Matroid-Constrained Maximum Vertex Cover: Approximate Kernels and Streaming Algorithms

Chien-Chung Huang
CNRS, DI ENS, École normale supérieure, Université PSL, Paris, France

François Sellier
Université Paris Cité, CNRS, IRIF, F-75013, Paris, France
MINES ParisTech, Université PSL, F-75006, Paris, France

Abstract

Given a graph with weights on the edges and a matroid imposed on the vertices, our problem is to choose a subset of vertices that is independent in the matroid, with the objective of maximizing the total weight of covered edges. This problem is a generalization of the much studied MAX k-VERTEX COVER problem, where the matroid is the simple uniform matroid, and it is also a special case of maximizing a monotone submodular function under a matroid constraint.

In this work, we give a Fixed Parameter Tractable Approximation Scheme (FPT-AS) when the given matroid is a partition matroid, a laminar matroid, or a transversal matroid. Precisely, if k is the rank of the matroid, we obtain $(1-\varepsilon)$ approximation using $\left(\frac{1}{\varepsilon}\right)^{O(k)} n^{O(1)}$ time for partition and laminar matroids and using $\left(\frac{1}{\varepsilon}+k\right)^{O(k)} n^{O(1)}$ time for transversal matroids. This extends a result of Manurangsi for uniform matroids [26]. We also show that these ideas can be applied in the context of (single-pass) streaming algorithms.

Our FPT-AS introduces a new technique based on matroid union, which may be of independent interest in extremal combinatorics.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Approximation algorithms analysis; Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Maximum vertex cover, matroid, approximate kernel, streaming

Digital Object Identifier 10.4230/LIPIcs.SWAT.2022.27

Funding This work was funded by the grants ANR-19-CE48-0016 and ANR-18-CE40-0025-01 from the French National Research Agency (ANR).

Acknowledgements The authors thank the anonymous reviewers for their helpful comments. One of them especially pointed out a mistake on a counter-example for gammoids in the submitted version.

1 Introduction

Let $G=(V,E)$ be a graph. A weight $w(e)$ is associated with each edge $e \in E$. By convention we set $n=|V|$ and $m=|E|$. For a vertex $v \in V$ we define $\delta(v)$ the set of edges that are incident to v. The *degree* of a vertex $v \in V$, denoted $\deg(v)$, is the size of $\delta(v)$, and we define the *weighted degree* of a vertex $v \in V$ as the sum $\deg_w(v) = \sum_{e \in \delta(v)} w(e)$. For two sets of vertices $S, T \subseteq V$ in a graph G, we denote $E_G(S,T) = \sum_{e \in E, e \cap S \neq \emptyset, e \cap T \neq \emptyset} w(e)$ the sum of the weights of the edges that have one endpoint in S and one endpoint in T. Then $E_G(S,S)$, abbreviated $E_G(S)$, denotes the sum of the weights of the edges that are covered by S (*i.e.* having at least one of its endpoints in S).

Let $\mathcal{M}=(V,\mathcal{I})$ be a matroid on the ground set V. Recall that $\mathcal{M}=(V,\mathcal{I})$ is a matroid if the following three conditions hold: (1) $\emptyset \in \mathcal{I}$, (2) if $X \subseteq Y \in \mathcal{I}$, then $X \in \mathcal{I}$, and (3) if $X, Y \in \mathcal{I}, |Y| > |X|$, there exists an element $e \in Y \backslash X$ so that $X \cup \{e\} \in \mathcal{I}$. The sets in $\mathcal{I}$

are the *independent sets* and the *rank* k of the matroid $\mathcal{M}$ is defined as $\max_{X\in\mathcal{I}}|X|$. For more details about matroids, we refer the reader to [33]. In this paper, given a set $S \subseteq V$ and $v \in V$, we will denote $S \cup \{v\}$ by $S + v$ and $S\backslash\{v\}$ by $S - v$ for conciseness.

The problem that we consider in this paper is to choose an independent set of vertices $S \in \mathcal{I}$, with the objective of maximizing $E_G(S)$, namely, the total weight of the edges covered by S.

Let us put our problem in a larger picture. When the given matroid $\mathcal{M}$ is a uniform matroid (see below for a formal definition), our problem reduces to the MAX k-VERTEX-COVER problem, where we want to choose k arbitrary vertices so as to maximize the total weight of covered edges. This is a classical problem with a long history: the greedy heuristics is known to give $1 - 1/e$ approximation as shown by Hochbaum and Pathria [23]. Ageev and Sviridenko [1] propose an LP-based approach and the technique of pipage rounding to obtain 3/4 approximation. Using SDP, Feige and Langberg [14] improve this ratio to $3/4 + \delta$ for some small constant $\delta > 0$. The current best approximation ratio is 0.92, achieved by Manurangsi [26]. For some special cases of the problem, different ratios are also obtained, e.g. see [4, 21, 22]. On the hardness side, to our knowledge, the best inapproximability ratio is due to Austrin and Stankovic [2], which is 0.929.

The MAX k-VERTEX-COVER has also been studied through the lens of fixed-parameterized-tracability. Guo et al. [19] show the problem to be $W[1]$-hard with k as parameter, thus showing the unlikelihood of getting an exact solution in FPT time. Nonetheless, Marx [28] shows that it is possible to get a near-optimal solution in FPT time. Precisely, he gives an FPT approximation scheme (FPT-AS), that delivers a $(1-\varepsilon)$-approximate solution in $(k/\epsilon)^{O(k^3/\epsilon)}n^{O(1)}$ time. This running time is later improved by Gupta et al. [20] and Manurangsi [26].

Here we recall the definition of an FPT-AS [28]:

▶ **Definition 1.** *Given a parameter function κ associating a natural number to each instance $x \in I$ of a given problem, a Fixed-Parameter Tractable Approximation Scheme (FPT-AS) is an algorithm that can provide a $(1-\varepsilon)$ approximation in $f(\varepsilon, \kappa(x)) \cdot |x|^{O(1)}$ time.*

In our case, the instances are made of a graph and a matroid, and the parameter of an instance is the rank k of its matroid.

Regarding the more general case of an arbitrary matroid of rank k, one can obtain 3/4 approximation in polynomial time by combining known techniques.[1] This is also a special case of maximizing a submodular function (more precisely, a coverage function) under matroid constraint, for which a $1 - 1/e$ approximation can be achieved in polynomial time [6, 18]. In this work, we try to do better than this ratio for some special cases of matroids, in the context of fixed-parameter algorithms. We also show that the ideas developed here can be applied in the streaming setting [31]. In streaming, maximizing of a submodular function under a general matroid constraint has received much attention recently [8, 10, 16].

[1] Ageev and Sviridenko [1] show that, for the case of a uniform matroid, the optimal fractional solution x^* of the LP has at least 3/4 of the optimal value. They then use the pipage rounding to transform it into an integral solution with value no less than x^*. The same LP approach can be generalized for arbitrary matroids. The optimal fractional solution can be obtained by Ellipsoid algorithm: even though the linear program to describe the independent sets of an arbitrary matroid may use exponentially many constraints, we can design a separation oracle using an algorithm of Cunningham [12]. What remains is just the pipage rounding with a general matroid – this is already known to be do-able by Calinescu et al. [7]. We thank Pasin Manurangsi for communicating to us this method.

