently large a with probability at least $1/2$.

□

Lemma 12. $\mathbb{P}\left[F \subseteq E'\right] \geq 1 - 1/\mathrm{poly}(k)$.

Proof. Pick an arbitrary edge $D \in F$. Consider $H \sim \mathsf{Sample}_{b,d,1}$ and let $c : [n] \to [b]$ be the coloring that defined H. It suffices to show that there is a unique edge that is $c(D)$-colored since then D is necessarily an edge in H. It suffices to show that this is the case with probability at least $1/2$ because repeating the process $O(\log k)$ times will ensure that such a family exists with high probability. The result then follows by taking the union bound over all $D \in F$ since $|F| = O(k^d)$ by Lemma 18.

Let $S = c(D)$. We first define a set A of vertices such that the only edge that is disjoint from A is D. It follows that D is the unique S-colored edge if $S \cap c(A) = \emptyset$, since every other edge intersects A and hence must share a color with it. We define A as follows:

$$A = (\mathrm{HS}(G) \setminus D) \cup \left(\bigcup_{C \subset D} \mathrm{HS}(M_{C,D}) \right) .$$

Note D itself is disjoint from A since each $\mathrm{HS}(M_{C,D})$ does not include vertices from D. If an edge is disjoint from $(\mathrm{HS}(G) \setminus D)$ then it must intersect D. Suppose there exists an edge D' such that $D \cap D' = C \neq D$, then D' intersects $\mathrm{HS}(M_{C,D})$. Hence, the only edge that is disjoint from A includes the vertices in D and so is equal to D on the assumption that all edges have the same number of vertices.

It remains to show that $S \cap c(A) = \emptyset$ with probability at least $1/2$. If $b \geq 2d|A|$ then we have

$$\mathbb{P}\left[S \cap c(A) = \emptyset\right] \geq 1 - d|A|/b \geq 1/2 \, .$$

Finally, note that $b = O(k)$ since $|A| \leq \mathrm{hs}(G) + \sum_{C \subset D} \mathrm{hs}(M_{C,D}) \leq k + 2^d akd = O(k)$ by appealing to Lemma 9 and using the fact that $s_G(C) \leq ak$ for all $C \subset D$ since $D \in F$. □

A result for hypergraph matching follows along similar lines.

Theorem 13. *Suppose* $\mathrm{match}(G) \leq k' = k/d$. *With probability* $1 - 1/\mathrm{poly}(k)$, $\mathrm{match}(G') = \mathrm{match}(G)$.

4 Sampling Kernels for Subgraph Search Problems

We extend our parameterized results to a class of problems where the objective is to search for a subgraph H of $G(V, E)$ which satisfies some property $\mathcal{P}$. In the parametrized setting, we typically search for the largest H which satisfies this property, subject to the promise that the size of any H satisfying $\mathcal{P}$ is at most k. For concreteness, we assume the size is captured by the number of vertices in H, and our objective is to find a maximum cardinality satisfying subgraph. The sampling primitive $\mathsf{Sample}_{b,2,1}$ can be used here when $\mathcal{P}$ is preserved under vertex contraction: if G' is a vertex contraction of G, then any subgraph H of G' satisfying $\mathcal{P}$ also satisfies $\mathcal{P}$ for G (with vertices suitably remapped). Here, the vertex contraction of vertices u and v creates a new vertex whose neighbors are $\Gamma(u) \cup \Gamma(v)$. Many well-studied problems posess the required structure, including:

— b-matching, to find a (maximum cardinality) subgraph H of G such that the degree of each vertex in H is at most b. Hence, the standard notion of matching in Section 2 is equivalent to 1-matching.

— k-colorable subgraph, to find a subgraph H that is k-colorable. The maximum cardinality 2-colorable subgraph forms a max-cut, and more generally the maximum cardinality k-colorable subgraph is a max k-cut.

— other maximum subgraph problems, such as to find the largest subgraph that is a forest, has at least c connected components, or is a collection of vertex disjoint paths.

Theorem 14. *Let $\mathcal{P}$ be a graph property preserved under vertex contraction. Suppose that the number of vertices in some optimum solution* $\mathrm{opt}(G)$ *is at most k. Let $G' \sim \mathsf{Sample}_{4k^2,2,1}(G)$. With constant probability, we can compute a solution H for $\mathcal{P}$ from G' that achieves $|H| = |\mathrm{opt}(G)|$.*

Note that the generality of the result comes at the cost of increasing the number of colors, and hence the space of the stream algorithms. To generalize the result to the weighted case (e.g., where the objective is to find the subgraph satisfying $\mathcal{P}$ with the greatest total weight), we take the approach used in Section 3.1. We perform the sampling in parallel for each distinct weight value, and then round each edge weight to the closest power of $(1 + \epsilon)$ to reduce the number of weight classes to $O(\epsilon^{-1} \log W)$, with a loss factor of $(1 + \epsilon)$.

5 Approximating Large Matchings

The problem of *approximating* large matchings in the dynamic graph stream model has seen a flurry of recent activity. Four sets of related results were submitted to arXiv almost simultaneously [9, 11, 12, 36] (including a version of this paper). Assadi et al. [9] present a different α-approximation algorithm for maximum matching that uses the same space as our algorithm (which they show is optimal). Konrad [36] proves slightly weaker bounds. Bury and Schwiegelshohn [11] present an algorithm for estimating the *size* of the maximum matching in graphs of bounded arboricity. Our second algorithm in this section is similar (the difference is that we can find the edges of an exact matching when it is small whereas they approximate the cardinality in this case by guessing and verifying the rank of a related matrix), so we relegate it to the appendix.

Intuition and Preliminaries. Given a hash function $c : V \to [b]$, we say an edge uv is colored i if $c(u) = c(v) = i$. If the endpoints have different colors, we say the edge is *uncolored*. The basic idea behind our algorithm is to repeatedly sample a set of colored edges with distinct colors. Note that a set of edges colored with different colors is a matching. We use the edges in this matching to augment the matching already constructed from previous rounds. In this section we require the hash functions to be $O(k)$-wise independent and, in the context of dynamic data streams, this will increase the update time by a $O(k)$ factor.

Theorem 15. *Suppose* $\operatorname{match}(G) \geq k$. *For any* $1 \leq \alpha \leq \sqrt{k}$ *and* $0 < \epsilon \leq 1$, *with probability* $1 - 1/\operatorname{poly}(k)$,

$$\operatorname{match}(G') \geq \left(\frac{1-\epsilon}{2\alpha}\right) \cdot k \ ,$$

where $G' \sim \mathsf{Sample}_{2k/\alpha,1,r}$ *where* $r = O(k\alpha^{-2}\epsilon^{-2}\log k)$.

Proof. Let $H_1, \ldots, H_r \sim \mathsf{Sample}_{2k/\alpha,1,1}$ and let G' be the union of these graphs. Consider the greedy matching M_r where $M_0 = \emptyset$ and for $t \geq 1$, M_t is the union of M_{t-1} and additional edges from H_t. We will show that if M_{t-1} is small, then we can find many edges in H_t that can be used to augment M_{t-1}.

Consider H_t and suppose $|M_{t-1}| < \frac{1-\epsilon}{2\alpha} \cdot k$. Let $c : V \to [b]$ be the hash-function used to define H_t where $b = \frac{2k}{\alpha}$. Let U be the set of colors that are not used to color the endpoints of M_{t-1}, i.e.,

$$U = \{c \in [b] : \text{ there does not exist a matched vertex } u \text{ in } M_{t-1} \text{ with } c(u) = c\} \ .$$

and note that $|U| \geq b - 2|M_{t-1}| \geq \frac{k}{\alpha}$. For each $c \in U$, define the indicator variable X_c where $X_c = 1$ if there exists an edge uv with $c(u) = c(v) = c$. We will find $X = \sum_{c \in U} X_c$ edges to add to the matching.