1.1 Our Contribution

Let us recall some definitions. A *uniform matroid* of rank k is a matroid where the independent sets are the sets S of cardinality at most k. A *partition matroid* is a matroid where we are given a partition $V_1, \ldots, V_r$ of the ground set V and bounds $k_1, \ldots, k_r$ such that a set S is independent if for all $1 \leq i \leq r$, $|S \cap V_i| \leq k_i$. A *laminar matroid* is given as a laminar family $V_1, \ldots V_r$ of V, *i.e.* given $V_i \neq V_j$, then either $V_i \cap V_j = \emptyset$, or $V_i \subset V_j$, or $V_j \subset V_i$, along with bounds $k_1, \ldots, k_r$. A set $S \subseteq V$ is independent if for all $1 \leq i \leq r$, $|S \cap V_i| \leq k_i$. Finally, a *transveral matroid* is given in a family $V_1, \ldots, V_k \subseteq V$, where V_is are not necessarily disjoint, and a set $S = \{u_1, \cdots, u_t\}$ is independent if and only if for each element u_i, there exists a distinct $\phi(i)$ so that $u_i \in V_{\phi(i)}$. These simple types of matroid have been extensively studied in a large variety of contexts.

A uniform matroid is a special case of a partition matroid, which is again a special case of a laminar or a transversal matroid. However, laminar matroids and transversal matroids are not inclusive of each other [32]. Transversal matroids were introduced in the 60s, by Edmonds and Fulkerson [13] and by Mirsky and Perfect [30]. They unified many results in transversal theory and are generally considered an important class of matroids, e.g. see [35]. Laminar matroids receive much attention recently in the community of theoretical computer science, especially in the context of matroid secretary problem, e.g., see [3, 9, 15, 24, 34]. Our results involve these kinds of matroids.

▶ **Theorem 2.** *For every $\varepsilon > 0$, we can extract an approximate kernel $V' \subseteq V$ in polynomial time so that a $(1-\varepsilon)$-approximate solution is contained in V'. The size of the kernel V' depends on the type of the given matroid $\mathcal{M}$.*

- **(i)** $|V'| \leq \frac{k}{\varepsilon}$ *when $\mathcal{M}$ is a partition matroid;*
- **(ii)** $|V'| \leq \frac{2k}{\varepsilon}$ *when $\mathcal{M}$ is a laminar matroid;*
- **(iii)** $|V'| \leq \frac{k}{\varepsilon} + k(k-1)$ *when $\mathcal{M}$ is a transversal matroid.*

Furthermore, by a brute force enumeration, we can find the desired $1-\varepsilon$ approximation in $\left(\frac{1}{\varepsilon}\right)^{O(k)} n^{O(1)}$ time for partition and laminar matroids and $\left(\frac{1}{\varepsilon}+k\right)^{O(k)} n^{O(1)}$ time for transversal matroids.

In addition, by a straightforward modification of our proofs in Section 2 (see Appendix A), we can show the following corollary.

▶ **Corollary 3.** *Suppose that we are given a hypergraph $G = (V, E)$ with edge size bounded by a constant $\eta \geq 2$. We can compute a $(1-(\eta-1)\cdot\varepsilon)$ approximation using $\left(\frac{1}{\varepsilon}\right)^{O(k)} n^{O(1)}$ time for partition and laminar matroids and $\left(\frac{1}{\varepsilon}+k\right)^{O(k)} n^{O(1)}$ time for transversal matroids.*

Put slightly differently, when G is a hypergraph with edge size at most η, we can obtain $1-\varepsilon$ approximation in $\left(\frac{\eta}{\varepsilon}\right)^{O(k)} n^{O(1)}$ or $\left(\frac{\eta}{\varepsilon}+k\right)^{O(k)} n^{O(1)}$ time, depending on the type of matroid. To see the interest of this corollary, we recall that recently Manurangsi [27] showed that if η is unbounded, one cannot obtain an approximation ratio better than $1-1/e+\varepsilon$, assuming GAP-ETH, in FPT time (where the matroid rank k is the parameter). This result holds even for the simplest uniform matroid. Thus Corollary 3 implies that one can circumvent this lower bound by introducing another parameter η, even for more general matroids.

Our algorithm is inspired by that of Manurangsi [26] for the case of uniform matroid. So let us briefly summarize his approach: an *approximate kernel*[2] V' is first extracted from V, where V' is simply made of the k/ε vertices with the largest weighted degrees. Let O be an optimal solution. Apparently, a vertex of O is either part of the kernel V', or its weighted degree is dominated by all vertices in $V'\backslash O$. To recover the optimal value, we can potentially use the vertices in $V'\backslash O$ to replace the vertices in $O\backslash V'$. However, there is a risk in doing this: an edge among the vertices in $V'\backslash O$ can be double-counted, if both of its endpoints are chosen to replace the vertices in $O\backslash V'$. To circumvent this issue, Manurangsi uses a random sampling argument to show that in expectation such double counting is negligible. Therefore, by the averaging principle, there exists a $(1-\varepsilon)$-approximate solution in the kernel V', which can be found using brute force.

To generalize the approach of Manurangsi for more general matroids, one has to answer the quintessential question: how does one guarantee that the sampled vertices, along with $O \cap V'$, are independent in $\mathcal{M}$? To overcome this difficulty, we introduce a new technique. We take the union of some number τ of matroids $\mathcal{M}$. Such a union is still a matroid, which we denote as $\tau\mathcal{M}$. We then apply a greedy algorithm on $\tau\mathcal{M}$ (based on non-increasing weighted degrees) to construct an independent set V' in $\tau\mathcal{M}$. We show that such a set V' is "robust" (see Definition 4) in the sense that we can sample vertices from V' so that they, along with $O \cap V'$, are *always* independent and in expectation cover edges of weight at least $1-\varepsilon$ times that of O.

We note that the value of τ automatically gives an upper bound on the kernel size V', which is τk. Theorem 6 shows the required scale of τ, depending on the type of the given matroid. We leave as an open question whether for matroids more general than considered in the paper, a larger τ can always yield the kernel.

In the last part of this work, we consider the problem in the semi-streaming model [31]. In that context, the edges in E arrive over time but we have only limited space (for instance, $O(n \cdot polylog(n)) = o(m)$) and cannot afford to store all edges in E. In this context we can also obtain a $(1-\varepsilon)$ approximation using $O(\frac{nk}{\varepsilon})$ space in a single pass.[3] The idea of using (parameterized) kernels for streaming algorithms has recently been introduced, for instance in [11, 29]. We also show that a FPT-streaming algorithm can be derived from our ideas to get a $(1-\varepsilon)$ approximation for a special form of maximization of a coverage function with bounded frequency (see Theorem 14, Remark 15 and Appendix B for details).

2 Kernelization Framework

In this section, we give a general framework to construct the kernel by a greedy procedure and show how such a kernel contains a $(1-\varepsilon)$-approximate solution.