Since $\operatorname{match}(G) \geq k$, there exists a set $k - 2|M_{t-1}| > k\epsilon$ vertex disjoint edges that can be added to M_{t-1}. Let $p = \frac{\alpha}{2k}$ and observe that $\mathbb{E}\left[X_c\right] \geq k\epsilon p^2 - \binom{k\epsilon}{2}p^4 > k\epsilon p^2/2 = \epsilon \cdot \frac{\alpha^2}{8k}$. Therefore, $\mathbb{E}\left[X\right] \geq (\frac{k}{\alpha}) \cdot \epsilon \cdot \frac{\alpha^2}{8k} = \frac{\epsilon\alpha}{8}$. Since X_c and $X_{c'}$ are negative correlated, $\mathbb{P}\left[X \geq E[X]/2\right] \geq 1 - \exp\left(-\Omega\left(\epsilon\alpha\right)\right) \geq \Omega(\epsilon)$. Hence, with each repetition we may increase the size of the matching by at least $\epsilon\alpha/2$ with probability $\Omega(\epsilon)$. After $O(k\alpha^{-2}\epsilon^{-2}\log k)$ repetitions the matching has size at least $\frac{1-\epsilon}{2\alpha} \cdot k$. □

By applying Theorem 15 for all $k \in \{1, 2, 4, 8, 16, \ldots\}$ and appealing to Theorem 3, we establish:

Corollary 16. *There exists a* $O(n \operatorname{polylog} n)$*-space algorithm that returns an* $O(n^{1/3})$*-approximation to the size of the maximum matching in the dynamic graph stream model.*

This result generalizes to the weighted case using the Crouch-Stubbs technique [15]. They showed that if we can find a β-approximation to the maximum *cardinality* matching amongst all edges of weight greater than $(1+\epsilon)^i$ for each i, then we can find a $2(1+\epsilon)\beta$-approximation to the maximum weighted matching.

Matchings in Planar and Bounded-Arboricity Graphs. We also provide an algorithm for estimating the size of the matching in a graph of bounded arboricity. Recall that a graph has *arboricity* ν if its edges can be partitioned into at most ν forests. See Appendix B for details.

Theorem 17. *There exists a* $\tilde{O}(\nu\epsilon^{-2}n^{4/5}\log\delta^{-1})$*-space dynamic graph stream algorithm that returns a* $(5\nu+9)(1+\epsilon)^2$ *approximation of* $\operatorname{match}(G)$ *with probability at least* $1-\delta$ *where* ν *is the arboricity of* G.

References

[1] List of open problems in sublinear algorithms: Problem 64. `http://sublinear.info/64`.

[2] F. M. Ablayev. Lower bounds for one-way probabilistic communication complexity and their application to space complexity. *Theor. Comput. Sci.*, 157(2):139–159, 1996.

[3] K. J. Ahn, G. Cormode, S. Guha, A. McGregor, and A. Wirth. Correlation clustering in data streams. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2237–2246, 2015.

[4] K. J. Ahn and S. Guha. Laminar families and metric embeddings: Non-bipartite maximum matching problem in the semi-streaming model. *Manuscript, available at http://arxiv.org/abs/1104.4058*, 2011.

[5] K. J. Ahn and S. Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. In *ICALP (2)*, pages 526–538, 2011.

[6] K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In *Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*, pages 459–467, 2012.

[7] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 5–14, 2012.

[8] K. J. Ahn, S. Guha, and A. McGregor. Spectral sparsification in dynamic graph streams. In *APPROX*, pages 1–10, 2013.

[9] S. Assadi, S. Khanna, Y. Li, and G. Yaroslavtsev. Tight bounds for linear sketches of approximate matchings. *CoRR*, abs/1505.01467, 2015.

[10] S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. E. Tsourakakis. Space and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *STOC*, 2015.

[11] M. Bury and C. Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. *CoRR (To appear at ESA)*, abs/1505.02019, 2015.

[12] R. H. Chitnis, G. Cormode, H. Esfandiari, M. Hajiaghayi, A. McGregor, M. Monemizadeh, and S. Vorotnikova. Kernelization via sampling with applications to dynamic graph streams. *CoRR*, abs/1505.01731, 2015.

[13] R. H. Chitnis, G. Cormode, M. T. Hajiaghayi, and M. Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1234–1251, 2015.

[14] G. Cormode and D. Firmani. A unifying framework for ℓ_0-sampling algorithms. *Distributed and Parallel Databases*, 32(3):315–335, 2014.

[15] M. Crouch and D. S. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, pages 96–104, 2014.

[16] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, New York, 1999.

[17] Y. Emek and A. Rosén. Semi-Streaming Set Cover - (Extended Abstract). In *ICALP*, pages 453–464, 2014.

[18] L. Epstein, A. Levin, J. Mestre, and D. Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM J. Discrete Math.*, 25(3):1251–1265, 2011.

[19] P. Erdos and R. Rado. Intersection theorems for systems of sets. *J. London Math. Soc.*, 35:85–90, 1960.

[20] H. Esfandiari, M. Hajiaghayi, and D. P. Woodruff. Applications of uniform sampling: Densest subgraph and beyond. *CoRR*, abs/1506.04505, 2015.

[21] H. Esfandiari, M. T. Hajiaghayi, V. Liaghat, M. Monemizadeh, and K. Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1217–1233, 2015.

[22] S. Fafianie and S. Kratsch. Streaming kernelization. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 275–286, 2014.

[23] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2):207–216, 2005.

[24] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[25] A. C. Gilbert and P. Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937–947, 2010.

[26] A. Goel, M. Kapralov, and S. Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485, 2012.

[27] A. Goel, M. Kapralov, and I. Post. Single pass sparsification in the streaming model with edge deletions. *CoRR*, abs/1203.4900, 2012.

[28] S. Guha, A. McGregor, and D. Tench. Vertex and hypergraph connectivity in dynamic graph streams. In *PODS*, 2015.

[29] B. V. Halldórsson, M. M. Halldórsson, E. Losievskaja, and M. Szegedy. Streaming algorithms for independent sets. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, pages 641–652, 2010.

[30] H. Jowhari, M. Saglam, and G. Tardos. Tight bounds for L_p samplers, finding duplicates in streams, and related problems. In *Proceedings of the 17th ACM SIGMOD Symposium on Principles of Database Systems (PODS)*, pages 49–58, 2011.

[31] M. Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1679–1697, 2013.

[32] M. Kapralov, S. Khanna, and M. Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 734–751, 2014.

[33] M. Kapralov, Y. T. Lee, C. Musco, C. Musco, and A. Sidford. Single pass spectral sparsification in dynamic streams. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 561–570, 2014.

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

[35] D. Kogan and R. Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 367–376, 2015.

[36] C. Konrad. Maximum matching in turnstile streams. *CoRR (To appear at ESA)*, abs/1505.01460, 2015.

[37] C. Konrad, F. Magniez, and C. Mathieu. Maximum matching in semi-streaming with few passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, pages 231–242, 2012.

[38] C. Konrad and A. Rosén. Approximating semi-matchings in streaming and in two-party communication. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 637–649, 2013.

[39] E. Kushilevitz and N. Nisam. *Commmunication Complexity*. Cambridge University Press, 1997.

[40] K. Kutzkov and R. Pagh. Triangle counting in dynamic graph streams. In *Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings*, pages 306–318, 2014.

[41] A. McGregor. Finding graph matchings in data streams. *APPROX-RANDOM*, pages 170–181, 2005.

[42] A. McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014.

[43] A. McGregor, D. Tench, S. Vorotnikova, and H. Vu. Densest subgraph in dynamic graph streams. In *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milano, Italy, August 24-28, 2014. Proceedings, Part I*, 2015.

[44] S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Now Publishers, 2006.

[45] J. Radhakrishnan and S. Shannigrahi. Streaming algorithms for 2-coloring uniform hypergraphs. In *Algorithms and Data Structures - 12th International Symposium, WADS 2011, New York, NY, USA, August 15-17, 2011. Proceedings*, pages 667–678, 2011.

[46] B. Saha and L. Getoor. On maximum coverage in the streaming model & application to multi-topic blog-watch. In *SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*, pages 697–708, 2009.

[47] H. Sun. Counting hypergraphs in data streams. *CoRR*, abs/1304.7456, 2013.

[48] M. Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62(1-2):1–20, 2012.

A Omitted Proofs

Proof of Lemma 2. Let $b = 4\epsilon r$. For a vertex $u \in U$, let I_u be the indicator random variable that equals one if there exists $u' \in U \setminus \{u\}$ such that $c(u) = c(u')$. Since c is pairwise independent,

$$\mathbb{P}\left[I_u\right] \leq \sum_{u' \in U \setminus \{u\}} \mathbb{P}\left[c(u) = c(u')\right] = \sum_{u' \in U \setminus \{u\}} 1/b < r/b = \epsilon/4 \ .$$

Let $I = \sum_{u \in U} I_u$ and note that $\mathbb{E}\left[I\right] \leq \epsilon r/4$. Then Markov's inequality implies $\mathbb{P}\left[I \geq \epsilon r\right] \leq 1/4$. □

Proof of Theorem 3. To sample a graph from $\mathsf{Sample}_{b,q,r}$ we simply sample r graphs from $\mathsf{Sample}_{b,q,1}$ in parallel. To draw a sample from $\mathsf{Sample}_{b,q,1}$, we employ one instance of an ℓ_0-sampling primitive for each of the $O(b^q)$ edge colorings [14, 30]: Given a dynamic graph stream, the behavior of an ℓ_0-sampler algorithm is defined as follows: It returns FAIL with probability at most δ and otherwise, it returns an edge chosen uniformly at random amongst the edges that have been inserted and not deleted. If there are no such edges, the ℓ_0-sampler returns NULL. The ℓ_0-sampling primitive can be implemented using $O(\log^2 n \log \delta^{-1})$ bits of space and $O(\operatorname{polylog} n)$ update time. In some cases, we can make use of simpler deterministic data structures. For Theorem 4, we can replace the ℓ_0 sampler with a counter and the exclusive-or of all the edge identifiers, since we only require to recover edges when they are unique within their color class. For Theorem 14, we only require a counter. In both cases, the space cost is reduced to $O(\log n)$.

At the start of the stream we choose a pairwise independent hash function $c : V \to [b]$. For each weight w and subset $S \subseteq [b]$ of size q, this hash function defines a sub-stream corresponding to the S-colored edges of weight w. We then use ℓ_0-sampling on each sub-stream to select a random edge from E_S. □

Proof of Lemma 5. We first argue that $\operatorname{vc}(G') = \operatorname{vc}(G)$. Since the vertex cover of G is of size at most $2k$, every vertex in U must be in the vertex cover of both G and G' since the degrees of such vertices in both graphs are strictly greater than $2k$. This follows because if a vertex in U was not in the minimum vertex cover then all its neighbors need to be in the vertex cover.

We next argue that $\operatorname{match}(G') = \operatorname{match}(G)$. If property (1) is satisfied then G' contains a matching of size $\operatorname{match}(F) + |U| \geq \operatorname{match}(G)$ since we may choose the optimum matching in F and then still be able to match every vertex in U. This follows because the optimum matching in F "consumes" at most $2k$ potential endpoints, since $\operatorname{match}(G) \leq k$. Hence, each of the (at most $2k$) vertices in U can still be matched to $3k$ possible vertices. □

Proof of Lemma 7. Let $E \setminus E' = \{e_1, e_2, \ldots e_t\}$ and let G'_i be the graph formed by removing $\{e_1, \ldots, e_i\}$ from G. So $G'_0 = G$ and $G'_t = G'$. For the sake of contradiction, suppose $\operatorname{match}(G) > \operatorname{match}(G')$ and let r be the minimal value such that $\operatorname{match}(G) > \operatorname{match}(G'_r)$.

By the minimality of r, $\operatorname{match}(G) = \operatorname{match}(G'_{r-1})$. Consider the maximum weight matching M in G'_{r-1}. If $e_r \notin M$ then $\operatorname{match}(G) = \operatorname{match}(G'_{r-1}) = \operatorname{match}(G'_r)$ and we have a contradiction. If $e_r \in M$, let u, v be the endpoints of e_r and the weight of e_r be w. Without loss of generality $\deg^w_{G'_r}(u) \geq d^w_{G'}(u) \geq 5k$. Hence, there exists edge ux of weight w in G'_r where x is not an endpoint in M. Therefore, the matching $(M \setminus \{e_r\}) \cup \{ux\}$ is contained in G'_r and has the same weight as M. Hence, $\operatorname{match}(G) = \operatorname{match}(G'_{r-1}) = \operatorname{match}(G'_r)$ and we again have a contradiction. □

Proof of Lemma 9. Consider the size of minimum hitting set of $M_{C,D}$. If $\operatorname{hs}(M_{C,D}) > s_G(C)d$, then $M_{C,D}$ has a matching of size greater than $s_G(C)$. This matching together with the set C forms a sunflower with core C and over $s_G(C)$ petals, which contradicts the assumption. Therefore, $\operatorname{hs}(M_{C,D}) \leq s_G(C)d$ as claimed. □

Proof of Theorem 13. $\mathrm{hs}(G) \leq dk' = k$. Let M be the matching. $F \cap M$ is preserved in G'. Consider an edge $D \in M$ such that $C \subseteq D$ for some $C \in U$. Then in G' we can find (by Lemma 11) at least $k+1$ petals in a sunflower with core either C itself or some $C' \subset C$. At most k of those intersect $M \setminus \{D\}$. Therefore, there is still at least one edge we can pick for the matching. □

Proof of Corollary 16. For $1 \leq i \leq \log n$, let $G'_i \sim \mathsf{Sample}_{b,1,r}$ where $r = O(2^i \alpha^{-2} \log k)$ and $b = 2^{i+1}/\alpha$. These graphs can be generated in $\tilde{O}(n^2\alpha^{-3})$ space. For some i, $2^i \leq \mathrm{match}(G) < 2^{i+1}$ and hence $\mathrm{match}(G'_i) = \Omega(\mathrm{match}(G)/\alpha)$. □

Proof of Theorem 14. We construct a *contracted graph* G'' from G' based on the color classes used in the Sample operator: we contract all vertices that are assigned the same color by the hash function $c()$. Fix an optimum solution $\mathrm{opt}(G)$ with at most k vertices. Lemma 2 shows that for $b = 4k^2$, all vertices involved in *opt*(G) are hashed into distinct color values. Hence, the subgraph $\mathrm{opt}(G)$ is a subgraph of G'': for any edge $e = (u, v) \in \mathrm{opt}(G)$, the edge itself was sampled from the data structure, or else a different edge with the same color values was sampled, and so can be used interchangeably in G''. Hence, (the remapped form of) $\mathrm{opt}(G)$ persists in G''. By the vertex contraction property of $\mathcal{P}$, this means that a maximum cardinality solution for $\mathcal{P}$ in G'' is a maximum cardinality solution in G.