▶ **Definition 4.** *Let $\mathcal{M} = (V, \mathcal{I})$ be a matroid with weights $\omega : V \to \mathbb{R}^+$. We say $V' \subseteq V$ is t-robust if given any base $O \in \mathcal{I}$, there is a bijection from the elements $u_1, \cdots, u_t \in O\backslash V'$ to subsets $U_{u_1}, \cdots, U_{u_t} \subseteq V'\backslash O$ so that*

- **(i)** *the U_{u_i}s are mutually disjoint and $|U_{u_i}| = t$,*
- **(ii)** *all elements in U_{u_i} have weights no less than u_i,*
- **(iii)** *by taking an arbitrary element $u'_i \in U_{u_i}$ for all i, $(V' \cap O) \cup \{u'_i\}_{i=1}^t$ is a base in $\mathcal{M}$.*

[2] In the rest of the paper, we will just say kernel, dropping the adjective. The interest for this kind of kernel has risen recently in the community [17, 25].

[3] Here we assume that the matroid is given in the form of oracle, in which the algorithm has access freely – this is a standard assumption in the streaming setting when matroids are involved.

We next recall the definition of matroid union.

▶ **Definition 5.** *Suppose that $\mathcal{M} = (V, \mathcal{I})$ is a matroid. Then we can define $\tau\mathcal{M} = (V, \mathcal{I}_\tau)$ as the union of τ matroids $\mathcal{M}$, as follows: $S \in \mathcal{I}_\tau$ if S can be partitioned into $S_1 \cup \cdots \cup S_\tau$ so that each $S_i \in \mathcal{I}$.*

Recall that the union of matroids is still a matroid and here the rank of $\tau\mathcal{M}$ is at most τ times the rank of $\mathcal{M}$. e.g. see [33, Chapter 42]. We can now state our main theorem.

▶ **Theorem 6.** *Let $\mathcal{M} = (V, \mathcal{I})$ be a matroid with weights $\omega : V \to \mathbb{R}^+$ and rank k. Consider the following greedy procedure on $\tau\mathcal{M} = (V, \mathcal{I}_\tau)$ to construct V': initially $V' = \emptyset$. Process the elements in V by non-increasing weights ω. For each element u, if $V' + u \in \mathcal{I}_\tau$, add u into V', otherwise, ignore it. The final V' is t-robust*

- **(i)** *if $\mathcal{M}$ is a partition matroid and $\tau \geq t$,*
- **(ii)** *if $\mathcal{M}$ is a laminar matroid and $\tau \geq 2t$,*
- **(iii)** *if $\mathcal{M}$ is a transversal matroid and $\tau \geq t + k - 1$.*

Notice that the rank of the matroid $\tau\mathcal{M}$ gives an upper-bound on the size of V'. The next section will give the proof of this theorem for each type of matroid considered. In the following we show how it can be used to construct the $1 - \varepsilon$ approximation.

Let the weight $\omega : V \to \mathbb{R}^+$ be the weighted degrees in the graph $G = (V, E)$, that is, $\omega(u) = \deg_w(u)$. Apply Theorem 6 by setting $t = \frac{1}{\varepsilon}$. Then V' is $\frac{1}{\varepsilon}$-robust. Note that we suppose that $\frac{1}{\varepsilon}$ is an integer, otherwise we could take $t = \lceil \frac{1}{\varepsilon} \rceil$.

Based on V', we create a new graph $G' = (V', E')$, where an original edge $e = \{u, v\}$ is retained in E' if both of its endpoints are in V'. In case only one endpoint, say u is in V', we add a self-loop to u in E' to represent this edge.

▶ **Lemma 7.** *Suppose that V' is the constructed set that is $\frac{1}{\varepsilon}$-robust. Then V' contains a set S such that $S \in \mathcal{I}$ and $E_G(S) \geq (1 - \varepsilon)E_G(O)$ where O denotes an optimal solution of the problem.*

Proof. Let $O \in \mathcal{I}$ be an optimal solution. We denote $O^{in} = O \cap V'$, $O^{out} = O \backslash O^{in}$. Then by $\frac{1}{\varepsilon}$-robustness, we have mutually disjoint sets $U_v \subseteq V' \backslash O$ for each $v \in O^{out}$, each of size $\frac{1}{\varepsilon}$. We set $U = \cup_{v \in O^{out}} U_v$. We construct a set $S \subseteq V'$ as follows: S is initialized as O^{in}. Then from each set U_v, for all $v \in O^{out}$, pick an element at random and add it into S. By definition of $\frac{1}{\varepsilon}$-robustness, S is independent in $\mathcal{M}$.

Next we will show that

$$\mathbb{E}[E_G(S)] \geq (1 - \varepsilon) \cdot \mathbb{E}[E_G(O)].$$

Let $U^* = S \backslash O^{in}$, i.e. those elements that are added into S randomly. First, we have that:

$$E_G(S) = E_G(O^{in}) + E_G(U^*) - E_G(O^{in}, U^*).$$

We bound $\mathbb{E}[E_G(O^{in}, U^*)]$ as follows. By construction, $\mathbb{P}[u \in U^*] = \varepsilon$ for all $u \in U$. Then,

$$\mathbb{E}[E_G(O^{in}, U^*)] = \sum_{u \in U} \sum_{v \in O^{in}} w(\{u, v\}) \cdot \mathbb{P}[u \in U^*] = \varepsilon \sum_{u \in U} \sum_{v \in O^{in}} w(\{u, v\}) \leq \varepsilon \cdot E_G(O^{in}).$$

Furthermore, the value $\mathbb{E}[E_G(U^*)]$ can be rearranged as follows:

$$\mathbb{E}[E_G(U^*)] = \mathbb{E}\left[\sum_{u \in U^*} \left(\deg_w(u) - \frac{1}{2} \sum_{v \in U^* \backslash \{u\}} w(\{u, v\})\right)\right]$$

$$= \sum_{u\in U}\left(\deg_w(u)\cdot\mathbb{P}[u\in U] - \frac{1}{2}\sum_{v\in U\setminus\{u\}} w(\{u,v\})\cdot\mathbb{P}[u\in U^*\wedge v\in U^*]\right)$$
$$\geq \sum_{u\in U}\left(\deg_w(u)\cdot\varepsilon - \frac{1}{2}\sum_{v\in U\setminus\{u\}} w(\{u,v\})\cdot\varepsilon^2\right) \geq \varepsilon(1-\varepsilon/2)\left(\sum_{u\in U}\deg_w(u)\right),$$

where the first inequality comes from the fact that $\mathbb{P}[u\in U^*\wedge v\in U^*]\leq \mathbb{P}[u\in U^*]\cdot\mathbb{P}[v\in U^*]$.