Note that for this application of the subgraph sampling primitive, it suffices to implement the sampling data structure with a counter for each pair of colors: any non-zero count corresponds to an edge in G''. □

A.1 Omitted Lemmas

Lemma 18. $|F| = O(k^d)$ *and* $|U'| = O(k^{d-1})$

Proof. For the sake of contradiction assume $|F| > d!(ak)^d$. Then, by the Sunflower Lemma, F contains a $(ak+1)$-sunflower. If the core of this sunflower is empty, F has a matching of size $(ak+1)$ and therefore cannot have a hitting set of size at most k. If the sunflower has a non-empty core C, then some edge $D \in F$ contains C, which contradicts the definition of F. Therefore, $|F| \leq d!(ak)^d$.

To prove $|U'| \leq (d-1)!k^{d-1}$, first note that $|C'| \leq d-1$ for all $C' \in U'$. For the sake of contradiction assume that $|U'| > (d-1)!k^{d-1}$. Then, by the Sunflower Lemma again, U' contains a $(k+1)$-sunflower. Note that it is a sunflower of cores, not hypergraph edges. Let $C_1, C_2, \ldots, C_{k+1}$ be the sets in the sunflower. Each of these sets has to contain at least one vertex of the minimum hitting set. Therefore, if $C_1, C_2, \ldots, C_{k+1}$ are disjoint (i.e., the core of the sunflower is empty), U' has a matching of size $(k+1)$ and cannot have a hitting set of size at most k. If the sunflower has a non-empty core C^*, we will show that union of the maximum sunflowers with cores $C_1, C_2, \ldots, C_{k+1}$ contains a sunflower with $k+1$ edges with core $C^* \subset C_1 \in U'$. This contradicts the definition of U' and therefore $|U'| \leq (d-1)!k^{d-1} = O(k^{d-1})$. To construct the sunflower on C^*, for $i = 1, \ldots, k+1$, we pick an edge D_i in the maximum sunflower with core C_i such that $D_i \cap C_j = C^*$ for $j \neq i$ and $D_i \cap D_j = C^*$ for $j < i$. This is possible if a is sufficiently large. □

B Matchings in Planar and Bounded-Arboricity Graphs

In this section, we present an algorithm for estimating the size of the matching in a graph of bounded arboricity. Recall that a graph has *arboricity* ν if its edges can be partitioned into at most ν forests. In particular, it can be shown that a planar graph has arboricity at most 3. We will make repeated use of the fact that the average degree of every subgraph of a graph with arboricity ν is at most 2ν.

Our algorithm is based on an insertion-only streaming algorithm due to Esfandiari et al. [21]. They first proved upper and lower bounds on the size of the maximum matching in a graph of arboricity ν.

Lemma 19 (Esfandiari et al. [21])**.** *For any graph G with arboricity ν, define a vertex to be* heavy *if its degree is at least $2\nu + 3$ and define an edge to be* shallow *if it is not incident to a heavy vertex. Then,*

$$\frac{\max\{h, s\}}{2.5\nu + 4.5} \leq \text{match}(G) \leq 2\max\{h, s\} \ .$$

where h is the number of heavy vertices and s is the number of shallow edges.

To estimate $\max\{h, s\}$, Esfandiari et al. sampled a set of vertices Z and (a) computed the exact degree of these vertices, then (b) found the set of all edges in the induced subgraph on these vertices. The fraction of heavy vertices in Z and shallow edges in the induced graph are then used to estimate h and s. By choosing the size of Z appropriately, they showed that the resulting estimate was sufficiently accurate on the assumption that $\max\{h, s\}$ is large. In the case where $\max\{h, s\}$ is small, the maximum matching is also small and hence a maximal matching could be constructed in small space using a greedy algorithm.

Algorithm for Dynamic Graph Streams. In the dynamic graph stream model, it is not possible to construct a maximal matching. However, we may instead use the algorithm of Theorem 4 to find the exact size of the maximum matching. Furthermore we can still recover the induced subgraph on sampled vertices Z via a sparse recovery sketch [25]. This can be done space-efficiently because the number of edges is at most $2\nu|Z|$. Lastly, rather than fixing the size of Z, we consider sampling each vertex independently with a fixed probability as this simplifies the analysis significantly. The resulting algorithm is as follows:

1. Invoke algorithm of Theorem 4 for $k = 2n^{2/5}$ and let r be the reported matching size.
2. In parallel, sample vertices with probability $p = 8\epsilon^{-2}n^{-1/5}$ and let Z be the set of sampled vertices. Find the degrees of vertices in Z in G and maintain a $2\nu|Z|$-sparse recovery sketch of the edges in the induced graph on Z. Let s_Z be the number of shallow edges in the induced graph on Z and let s_Z be the number of heavy vertices in Z. Return $\max\{r, h_Z/p, s_Z/p^2\}$.

Analysis. Our analysis relies on the following lemma that shows that $\max\{h_Z/p, s_Z/p^2\}$ is a $1 + \epsilon$ approximation for $\max\{s, h\}$ on the assumption that $\max\{s, h\} \geq n^{2/5}$.

Lemma 20. $\mathbb{P}\left[|\max\{h_Z/p, s_Z/p^2\} - \max\{s, h\}| \leq \epsilon \cdot \max\{n^{2/5}, s, h\}\right] \geq 4/5$.

Proof. First we show s_Z/p^2 is a sufficiently good estimate for s. Let S be the set of shallow edges in G and let E_Z be the set of edges in the induced graph on Z. For each shallow edge $e \in S$, define an indicator random variable X_e where $X_e = 1$ iff $e \in E_Z$ and note that $s_Z = \sum_{e \in S} X_e$. Then,

$$\mathbb{E}\left[s_Z\right] = sp^2 \quad \text{ and } \quad \mathbb{V}\left[s_Z\right] = \sum_{e \in S} \sum_{e' \in S} \mathbb{E}\left[X_e X_{e'}\right] - \mathbb{E}\left[X_e\right] \mathbb{E}\left[X_{e'}\right] \ .$$

Note that

$$\sum_{e' \in S} \mathbb{E}\left[X_e X_{e'}\right] - \mathbb{E}\left[X_e\right] \mathbb{E}\left[X_{e'}\right] = \begin{cases} p^2 - p^4 & \text{if } e = e' \\ p^3 - p^4 & \text{if } e \text{ and } e' \text{ share exactly one endpoint} \\ 0 & \text{if } e \text{ and } e' \text{ share no endpoints} \end{cases} \ .$$

and since there are at most $2\nu + 3$ edges that share an endpoint with a shallow edge,

$$\mathbb{V}\left[s_Z\right] \leq s(p^2 - p^4 + (2\nu + 3)p^3 - p^4) \leq 2sp^2$$

on the assumption that $(2\nu + 3) \leq 1/p$. We then use Chebyshev's inequality to obtain

$$\mathbb{P}\left[|s_Z - sp^2| \leq p^2\epsilon \cdot \max\{n^{2/5}, s\}\right] \leq \frac{2sp^2}{(p^2\epsilon \cdot \max\{n^{2/5}, s\})^2} \leq 9/10\ . \tag{2}$$

Next we show that h_Z/p is a sufficiently good estimate for h. Let H denote the set of h heavy vertices in G and define an indicator random variable Y_v for each $v \in H$, where $Y_v = 1$ iff $v \in Z$. Note that $h_Z = \sum_{v \in H} Y_v$ and $\mathbb{E}[h_Z] = hp$. Then, by an application of the Chernoff-Hoeffding bound,

$$\mathbb{P}\left[|h_Z - hp| \geq \epsilon p \max\{h, n^{2/5}\}\right] \leq \exp(-\epsilon^2 p n^{2/5}/3) \leq 9/10\ . \tag{3}$$

Therefore, it follows from Eq. 2 and 3 that

$$\mathbb{P}\left[|\max\{h_Z/p, s_Z/p^2\} - \max\{s, h\}| \leq \epsilon \cdot \max\{n^{2/5}, s, h\}\right] \geq 4/5\ .$$

□

Theorem 21. *There exists a $\tilde{O}(\nu\epsilon^{-2}n^{4/5}\log\delta^{-1})$-space dynamic graph stream algorithm that returns a $(5\nu + 9)(1 + \epsilon)^2$ approximation of $\mathrm{match}(G)$ with probability at least $1 - \delta$ where ν is the arboricity of G.*

Proof. To argue the approximation factor, first suppose $\mathrm{match}(G) \leq 2n^{2/5}$. In this case $r = \mathrm{match}(G)$ and $\max\{s, h\} \leq (2.5\nu + 4.5)\,\mathrm{match}(G)$ by appealing to Lemma 19. Hence,

$$\mathrm{match}(G) \leq \max\{r, h_Z/p, s_Z/p^2\} \leq (2.5\nu + 4.5)\,\mathrm{match}(G)$$

Next suppose $\mathrm{match}(G) \geq 2n^{2/5}$. In this case, $\max\{s, h\} \geq n^{2/5}$ by Lemma 19. Therefore, by Lemma 20, $\max\{h_Z/p, s_Z/p^2\} = (1 \pm \epsilon)\max\{s, h\}$, and so

$$\frac{\mathrm{match}(G)}{2(1+\epsilon)} \leq \max\{r, h_Z/p, s_Z/p^2\} \leq (1+\epsilon)\max\{s, h\} \leq (1+\epsilon)(2.5\nu + 4.5)\,\mathrm{match}(G)$$

To argue the space bound, recall that the algorithm used in Theorem 4 requires $\tilde{O}(n^{4/5})$ space. Note that $|Z| \leq 2np = \tilde{O}(\epsilon^{-2}n^{4/5})$ with high probability. Hence, to sample the vertices Z and maintain a $2\nu|Z|$-sparse recovery data structure requires $\tilde{O}(n^{4/5}\nu)$ space. □

C Figures

See Figure 1.

D Lower Bounds

D.1 Matching and Hitting Set Lower Bounds

The following theorem establishes that the space-use of our matching, vertex cover, hitting set, and hyper matching algorithms is optimal up to logarithmic factors.

Theorem 22. *Any (randomized) parametrized streaming algorithm for the minimum d-hitting set or maximum (hyper)matching problem with parameter k requires $\Omega(k^d)$ space.*

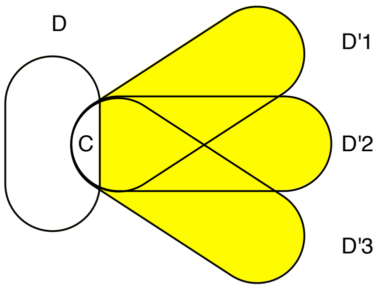

Figure 1: Given sets D'_1, D'_2, D'_3 that intersect set D exactly at C then $M_{C,D}$ consists of the shaded subsets of D'_1, D'_2, and D'_3. Assuming C is non-empty, $\{D'_1, D'_2, D'_3\}$ has a hitting set of size 1 since any vertex in C hits all sets. Lemma 9 bounds the size of the minimum hitting set of $\{D'_1, D'_2, D'_3\}$ that may not include any vertices in C.

Proof. We reduce from the MEMBERSHIP communication problem:

MEMBERSHIP
Input: Alice has a set $X \subseteq [n]$, and Bob has an element $1 \leq x \leq n$.
Question: Bob wants to check whether $x \in X$.

There is a lower bound of $\Omega(n)$ bits of communication from Alice to Bob, even allowing randomization [2].

Let $S = s_1 s_2 ... s_n$ be the characteristic string of X, i.e. a binary string such that $s_i = 1$ iff $i \in X$. Let $k = \sqrt[d]{n}$. Fix a canonical mapping $h : [n] \to [k]^d$. This way we can view an n bit string as an adjacency matrix of a d-partite graph. Construct the following graph G with d vertex partitions $V_1, V_2, ..., V_d$:

- Each partition V_i has dk vertices: for each $j \in [k]$ create vertices $v^*_{i,j}$, $v^1_{i,j}$, $v^2_{i,j}$,..., $v^{d-1}_{i,j}$.
- Alice inserts a hyperedge $(v^*_{1,j_1}, v^*_{2,j_2}, ..., v^*_{d,j_d})$ iff the corresponding bit in the string S is 1, i.e., $s_a = 1$ where $h(a) = (j_1, j_2, ..., j_d)$.
- Let $h(x) = (J_1, J_2, ..., J_d)$. Bob inserts edge $(v^*_{i,j}, v^1_{i,j}, v^2_{i,j}, ..., v^{d-1}_{i,j})$ iff $j \neq J_i$.

Alice runs the hitting set algorithm on the edges she is inserting using space $f(k)$. Then she sends the memory contents of the algorithm to Bob, who finishes running the algorithm on his edges.

The minimum hitting set should include vertices $v^*_{i,j}$ such that $j \neq J_i$. If edge $(v^*_{1,J_1}, v^*_{2,J_2}, ..., v^*_{d,J_d})$ is in the graph, we also need to include one of its vertices. Therefore,

$$x \in X \iff s_x = 1 \iff (v^*_{1,J_1}, v^*_{2,J_2}, ..., v^*_{d,J_d}) \text{ is in } G \iff \text{hs}(G) = dk - d + 1$$

On the other hand,

$$x \notin X \iff s_x = 0 \iff (v^*_{1,J_1}, v^*_{2,J_2}, ..., v^*_{d,J_d}) \text{ is not in } G \iff \text{hs}(G) = dk - d$$

Alice only sends $f(k)$ bits to Bob. Therefore, $f(k) = \Omega(n) = \Omega(k^d)$.

For the lower bound on matching we use the same construction. For each vertex $v^*_{i,j}$ such that $j \neq J_i$ maximum matching should include $(v^*_{i,j}, v^1_{i,j}, v^2_{i,j}, ..., v^{d-1}_{i,j})$. If edge $(v^*_{1,J_1}, v^*_{2,J_2}, ..., v^*_{d,J_d})$ is in the graph, we include it in the matching as well. Therefore,

$$x \in X \iff s_x = 1 \iff (v^*_{1,J_1}, v^*_{2,J_2}, ..., v^*_{d,J_d}) \text{ is in } G \iff \text{match}(G) = dk - d + 1$$

And

$$x \notin X \iff s_x = 0 \iff (v^*_{1,J_1}, v^*_{2,J_2}, ..., v^*_{d,J_d}) \text{ is not in } G \iff \text{match}(G) = dk - d$$

□

D.2 Lower Bounds for Problems considered by Fafianie and Kratsch [22]

Comparison with Lower Bounds for Streaming Kernels: Fafianie and Kratsch [22] introduced the notion of kernelization in the streaming setting as follows:

Definition 23. *A 1-pass streaming kernelization algorithm is receives an input (x, k) and returns a kernel, with the restriction that the space usage of the algorithm is bounded by $p(k) \cdot \log |x|$ for some polynomial p.*

Fafianie and Kratsch [22] gave deterministic lower bounds for several parameterized problems. In particular, they showed that:

- Any 1-pass kernel for EDGE DOMINATING SET(k) requires $\Omega(m)$ bits, where m is the number of edges. However, there is a 2-pass kernel which uses $O(k^3 \cdot \log n)$ bits of local memory and $O(k^2)$ time in each step and returns an equivalent instance of size $O(k^3 \cdot \log k)$.
- The lower bound of $\Omega(m)$ bits for any 1-pass kernel also holds for several other problems such as CLUSTER EDITING(k), CLUSTER DELETION(k), CLUSTER VERTEX DELETION(k), COGRAPH VERTEX DELETION(k), MINIMUM FILL-IN(k), EDGE BIPARTIZATION(k), FEEDBACK VERTEX SET(k), ODD CYCLE TRANSVERSAL(k), TRIANGLE EDGE DELETION(k), TRIANGLE VERTEX DELETION(k), TRIANGLE PACKING(k), s-STAR PACKING(k), BIPARTITE COLORFUL NEIGHBORHOOD(k).
- Any t-pass kernel for CLUSTER EDITING(k) and MINIMUM FILL-IN(k) requires $\Omega(n/t)$ space.