Recall that by robustness, for all $u\in O^{out}$, the elements of U_u have weighted degrees no less than that of u. Therefore,

$$\mathbb{E}[E_G(U^*)] \geq \varepsilon(1-\varepsilon/2)\left(\sum_{u\in O^{out}}\sum_{v\in U_u}\deg_w(v)\right) \geq \varepsilon(1-\varepsilon/2)\left(\sum_{u\in O^{out}}\frac{1}{\varepsilon}\cdot\deg_w(u)\right)$$
$$\geq (1-\varepsilon/2)\cdot E_G(O^{out}).$$

As a result, we get:

$$\mathbb{E}[E_G(S)] \geq E_G(O^{in}) + (1-\varepsilon/2)\cdot E_G(O^{out}) - \varepsilon\cdot E_G(O^{in}) \geq (1-\varepsilon)\cdot E_G(O).$$

By averaging principle, there exists $S\subseteq V'$ such that $S\in\mathcal{I}$ and $E_G(S)\geq(1-\varepsilon)\cdot E_G(O)$. ◀

3 Proof of Theorem 6

3.1 Partition Matroids

Consider a partition matroid $\mathcal{M}=(V,\mathcal{I})$ defined by a partition $V_1,\ldots,V_r$ of V and bounds $k_1,\ldots,k_r$. Given $t\in\mathbb{N}$, we take in each set V_i of the partition the $\min\{|V_i|, t\cdot k_i\}$ elements having the largest weighted degrees. We denote these extracted sets as V_i' and their union as V'. Clearly, this is the same as the greedy algorithm stated in Theorem 6 applied on $t\mathcal{M}$.

To see that V' is t-robust, let $O\in\mathcal{I}$ be a base. We denote $O^{in}=O\cap V'$, $O^{out}=O\setminus O^{in}$, $O_i=O\cap V_i$, $O_i^{in}=O\cap V_i'$, $O_i^{out}=O_i\setminus O_i^{in}$, and we set $\overline{U}_i\subseteq V_i'\setminus O_i^{in}$ as an arbitrary subset of cardinality $t\cdot|O_i^{out}|$, for $1\leq i\leq r$ (it is possible as $O_i\neq O_i^{in}$ implies that $|V_i'|=t\cdot k_i$). We then partition $\overline{U}_i$ into $U_i^1,\ldots,U_i^{|O_i^{out}|}$, each one of size t. It is easy to verify that the generated sets $\{U_i^j\}_{i,j}$ satisfy the three conditions stated in Definition 4.

3.2 Laminar Matroids

Recall that a laminar matroid $\mathcal{M}=(V,\mathcal{I})$ is given as a laminar family $V_1,\ldots,V_r$ along with bounds $k_1,\ldots,k_r$. Without loss of generality, we can assume that $V=V_0$ with bound $k_0=k$ (being the rank of $\mathcal{M}$) is a member in the family. Furthermore, we can also assume that each vertex $v\in V$ by itself is also a member $V_i=\{v\}$ with bound $k_i=1$ in this family.

Such a laminar matroid $\mathcal{M}$ can be naturally associated with a laminar tree T where the each tree node $T_i=(V_i,k_i)$ corresponds to V_i, and the structure of the tree reflects the inclusion relationship of the members V_i in the laminar family. In such a tree $T_0=(V,k)$ corresponds to the root of the tree.

For ease of our study, we will assume that such a tree T is binary (so in total T contains $2n-1$ nodes, with $n=|V|$). Such an assumption can be easily justified by adding more sets V_i into the laminar family with the appropriately defined bounds k_i.

In the following, the elements of $\{T_i=(V_i,k_i)\}_{0\leq i\leq 2n-2}$ will be referred as "nodes", whereas the elements of V will be referred as "vertices". We are given $t\in\mathbb{N}$. To choose the vertices that are to be added into the kernel, we employ the following greedy procedure. We

process the vertices in non-increasing order with respect to their weighted degrees. At the beginning, V' is empty. When we consider a new vertex v, if for all i such that $v \in V_i$, we have $|V' \cap V_i| < 2t \cdot k_i$, then v is added to V', otherwise v is simply ignored. This procedure is equivalent to the greedy algorithm described in described in Theorem 6 applied on $2t\mathcal{M}$.

A node T_j of the tree $\{T_i = (V_i, k_i)\}_{0 \le i \le 2n-2}$ is called *saturated* if $|V' \cap V_j| = 2t \cdot k_j$. Let O be an arbitrary solution. As in the previous subsection we will use the notations $V'_i = V_i \cap V'$, $O^{in} = V' \cap O$, and $O^{out} = O \backslash V'$. In the following, we say that a vertex or a set of vertices is "contained" in a tree node T_i if they are part of V'_i (equivalently, the leaves corresponding to these elements of V' are in the subtree of root T_i).

For every element $v \in V$, there exists a leaf T_{i_v} in the laminar tree such that $V_{i_v} = \{v\}$, and we have a unique path from the root T_0 to T_{i_v}. If a vertex $v \in O$ is not in V', it means that some node along the path from T_0 to T_{i_v} was already saturated when v was processed: the *blocking node* of v is the deepest saturated node along this path. For each node T_i, we denote by B_i the set of vertices of O that are blocked by the node T_i, and we set $b_i = |B_i|$. Then, $b_i = 0$ when T_i is not saturated. Moreover, the B_is are mutually disjoint and $\bigcup B_i = O^{out}$.

Then for each vertex $v \in O^{out}$, we construct a set $\overline{U}_v$ of at least t vertices drawn from V'. Then an arbitrary subset $U_v \subseteq \overline{U}_v$ of t vertices is retained. We will argue that the generated sets U_vs ensure the robustness.

Constructing the sets $\overline{U}_v$s for all $v \in O^{out}$

We want the constructed sets $\overline{U}_v$s to satisfy the following three properties.

- **(i)** The sets $\overline{U}_v$s are mutually disjoint and are drawn from $V' \backslash O^{in}$.
- **(ii)** For each $v \in O^{out}$ and each $u \in \overline{U}_v$, $\deg_w(u) \ge \deg_w(v)$.
- **(iii)** Choosing an arbitrary $\overline{v} \in \overline{U}_v$ for each $v \in O^{out}$, the set $S = O^{in} \cup \{\overline{v}\}_{v \in O^{out}}$ is independent in the laminar matroid $\mathcal{M}$.

The formal algorithm for constructing the sets $\{\overline{U}_v\}_{v \in O^{out}}$ is given in Algorithm 1. Here we give the intuition behind it.

To guarantee Property (i), we first mark all elements in O^{in} as unusable. Then, each $\overline{U}_v$ is chosen among the usable vertices. Once a set $\overline{U}_v$ is allocated, all its vertices will be marked as unusable.

To guarantee Property (ii), first recall that each vertex $v \in O^{out}$ has a corresponding blocking node T_i (and $v \in B_i$). By our greedy procedure to build the kernel V', we know that there exist $2t \cdot k_i$ vertices u in the set V'_i, all of whom contained in T_i and $\deg_w(u) \ge \deg_w(v)$. What we do is to choose a deepest blocking node T_i and to process one of its vertex $v \in B_i$ (Lines 6-7). As we will show later (Claim 10), such a blocking node must contain at least $2t$ usable vertices. We climb down the tree from the blocking node T_i until we reach a node T_j neither of whose child nodes contains more than t usable vertices (Lines 9-10). Recall that our tree is binary, as a result, the number of usable vertices contained in T_j is between t and $2t - 2$. All these usable vertices constitute a new set $\overline{U}_v$ and then are marked as unusable.