In this section, we give $\Omega(n)$ randomized lower bounds for the space complexity of all the problems considered by Fafianie and Kratsch. In addition, we also consider some other problems such as PATH(k) which were not considered by Fafianie and Kratsch. A simple observation shows that any lower bound for parameterized streaming kernels also transfers for the parameterized streaming algorithms. Thus the results of Fafiane and Kratsch [22] also give lower bounds for the parameterized streaming algorithms for these problems. However, our lower bounds have the following advantage over the results of [22]:

- All our lower bounds also hold for *randomized algorithms*, whereas the kernel lower bounds were for deterministic algorithms.
- With the exception of EDGE DOMINATING SET(k), all our lower bounds also hold for *constant number of passes*.

D.2.1 Lower Bound for EDGE DOMINATING SET

We now show a lower bound for the EDGE DOMINATING SET(k) problem.

Definition 24. *Given a graph $G = (V, E)$ we say that a set of edges $X \subseteq E$ is an edge dominating set if every edge in $E \setminus X$ is incident on some edge of X.*

EDGE DOMINATING SET(k) *Parameter*: k
Input: An undirected graphs G and an integer k
Question: Does there exist an edge dominating set $X \subseteq E$ of size at most k?

Theorem 25. *For the* EDGE DOMINATING SET(k) *problem, any (randomized) streaming algorithm needs $\Omega(n)$ space .*

Proof. Given an instance of MEMBERSHIP, we create a graph G on $n + 2$ vertices as follows. For each $i \in [n]$ we create a vertex v_i. Also add two special vertices a and b. For every $y \in X$, add the edge (a, y). Finally add the edge (b, x).

Now we will show that G has an edge dominating set of size 1 iff MEMBERSHIP answers YES. In the first direction suppose that G has an edge dominating set of size 1. Then it must be the case that $x \in X$: otherwise for a minimum edge dominating set we need one extra edge to dominate the star incident on a, in addition to the edge (b, x) dominating itself. Hence MEMBERSHIP answers YES. In reverse direction, suppose that MEMBERSHIP answers YES. Then the edge (a, x) is clearly an edge dominating set of size 1.

Therefore, any (randomized) streaming algorithm that can determine whether a graph has an edge dominating set of size at most $k = 1$ gives a communication protocol for MEMBERSHIP, and hence requires $\Omega(n)$ space. □

D.2.2 Lower Bound for $\mathcal{G}$-FREE DELETION

Definition 26. *A set of connected graphs $\mathcal{G}$ is* bad *if there is a minimal (under operation of taking subgraphs) graph $H \in \mathcal{G}$ such that H contains at least two distinct edges.*

For any bad set of graphs $\mathcal{G}$, we now show a lower bound for the following general problem:

$\mathcal{G}$-FREE DELETION(k) *Parameter*: k
Input: A bad set of graphs $\mathcal{G}$, an undirected graph $G = (V, E)$ and an integer k
Question: Does there exist a set $X \subseteq V$ such that $G \setminus X$ contains no graph from $\mathcal{G}$?

The reduction from the DISJOINTNESS problem in communication complexity.

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 $\exists\ i \in [n]$ such that $x_i = y_i = 1$.

There is a lower bound of $\Omega(n/p)$ bits of communication between Alice and Bob, allowing p-rounds and randomization [39].

Theorem 27. *For a* bad *set of graphs $\mathcal{G}$, any p-pass (randomized) streaming algorithm for the* $\mathcal{G}$-FREE DELETION *problem needs $\Omega(n/p)$ space .*

Proof. Since $\mathcal{G}$ is a *bad* set of graphs, there is a minimal graph $H \in \mathcal{G}$ which has at least two distinct edges, say e_1 and e_2. Let $H' := H \setminus \{e_1, e_2\}$. Given an instance of DISJOINTNESS, we create a graph G which consists of n disjoint copies say $G_1, G_2, \ldots, G_n$ of H'. For each $i \in [n]$, to the copy G_i of H' we add the edge e_1 iff $x_i = 1$ and the edge e_2 iff $y_i = 1$. We now show that the resulting graph G contains a copy of H if and only if it is a YES instance of DISJOINTNESS.

Suppose that it is a YES instance of DISJOINTNESS. So there is a $j \in [n]$ such that $x_j = 1 = y_j$. Therefore, to the copy G_j of H' we would have added the edges e_1 and e_2 which would complete it into H. So G contains a copy of H. In other direction, suppose that G contains a copy of H. Note that since we add n disjoint copies of H' and add at most two edges (e_1 and e_2) to each copy, it follows that each connected component of G is in fact a subgraph of $H = H' \cup (e_1 + e_2)$. Since H is connected and G contains a copy of H, some connected component of G must exactly be the graph H, i.e, to some copy G_i of H' we must have added both the edges e_1 and e_2. This implies $x_i = 1 = y_i$, and so DISJOINTNESS answers YES.

Since each connected component of G is a subgraph of H, the minimality of H implies that $\mathcal{G}$ contains a graph from G iff G contains a copy of H, which in turn is true iff DISJOINTNESS answers YES. Therefore, any p-pass (randomized) streaming algorithm that can determine whether a graph is $\mathcal{G}$-free (i.e., answers the question with $k = 0$) gives a communication protocol for DISJOINTNESS, and hence requires $\Omega(n/p)$ space. □

This implies lower bounds for the following set of problems:

Theorem 28. *For each of the following problems, any p-pass (randomized) algorithm requires $\Omega(n/p)$ space:* FEEDBACK VERTEX SET(k), ODD CYCLE TRANSVERSAL(k), EVEN CYCLE TRANSVERSAL(k) *and* TRIANGLE DELETION(k).

Proof. We first define the problems below:

FEEDBACK VERTEX SET(k) *Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k
Question: Does there exist a set $X \subseteq V$ of size at most k such that $G \setminus X$ has no cycles?

ODD CYCLE TRANSVERSAL(k) *Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k
Question: Does there exist a set $X \subseteq V$ of size at most k such that $G \setminus X$ has no odd cycles?

EVEN CYCLE TRANSVERSAL(k) *Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k
Question: Does there exist a set $X \subseteq V$ of size at most k such that $G \setminus X$ has no even cycles?

TRIANGLE DELETION(k) *Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k
Question: Does there exist a set $X \subseteq V$ of size at most k such that $G \setminus X$ has no triangles?

Now we show how each of these problems can be viewed as a $\mathcal{G}$-FREE DELETION problem for an appropriate choice of *bad* $\mathcal{G}$.

- FEEDBACK VERTEX SET(k): Take $\mathcal{G} = \{C_3, C_4, C_5, \ldots\}$ and $H = C_3$
- ODD CYCLE TRANSVERSAL(k): Take $\mathcal{G} = \{C_3, C_5, C_7, \ldots\}$ and $H = C_3$

- EVEN CYCLE TRANSVERSAL(k): Take $\mathcal{G} = \{C_4, C_6, C_8, \ldots\}$ and $H = C_4$
- TRIANGLE DELETION(k): Take $\mathcal{G} = \{C_3\}$ and $H = C_3$

We verify the conditions for FEEDBACK VERTEX SET(k); the proofs for other problems are similar. Note that the choice of $\mathcal{G} = \{C_3, C_4, C_5, \ldots\}$ and $H = C_3$ implies that $\mathcal{G}$ is *bad* since each graph in $\mathcal{G}$ is connected, the graph H belongs to $\mathcal{G}$, has at least two distinct edges and is a minimal element of $\mathcal{G}$ (under operation of taking subgraphs). Finally, finding a set X such that the graph $G \setminus X$ is $\mathcal{G}$-free implies that it has no cycles, i.e., X is a feedback vertex set for G. □

It is easy to see that the same proofs also work for the edge deletion versions of the ODD CYCLE TRANSVERSAL(k), EVEN CYCLE TRANSVERSAL(k) and the TRIANGLE DELETION(k) problems.