How to guarantee Property (iii) is the most tricky part of our algorithm. Recall that we will choose an arbitrary vertex from $\overline{U}_v$ to construct a solution S stated in (iii). Apparently we have no control over the choice of the arbitrary vertex from $\overline{U}_v$, nonetheless, we need to ensure that S does not violate any of the rank constraints k_i. What we do is to associate a variable s_i with each tree node T_i. This variable indicates how many vertices contained in T_i will *certainly* be part of S, according to O^{in} and the sets $\overline{U}_v$s that have been constructed so far. Once s_i is set to k_i, it is a warning that we should not use any more remaining usable vertices contained in T_i to construct the future sets $\overline{U}_v$.

Initially, $s_i = |V'_i \cap O^{in}|$. Each time that we have decided on a tree node T_j to form a new set U_v (Lines 9-12), we increase the value of s_j from T_j all the way up to the root (Lines 13-14). If any node T_j has its variable $s_j = k_j$, we say such a node is full-booked and we mark all its (remaining) usable vertices as unusable (Lines 15-16).

Algorithm 1 Algorithm constructing the sets $\overline{U}_v$.

1: $\forall v \in O \backslash V'$, $\overline{U}_v \leftarrow \emptyset$
2: $\forall 1 \leq i \leq 2n-2$, $s_i \leftarrow |V'_i \cap O^{in}|$
3: the elements of $V' \backslash O^{in}$ are marked as usable
4: the elements of O^{in} are marked as unusable
5: **while** there exists a set B_i which is not empty **do**
6: let T_i be one of the deepest nodes such that $B_i \neq \emptyset$
7: let $v \in B_i$ be an arbitrary vertex
8: $B_i \leftarrow B_i - v$
9: **while** T_i has a child T_j containing at least t usable vertices in V'_j **do**
10: $i \leftarrow j$
11: set $\overline{U}_v$ as the set of usable elements in V'_i
12: mark all the elements of $\overline{U}_v$ as unusable
13: **for** all nodes T_j on the path from T_i to the root of the tree **do**
14: $s_j \leftarrow s_j + 1$
15: **if** $s_j = k_j$ **then** ▷ in that case, we say that T_j is *fully-booked*
16: mark all the elements of V'_j as unusable

We want to show that Algorithm 1 manages to build the sets $\overline{U}_v$s of size at least t satisfying the aforementioned properties.

▷ Claim 8. At any time during the execution of Algorithm 1, for a saturated node T_i that is not a descendant of a fully-booked node (no node above it is fully-booked), the number of usable vertices in V'_i is at least $2t \cdot (k_i - s_i)$.

Proof. As T_i is saturated, V'_i contains exactly $2t \cdot k_i$ vertices. The unusable vertices in V'_i fall into three categories:

(i) the vertices in $V'_i \cap O^{in}$ (see Line 4), but that are not contained in any fully booked descendent node of T_i,
(ii) the vertices made unusable during the allocation of a set $\overline{U}_v$ (see Line 12), but that are not contained in any fully booked descendent node of T_i,
(iii) the vertices that are contained in a fully-booked descendent node of T_i (see Line 16).

The vertices $v_1, \ldots, v_{l_1}$ in the first category each has a contribution of $+1$ in the value of s_i. Then, we can observe that an allocated set $\overline{U}_v$ is either entirely contained in a fully-booked node or has no element at all in any fully-booked node. Let us denote $\overline{U}_{v_1}, \ldots, \overline{U}_{v_{l_2}}$ the allocated sets contained in T_i that are not contained in any fully-booked node. In addition, by construction, each set $\overline{U}_{v_j}$ contains at most $2t-2$ vertices (otherwise we would be able to go deeper in the binary tree for the allocation, see Lines 9-10, because at least one child would contain at least t usable elements). Each set $\overline{U}_{v_j}$ has a contribution of $+1$ in the value of s_i. Finally, among the fully-booked nodes in the subtree of root T_i, we consider the nodes $T_{i_1}, \ldots, T_{i_{l_3}}$ that are inclusion-wise maximal (*i.e.* the roots of the fully-booked parts of the subtree). A fully-booked subtree of root T_j has to bring a $+k_j$ contribution to s_i (otherwise it would not be fully-booked, and observe that a set $\overline{U}_v$ is included in at most one such fully-booked maximal node), and is making at most $2t \cdot k_j$ vertices of V'_i

unusable (the worst case being that T_j was also a saturated node). We also have the equality $s_i = l_1 + l_2 + \sum_{j=1}^{l_3} k_{i_j}$. As a result, we have at most $l_1 + (2t-2) \cdot l_2 + \sum_{j=1}^{l_3} 2t \cdot k_{i_j} \leq 2t \cdot s_i$ unusable vertices in V_i'. ◁

▷ Claim 9. At any time during the execution of the algorithm, for all $i \in [\![1, 2n-2]\!]$, we have $k_i \geq s_i + b_i + \sum_{T_j \text{ below } T_i} b_j$ as an invariant.

Proof. As $O \in \mathcal{I}$, these inequalities hold at the beginning of Algorithm 1. To see this, note that $s_i = |V_i \cap O^{in}|$ and $b_i + \sum_{T_j \text{ below } T_i} b_j \leq |V_i \cap O^{out}|$. For the induction step, observe that Line 6 guarantees that the node T_i selected is the only one with a non-zero b_i value in the subtree of root T_i. As a result, when the set $\overline{U}_v$ is allocated, for the nodes T_j between T_i and the allocated node, the augmentation by one of the value s is not an issue because these nodes are chosen to be non-fully-booked, *i.e.* $k_j > s_j$. For the nodes T_j above T_i, the value s_j are increased by one but as b_i was decreased by one, the total value $s_j + b_j + \sum_{T_{j'} \text{ below } T_j} b_{j'}$ remains unchanged. ◁

▷ Claim 10. If $B_i \neq \emptyset$, T_i contains at least $2t$ usable vertices. Consequently, Algorithm 1 (Lines 6-11) always build the set $\overline{U}_v$ of size at least t.

Proof. In fact, as $B_i \neq \emptyset$, b_i is still non-zero, and by Claim 9 we get $k_i - s_i \geq b_i$, so V_i' contains at least $2t$ usable vertices because of Claim 8. Then the set $\overline{U}_v$ can be built as required, containing at least t elements. ◁

The above claim lower-bounds the size of each $\overline{U}_v$. The fact that the constructed $\overline{U}_v$s are mutually disjoint follows from the algorithm (Line 12). Now we want to show that the $\overline{U}_v$s have the desired properties regarding independence and weighted degrees.

▷ Claim 11. At any time during the execution of Algorithm 1, if we build a set S by taking the elements of O^{in} and one arbitrary element in each set $\overline{U}_v$ that has already been constructed, then S is independent. Moreover, for a node T_i that does not contain only unusable vertices in V_i', any arbitrary choice of elements in the $\overline{U}_v$s will lead to the equality $|S \cap V_i'| = s_i$.