D.2.3 $\mathcal{G}$-EDITING

Definition 29. *A set of graphs $\mathcal{G}$ is* good *if there is a minimal (under operation of taking subgraphs) connected graph $H \in \mathcal{G}$ such that H has at least two distinct edges.*

Definition 29 looks very similar to Definition 26: however there is a subtle difference. Each graph in a *bad* set of graphs must be connected while only a minimal graph in a *good* set of graphs is required to be connected. This difference is used crucially in the proofs of Theorem 27 and Theorem 30.

For any *good* set of graphs $\mathcal{G}$, we now show a lower bound for the following general problem:

$\mathcal{G}$-EDITING(k) *Parameter*: k
Input: A graph class $\mathcal{G}$, an undirected graph $G = (V, E)$ and an integer k
Question: Does there exist a set X of k edges such that $(V, E \cup X)$ contains a graph from $\mathcal{G}$?

Theorem 30. *For a* good *set of graphs $\mathcal{G}$, any p-pass (randomized) streaming algorithm for the* $\mathcal{G}$-EDITING(k) *problem needs $\Omega(n/p)$ space .*

Proof. Since $\mathcal{G}$ is a *good* set of graphs, there is a minimal graph $H \in \mathcal{G}$ such that H is connected and has at least two distinct edges, say e_1 and e_2. Let $H' := H \setminus \{e_1, e_2\}$. Given an instance of DISJOINTNESS, we create a graph G which consists of n disjoint copies say $G_1, G_2, \ldots, G_n$ of H'. By minimality of H, it follows that $H' \notin \mathcal{G}$. For each $i \in [n]$ we add to G_i the edge e_1 iff $x_i = 1$ and the edge e_2 iff $y_i = 1$. Let the resulting graph be G.

We now show that G contains a copy of H if and only if DISJOINTNESS answers YES. Suppose that G contains a copy of H. Note that since we add n disjoint copies of H' and add at most two edges (e_1 and e_2) to each copy, it follows that each connected component of G is in fact a subgraph of $H = H' \cup (e_1 + e_2)$. Since H is connected and G contains a copy of H, some connected component of G must exactly be the graph H, i.e, to some copy G_i of H' we must have added both the edges e_1 and e_2. This implies $x_i = 1 = y_i$, and so DISJOINTNESS answers YES. Now suppose that DISJOINTNESS answers YES, i.e., there exists $j \in [n]$ such that $x_j = 1 = y_j$. Therefore, to the copy G_j of H' we would have added the edges e_1 and e_2 which would complete it into H. So G contains a copy of H.

Otherwise due to minimality of H, the graph G does not contain any graph from $\mathcal{G}$. Therefore, any p-pass (randomized) streaming algorithm that can determine whether a graph G contains a graph from $\mathcal{G}$ (i.e., answers the question with $k = 0$) gives a communication protocol for DISJOINTNESS, and hence requires $\Omega(n/p)$ space. □

This implies lower bounds for the following set of problems:

Theorem 31. *For each of the following problems, any p-pass (randomized) algorithm requires $\Omega(n/p)$ space:* TRIANGLE PACKING(k)*,* s-STAR PACKING(k) *and* PATH(k)*.*

Proof. We first define the problems below:

TRIANGLE PACKING(k) *Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k
Question: Do there exist at least k vertex disjoint triangles in G?

s-STAR PACKING(k) *Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k
Question: Do there exist at least k vertex disjoint instances of $K_{1,s}$ in G (where $s \geq 3$)?

PATH(k) *Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k
Question: Does there exist a path in G of length $\geq k$?

Now we show how each of these problems can be viewed as a $\mathcal{G}$-EDITING problem for an appropriate choice of *good* $\mathcal{G}$.

- TRIANGLE PACKING(k) with $k = 1$: Take $\mathcal{G} = \{C_3\}$ and $H = C_3$
- s-STAR PACKING(k) with $k = 1$: Take $\mathcal{G} = \{K_{1,s}\}$ and $H = K_{1,s}$
- PATH(k) with $k = 3$: Take $\mathcal{G} = \{P_3, P_4, P_5, \ldots\}$ and $H = P_3$

We verify the conditions for TRIANGLE PACKING(k) with $k = 1$; the proofs for other problems are similar. Note that the choice of $\mathcal{G} = \{C_3\}$ and $H = C_3$ implies that $\mathcal{G}$ is *good* since $\mathcal{G}$ only contains one graph H which is connected and has at least two distinct edges. Finally, finding a set of edges X such that the graph $(V, E \cup X)$ contains a graph from $\mathcal{G}$ implies that it has at least one C_3, i.e., X is a solution for TRIANGLE PACKING(k) with $k = 1$. □

D.2.4 Lower Bound for CLUSTER VERTEX DELETION

We now show a lower bound for the CLUSTER VERTEX DELETION(k) problem.

Definition 32. *We say that G is a* cluster *graph if each connected component of G is a clique.*

CLUSTER VERTEX DELETION(k) *Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k
Question: Does there exist a set $X \subseteq V$ of size at most k such that $G \setminus X$ is a cluster graph?

Theorem 33. *For the* CLUSTER VERTEX DELETION(k) *problem, any p-pass (randomized) streaming algorithm needs $\Omega(n/p)$ space .*

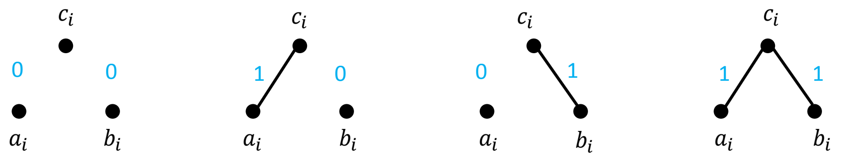

Figure 2: Gadget for reduction from DISJOINTNESS to CLUSTER VERTEX DELETION

Proof. Given an instance of DISJOINTNESS, we create a graph G on $3n$ vertices as follows. For each $i \in [n]$ we create three vertices a_i, b_i, c_i. Insert the edge (a_i, c_i) iff $x_i = 1$ and the edge (b_i, c_i) iff $y_i = 1$ This is illustrated in Figure 2.

Now we will show that each connected component of G is a clique iff DISJOINTNESS answers NO. In the first direction suppose that each connected component of G is a clique. Then there cannot exist $i \in [n]$ such that $x_i = 1 = y_i$ because then the vertices a_i, b_i, c_i will form a connected component which is a P_3; this contradicts the assumption that each connected component of G is a clique. In reverse direction, suppose that DISJOINTNESS answers NO. Then it is easy to see that each connected component of G is either P_1 or P_2, both of which are cliques.

Therefore, any p-pass (randomized) streaming algorithm that can determine whether a graph is a cluster graph (i.e., answers the question with $k = 0$) gives a communication protocol for DISJOINTNESS, and hence requires $\Omega(n/p)$ space. [2] □

D.2.5 Lower Bound for MINIMUM FILL-IN

We now show a lower bound for the MINIMUM FILL-IN(k) problem.

Definition 34. *We say that G is a* chordal *graph if it does not contain an induced cycle of length ≥ 4.*

MINIMUM FILL-IN(k)	*Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k	
Question: Does there exist a set X of at most k edges such that $(V, E \cup X)$ is a chordal graph?	