Proof. We proceed by induction. These properties are clearly satisfied at the beginning of the algorithm (because then $S = O^{in}$). Now suppose that these properties hold at some time, and then we allocate a new set $\overline{U}_{v'}$ for some $v' \in O^{out}$. Let S be made of O^{in} and an arbitrary choice for the $\overline{U}_v$s that were constructed so far (excluding $\overline{U}_{v'}$). By the induction hypothesis, $S \in \mathcal{I}$. The vertices of $\overline{U}_{v'}$ are supposed to be usable, so the nodes containing them are not fully-booked and these nodes contain usable vertices. By induction on the second part of the claim, a usable element in such a node T_j can be selected, as any choice for the other $\overline{U}_v$s will use exactly $s_j < k_j$ vertices of the laminar constraint of that node. Therefore any vertex $u \in V_{v'}$ added to S does not cause any constraint to be violated, and $S \cup \{u\} \in \mathcal{I}$. Let T_i be the node used for the allocation at Line 11 of Algorithm 1. All the nodes in the subtree of root T_i will be subsequently ignored by the algorithm, as all the nodes inside it are marked as unusable. The values s_j of the nodes T_j in that subtree are not updated by the algorithm, but it is not an issue given that the second part of the claim does not affect them. The nodes above T_i are updated, and it is true that for any vertex chosen in $\overline{U}_{v'}$, that vertex will count in the laminar inequalities for these nodes as a $+1$. This concludes the induction. ◁

▷ Claim 12. For all $v \in O \backslash V'$, for all $u \in \overline{U}_v$, it holds that $\deg_w(u) \geq \deg_w(v)$.

Proof. By construction, $\overline{U}_v \subseteq V'_i$ where T_i is the blocking node of v. As T_i is the blocking node of v, all the elements in V'_i have a larger weighted degree than v. ◁

Finally we construct the sets U_vs by choosing arbitrarily t vertices from $\overline{U}_v$. By Claims 10, 11, and 12, they satisfy the properties of robustness in Definition 4.

3.3 Transversal Matroids

Recall that a transversal matroid $\mathcal{M} = (V, \mathcal{I})$ can be represented as a bipartite graph $G = (A \cup V, E)$ and $A = \{A_1, \cdots, A_k\}$. A subset $V' \subseteq V$ is independent in $\mathcal{M}$ if and only if there is a matching where all of V' are matched to some subset of A. Let $t \in \mathbb{N}$. The matroid union $(t+k-1)\mathcal{M}$ can be regarded as making the capacity of each vertex A_i in A increased to $t+k-1$ (equivalently, create $t+k-1$ copies of each A_i and modify the edge set E accordingly). Our algorithm is as follows. Again process the vertices in non-increasing order of their weighted degrees. We start with an empty matching, and we maintain a matching throughout the execution of the algorithm. For each new vertex $v \in V$, try to find an augmenting path so that it can be matched. If we cannot find such a path, v is discarded. At any time during the execution of the algorithm, the current kernel $V' \subseteq V$ is simply the set of vertices in V that are matched in the current matching. We can observe that a vertex in V' cannot be evicted once it belongs to V'. In the following, we write $V'_i \subseteq V'$ to denote the vertices in V that are matched to A_i in the current kernel V'.

First we argue that our procedure is the same as the greedy described in Theorem 6 applied on $(t+k-1)\mathcal{M}$. We need to show that a vertex v, if discarded, is spanned by vertices in V' that arrived earlier than it. To see this, observe that, at the moment v arrives, V' is independent in $(t+k-1)\mathcal{M}$. Moreover, as we cannot find an augmenting path when v is added to V', is means that $V'+v$ is not independent, *i.e.* v is spanned by V' and this holds until the end of the algorithm.

Now let us consider the robustness. Let $O = \{o_1, \ldots, o_k\}$ be an arbitrary base in $\mathcal{M}$. We can assume that o_i is assigned to A_i for all i in the corresponding matching. For an element $o_i \in O \backslash V'$, when it was discarded, some $V'_i \subseteq V'$ elements (exactly $t+k-1$) that arrived earlier were already assigned to A_i. As no augmenting path could go through A_i when o_i was discarded, the set of elements assigned to A_i would not have changed till the end: otherwise, that would mean that at some point an augmenting path passed through A_i, which is not possible as o_i was discarded because no augmenting path passing through A_i was found at that point. As a result the $t+k-1$ elements of V'_i assigned to A_i are all of weighted degrees larger than that of o_i. As $|V'_i \cap O| \leq |V' \cap O| < k$ there remains at least t elements of V'_i that can be used to build a set U_{o_i} of cardinality t as stated in Definition 4.

4 Steaming Algorithms

In this section, we turn our algorithms into streaming form.

First, we show that it is easy compute a $(1-\varepsilon)$ approximation in two passes, using $O(n+\tau^2)$ space (τ depends on the type of matroids involved, as defined in Theorem 6). In the first pass, we compute the weighted degrees $\deg_w(v)$ of all vertices v to define the kernel $V' \subseteq V$. This requires $O(n)$ space. In the second pass, we retain a subset of edges $E' \subseteq E$, those both of whose end-points are in V'. Easily $|E'| = O(\tau^2)$. Using E', we can compute the exact value $E_G(S) = \sum_{v \in S} \deg_w(v) - \sum_{e \in (S \times S) \cap E'} w(e)$ for each feasible independent set $S \subseteq V'$. Then an enumeration of all such sets gives the desired $(1-\varepsilon)$ approximation.

We now explain how to achieve the same goal in one pass, at the expense of higher space requirement.

▶ **Theorem 13.** *In the edge arrival steaming model (each edge appearing exactly once in the stream), one can extract a $(1-\varepsilon)$-approximate solution of the matroid-constrained maximum vertex cover using $O(\frac{nk}{\varepsilon})$ variables for uniform, partition, laminar, and transversal matroids.*

Proof. Let $\varepsilon > 0$. During the streaming phase, we keep track of the weighted degrees of all the vertices, as well as for each vertex v the set of the $\frac{2k}{\varepsilon}$ edges incident to v that have the largest weight. We denote the set of memorized edges as E'.

Then, we can choose, depending on the type of matroid, the value τ corresponding to the right type of matroid (as prescribed in Theorem 6) for the parameter $\frac{\varepsilon}{2}$ and we build the kernel V' that is supposed to contain a $(1-\frac{\varepsilon}{2})$ approximation of the maximum cover. However, we do not know all the edges between the elements in V', as only the $\frac{2k}{\varepsilon}$ heaviest incident edges are known for each vertex.

We will compute the value of S *pretending* that the edges in $((S\times S)\cap E)\backslash E'$ are not present. Precisely, for each set $S\subseteq V$, we define

$$\tilde{E}_G(S) = \sum_{v\in S}\deg_w(v) - \sum_{e\in(S\times S)\cap E'} w(e) = E_G(S) + \sum_{e\in((S\times S)\cap E)\setminus E'} w(e).$$

Notice that $\tilde{E}_G(S)\geq E_G(S)$. Let $S^*\subseteq V'$ be the independent set reaching the maximum $\tilde{E}_G(S^*)$. This set S^* will be our final output. We next lower-bound its real value $E_G(S^*)$.