Theorem 35. *For the* MINIMUM FILL-IN(k) *problem, any p-pass (randomized) streaming algorithm needs $\Omega(n/p)$ space .*

Proof. We reduce from the DISJOINTNESS problem in communication complexity. Given an instance of DISJOINTNESS, we create a graph G on $4n$ vertices as follows. For each $i \in [n]$ we create vertices a_i, b_i, c_i, d_i and insert edges (a_i, b_i) and (c_i, d_i). Insert the edge (a_i, c_i) iff $x_i = 1$ and the edge (b_i, c_i) iff $y_i = 1$. This is illustrated in Figure 3.

Now we will show that G is chordal iff DISJOINTNESS answers NO. In the first direction suppose that G is chordal. Then there cannot exist $i \in [n]$ such that $x_i = 1 = y_i$ because then the vertices a_i, b_i, c_i, d_i

[2] It is easy to see that the same proof also works for the problems of CLUSTER EDGE DELETION(k) where we can delete at most k edges and CLUSTER EDITING(k) where we can delete/add at most k edges

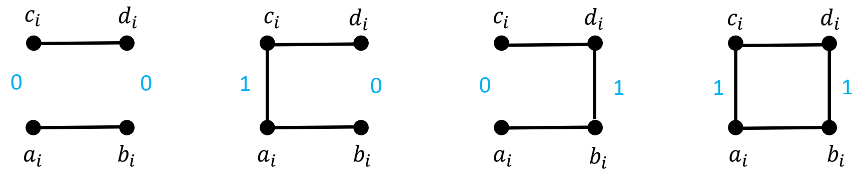

Figure 3: Gadget for reduction from DISJOINTNESS to MINIMUM FILL-IN

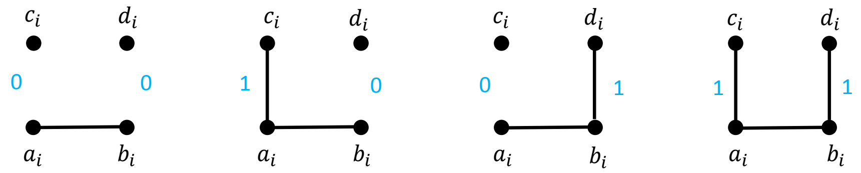

Figure 4: Gadget for reduction from DISJOINTNESS to COGRAPH VERTEX DELETION

will form an induced C_4; contradicting the assumption that G is chordal. In reverse direction, suppose that DISJOINTNESS answers NO. Then it is easy to see that each connected component of G is either P_2 or P_3. Hence, G cannot have an induced cycle of length ≥ 4, i.e., G is chordal.

Therefore, any p-pass (randomized) streaming algorithm that can determine whether a graph is a chordal graph (i.e., answers the question with $k = 0$) gives a communication protocol for DISJOINTNESS, and hence requires $\Omega(n/p)$ space. □

D.2.6 Lower Bound for COGRAPH VERTEX DELETION

We now show a lower bound for the COGRAPH VERTEX DELETION(k) problem.

Definition 36. *We say that G is a* cograph *if it does not contain an induced P_4.*

COGRAPH VERTEX DELETION(k) *Parameter*: k
Input: An undirected graph $G = (V, E)$ and an integer k
Question: Does there exist a set $X \subseteq V$ of size at most k such that $G \setminus X$ is a cograph?

Theorem 37. *For the* COGRAPH VERTEX DELETION(k) *problem, any p-pass (randomized) streaming algorithm needs $\Omega(n/p)$ space .*

Proof. We reduce from the DISJOINTNESS problem in communication complexity. Given an instance of DISJOINTNESS, we create a graph G on $4n$ vertices as follows. For each $i \in [n]$ we create vertices a_i, b_i, c_i, d_i

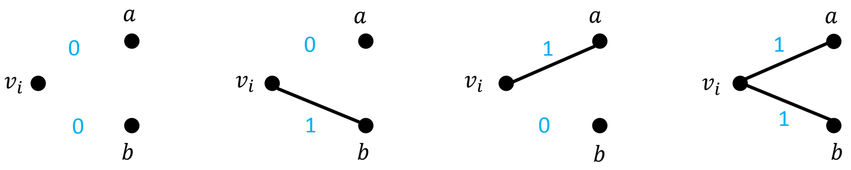

Figure 5: Gadget for reduction from Disjointness to BIPARTITE COLORFUL NEIGHBORHOOD

and insert edges (a_i, b_i). Insert the edge (a_i, c_i) iff $x_i = 1$ and the edge (b_i, c_i) iff $y_i = 1$. This is illustrated in Figure 4.

Now we will show that G has an induced P_4 if and only if DISJOINTNESS answers YES. In the first direction suppose that G has an induced P_4. Since each connected component of G can have at most 4 vertices, it follows that the P_4 is indeed given by the path $c_i - a_i - b_i - d_i$ for some $i \in [n]$. By construction of G, this implies that $x_i = 1 = y_i$, i.e., DISJOINTNESS answers YES. In reverse direction, suppose that DISJOINTNESS answers YES. Then there exists $j \in [n]$ such that the edges (a_i, c_i) and (b_i, d_i) belong to G. Then G has the following induced P_4 given by $c_j - a_j - b_j - d_j$.

Therefore, any p-pass (randomized) streaming algorithm that can determine whether a graph is a cograph (i.e., answers the question with $k = 0$) gives a communication protocol for DISJOINTNESS, and hence requires $\Omega(n/p)$ space. □

D.2.7 BIPARTITE COLORFUL NEIGHBORHOOD

We now show a lower bound for the BIPARTITE COLORFUL NEIGHBORHOOD(k) problem.

BIPARTITE COLORFUL NEIGHBORHOOD(k) *Parameter*: k
Input: A bipartite graph $G = (A, B, E)$ and an integer k
Question: Is there a 2-coloring of B such that there exists a set $S \subseteq A$ of size at least k such that each element of S has at least one neighbor in B of either color?

Theorem 38. *For the* BIPARTITE COLORFUL NEIGHBORHOOD(k) *problem, any p-pass (randomized) streaming algorithm needs $\Omega(n/p)$ space .*

Proof. We reduce from the DISJOINTNESS problem in communication complexity. Given an instance of DISJOINTNESS, we create a graph G on $n + 2$ vertices as follows. For each $i \in [n]$ we create a vertex v_i. In addition, we have two special vertices a and b. For each $i \in [n]$, insert the edge (a, v_i) iff $x_i = 1$ and the edge (b, v_i) iff $y_i = 1$. Let $A = \{v_1, v_2, \ldots, v_n\}$ and $B = \{a, b\}$. This is illustrated in Figure 5.

Now we will show that G answers YES for BIPARTITE COLORFUL NEIGHBORHOOD(k) with $k = 1$ iff DISJOINTNESS answers YES. In the first direction suppose that G answers YES for BIPARTITE COLORFUL NEIGHBORHOOD(k) with $k = 1$. Let v_i be the element in A which has at least one neighbor in B of either color. Since $|B| = 2$, this means that v_i is adjacent to both a and b, i.e., $x_i = 1 = y_i$ and hence DISJOINTNESS answers YES. In reverse direction, suppose that DISJOINTNESS answers YES. Hence, there exists $j \in [n]$ such that $x_j = 1 = y_j$. This implies that v_j is adjacent to both a and b. Consider the 2-coloring of B by giving different colors to a and b. Then $S = \{v_j\}$ satisfies the condition of having a neighbor of each color in B, and hence G answers YES for BIPARTITE COLORFUL NEIGHBORHOOD(k) with $k = 1$.

Therefore, any p-pass (randomized) streaming algorithm that can solve BIPARTITE COLORFUL NEIGHBORHOOD(k) with $k = 1$ gives a communication protocol for DISJOINTNESS, and hence requires $\Omega(n/p)$ space.

□