Let O denote the original optimal solution (with respect to the entire graph), and S' denote the optimal vertex cover in the kernel V' (also with respect to the entire graph), so that $S'\subseteq V'$, $S'\in\mathcal{I}$, and $E_G(S')\geq(1-\frac{\varepsilon}{2})\cdot E_G(O)$. Then

$$\tilde{E}_G(S^*)\geq\tilde{E}_G(S')\geq E_G(S')\geq\left(1-\frac{\varepsilon}{2}\right)\cdot E_G(O).$$

To compare the real value of $E_G(S^*)$ with $\tilde{E}_G(S^*)$, we just need to compute the total weight of the edges in $((S^*\times S^*)\cap E)\backslash E'$:

$$\begin{aligned}\sum_{(u,v)\in((S^*\times S^*)\cap E)\setminus E'} w(u,v) &= \frac{1}{2}\sum_{v\in S^*}\left(\sum_{u\in S^*:(u,v)\in E\setminus E'} w(u,v)\right)\\ &\leq \frac{1}{2}\sum_{v\in S^*} k\cdot\deg_w(v)\cdot\frac{\varepsilon}{2k}\\ &= \frac{\varepsilon}{4}\sum_{v\in S^*}\deg_w(v)\leq\frac{\varepsilon}{2}\cdot\tilde{E}_G(S^*),\end{aligned}$$

where the first inequality comes from the fact that the edges that are not among the $\frac{2k}{\varepsilon}$ heaviest edges incident on v must be of weight at most $\deg_w(v)\cdot\frac{\varepsilon}{2k}$. Therefore the real value $E_G(S^*)$ is at least $(1-\frac{\varepsilon}{2})\cdot\tilde{E}_G(S^*)\geq(1-\frac{\varepsilon}{2})^2\cdot E_G(O)\geq(1-\varepsilon)\cdot E_G(O)$. ◀

Next we consider a particular kind of stream of edges, where each edge appears twice: given an arbitrary order of the vertices, for each vertex, all its incident edges are given in a row. For this *incidence streaming model* [5] (sometimes called *adjacency list model* [29]), the next theorem shows that we can use much less space with just a single pass.

▶ **Theorem 14.** *In the incidence streaming model, one can extract a $(1-\varepsilon)$-approximate solution using $O((\frac{k}{\varepsilon})^2)$ variables for uniform, partition, laminar, and $O((\frac{k}{\varepsilon}+k)^2)$ for transversal matroids.*

Proof. Let $\varepsilon > 0$. Given the type of the matroid $\mathcal{M}$, choose the corresponding value of τ as prescribed in Theorem 6. Start with an empty kernel $V' = \emptyset$. Through the execution of the algorithm, V' will contain the largest independent set in $\tau\mathcal{M}$ with respect to the sum of the weighted degrees. When we process a vertex v (*i.e.* its set of incident edges) we can compute its weighted degree $\deg_w(v)$ and store the edges linking v to elements of V'. If $V' + v$ is not independent in $\tau\mathcal{M}$, consider the element with the smallest weighted degree u in the circuit formed in $V' + v$. Then, set $V' \leftarrow (V' + v) - u$. If v is added into V', we keep in memory all the edges linking v to other vertices of V'. When an element is discarded or evicted from V', all its incident edges are deleted. As a result, at any time during the execution of the algorithm, only $O(\tau^2)$ edges are stored, so the the overall memory consumption is $O(\tau^2)$.

In the end, we obtain exactly the approximate kernels described in the previous sections, and because we know all the values of the weighted degrees of V' as well as the weights of the edges between them we can find the largest vertex cover in that kernel using bruteforce. ◀

▶ Remark 15. This model has an interesting interpretation in the context of coverage function[4] maximization in the streaming setting – here the sets arrive over time in such a way that the values of singletons $f(\{v\})$, for $v \in V$, are revealed one by one. In case where a coverage function has bounded frequency larger than 2, we also present in Appendix B a streaming algorithm.

5 Conclusion and Open Questions

Theorem 6 allows us to generalize the cardinality constraint to some special cases of matroid constraints, and these ideas could be useful for other kernelization algorithms. Regarding the bounds of Theorem 6, tight examples can be built to show that the values of τ provided are the best possible for partition and laminar matroids. For transversal matroids it is less clear whether the bound for τ can be improved or not. The most important open question is obviously whether Theorem 6 can be generalized to other types of matroids.

References

1 Alexander A. Ageev and Maxim I. Sviridenko. Approximation algorithms for maximum coverage and max cut with given sizes of parts. In *IPCO 1999*, 1999.

2 Per Austrin and Aleksa Stankovic. Global cardinality constraints make approximating some max-2-csps harder. In *APPROX/RANDOM 2019*, pages 24:1–24:17, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.24.

3 M. Babaioff, N. Immorlica, and R. Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA*, pages 434–443, 2007.

4 Édouard Bonnet, Bruno Escoffier, Vangelis Th. Paschos, and Georgios Stamoulis. Purely combinatorial approximation algorithms for maximum k-vertex cover in bipartite graphs. *Discrete Optimization*, 27:26–56, 2018.

[4] A coverage function f over a ground set $\{1, \ldots, m\}$, associated with a universe U of weighted elements and m sets $A_1, \ldots, A_m$, where $A_i \subseteq U$ for all i, is defined over all $S \subseteq \{1, \ldots, m\}$ so that $f(S)$ is the sum of the weight of the elements in $\cup_{i \in S} A_i$. The frequency of an element of the universe is the number of sets A_i it appears in. Here in our problem of maximum vertex cover, the vertices correspond to the ground set and the edges to the universe U. Note that for a vertex cover the frequency (the maximum number of sets where an element of the universe appears in) is exactly 2, as an edge has only two endpoints.

5 Vladimir Braverman, Zaoxing Liu, Tejasvam Singh, N. V. Vinodchandran, and Lin F. Yang. New bounds for the CLIQUE-GAP problem using graph decomposition theory. *Algorithmica*, 80(2):652–667, 2018. doi:10.1007/s00453-017-0277-5.

6 Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In *Proc. 12th IPCO*, pages 182–196, 2007. doi:10.1007/978-3-540-72792-7_15.

7 Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011.

8 Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: Matchings, matroids and more. In *Proc. 17th IPCO*, pages 210–221, 2014.

9 S. Chakraborty and O. Lachish. Improved competitive ratio for the matroid secretary problem. In *SODA*, pages 1702–1712, 2012.

10 Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *Proc. 42nd ICALP*, pages 318–330, 2015.

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

12 William H Cunningham. Testing membership in matroid polyhedra. *Journal of Combinatorial Theory, Series B*, 36(2):161–188, 1984.

13 J. Edmonds and D.R. Fulkerson. Transversals and matroid partition. *Journal of Research National Bureau of Standards Section B*, 69:147–153, 1965.

14 Uriel Feige and Michael Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *Journal of Algorithms*, 41(2):174–211, 2001.

15 M. Feldman, O. Svensson, and R. Zenklusen. A simple $o \log \log(rank))$-competitive al- gorithm for the matroid secretary problem. In *SODA*, pages 1189–1201, 2015.

16 Moran Feldman, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. Streaming submodular maximization with matroid and matching constraints. *CoRR*, abs/2107.07183, 2021. arXiv:2107.07183.

17 Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020. doi:10.3390/a13060146.

18 Yuval Filmus and Justin Ward. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. In *Proc. 53nd FOCS*, 2012.

19 Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. Parameterized complexity of generalized vertex cover problems. In *Algorithms and Data Structures*, pages 36–48, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

20 Anupam Gupta, Euiwoong Lee, and Jason Li. Faster exact and approximate algorithms for k-cut. In *FOCS 2018*, pages 113–123. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00020.

21 Qiaoming Han, Yinyu Ye, Hantao Zhang, and Jiawei Zhang. On approximation of max-vertex-cover. *Eur. J. Oper. Res.*, 143(2):342–355, 2002. doi:10.1016/S0377-2217(02)00330-2.

22 Qiaoming Han, Yinyu Ye, and Jiawei Zhang. An improved rounding method and semidefinite programming relaxation for graph partition. *Math. Program.*, 92(3):509–535, 2002. doi:10.1007/s101070100288.

23 Dorit S Hochbaum and Anu Pathria. Analysis of the greedy approach in covering problems. *Naval Research Quarterly*, 45:615–627, 1998.

24 S. Im and Y. Wang. Secretary problems: laminar matroids and interval scheduling. In *SODA*, pages 1265–1274, 2011.

25 Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. In *STOC 2017*, pages 224–237. ACM, 2017. doi:10.1145/3055399.3055456.

26 Pasin Manurangsi. A Note on Max k-Vertex Cover: Faster FPT-AS, Smaller Approximate Kernel and Improved Approximation. In *SOSA 2019*, pages 15:1–15:21, 2018.

27 Pasin Manurangsi. Tight running time lower bounds for strong inapproximability of maximum k-coverage, unique set cover and related problems (via t-wise agreement testing theorem). In *SODA*, pages 62–81, 2020.

28 Dániel Marx. Parameterized Complexity and Approximation Algorithms. *The Computer Journal*, 51(1):60–78, July 2008.

29 Andrew McGregor, David Tench, and Hoa T. Vu. Maximum coverage in the data stream model: Parameterized and generalized. In *ICDT 2021*, volume 186, pages 12:1–12:20, 2021. doi:10.4230/LIPIcs.ICDT.2021.12.

30 L. Mirsky and H. Perfect. Applications of the notion of independence to problems of combinatorial analysis. *Journal of Combinatorial Theory*, 2:327–357, 1965.

31 S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Now Publishers Inc, January 2005.

32 András Recski. *Matroid Theory and its Applications in Electric Network Theory and in Statics*. Springer-Verlag, 1989.

33 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Springer, 2003.

34 J. Soto. Matroid secretary problem in the random assignment model. *SIAM Journal on Computing*, 42:178–211, 2013.

35 D.J.A Welsh. Transversal theory and matroids. *Canadian Journal of Mathematics*, 21:1323–1302, 1969.

A An FPT-AS for Hypergraphs

Suppose that we are given a hypergraph $G = (V, E)$ with edge size bounded by a constant $\eta \geq 2$. We proceed as in Section 2. First, notice that:

$$E_G(S) = E_G(O^{in}) + E_G(U^*) - E_G(O^{in}, U^*).$$

We bound $\mathbb{E}[E_G(O^{in}, U^*)]$ as follows. By construction, $\mathbb{P}[u \in U^*] = \varepsilon$ for all $u \in U$. Then,

$$\begin{aligned}\mathbb{E}[E_G(O^{in}, U^*)] = \sum_{e \in E, e \cap O^{in} \neq \emptyset} w(e) \cdot \mathbb{1}[e \cap U^* \neq \emptyset] &\leq \sum_{e \in E, e \cap O^{in} \neq \emptyset} w(e) \cdot (\eta - 1) \cdot \varepsilon \\ &= \varepsilon \cdot (\eta - 1) \cdot \mathbb{E}[E_G(O^{in})].\end{aligned}$$

using union bound and the fact that at most $\eta - 1$ endpoints can be in U. Furthermore, the value $\mathbb{E}[E_G(U^*)]$ can be rearranged as follows:

$$\begin{aligned}\mathbb{E}[E_G(U^*)] &= \mathbb{E}\left[\sum_{u \in U^*} \left(\deg_w(u) - \sum_{e \in \delta(u)} w(e) \cdot \frac{|e \cap U^*| - 1}{|e \cap U^*|}\right)\right] \\ &\geq \mathbb{E}\left[\sum_{u \in U^*} \left(\deg_w(u) - \sum_{e \in \delta(u)} w(e) \cdot \frac{\eta - 1}{\eta} \cdot \mathbb{1}[e \cap U^* \backslash \{u\} \neq \emptyset]\right)\right] \\ &\geq \mathbb{E}\left[\sum_{u \in U} \left(\deg_w(u) \cdot \mathbb{1}[u \in U^*] - \sum_{e \in \delta(u)} w(e) \cdot \frac{\eta - 1}{\eta} \cdot \mathbb{1}[u \in U^* \wedge e \cap U^* \backslash \{u\} \neq \emptyset]\right)\right] \\ &\geq \mathbb{E}\left[\sum_{u \in U} \left(\deg_w(u) \cdot \varepsilon - \frac{\eta - 1}{\eta} \sum_{e \in \delta(u)} w(e) \cdot (\eta - 1) \cdot \varepsilon^2\right)\right]\end{aligned}$$

$$\geq \varepsilon \cdot (1 - \varepsilon \cdot (\eta - 1)) \left(\sum_{u \in U} \deg_w(u) \right).$$

Recall that by robustness, for all $u \in O^{out}$, the elements of U_u have weighted degree no less than the one of u. Therefore,

$$\begin{aligned}
\mathbb{E}[E_G(U^*)] &\geq \varepsilon(1 - \varepsilon \cdot (\eta - 1)) \left(\sum_{u \in O^{out}} \sum_{v \in U_u} \deg_w(v) \right) \\
&\geq \varepsilon(1 - \varepsilon \cdot (\eta - 1)) \left(\sum_{u \in O^{out}} \frac{1}{\varepsilon} \cdot \deg_w(u) \right) \\
&\geq (1 - \varepsilon \cdot (\eta - 1)) \cdot E_G(O^{out}).
\end{aligned}$$

As a result, we get:

$$\begin{aligned}
\mathbb{E}[E_G(S)] &\geq E_G(O^{in}) + (1 - \varepsilon \cdot (\eta - 1)) \cdot E_G(O^{out}) - \varepsilon \cdot (\eta - 1) \cdot E_G(O^{in}) \\
&\geq (1 - \varepsilon \cdot (\eta - 1)) \cdot E_G(O).
\end{aligned}$$

By averaging principle, there exists a set $S \subseteq V'$ such that $S \in \mathcal{I}$ and such that $E_G(S) \geq (1 - (\eta - 1) \cdot \varepsilon) \cdot E_G(O)$.

B Streaming Algorithm for Hypergraphs

Here we suppose that we are given a hypergraph $G = (V, E)$ with edge size bounded by a constant $\eta \geq 2$ as an adjacency list stream. Using the idea of Theorem 14 and the result of Appendix A, one can get in the incidence streaming model a $(1 - (\eta - 1) \cdot \varepsilon)$ approximation using $O(\tau^\eta)$ memory, where τ depends on the type of matroid, as prescribed in Theorem 6. In fact, we can maintain through the execution of the algorithm for each subset of at most η elements $S \subseteq V'$ the sum of the weight of the hyper-edges e such that $e \cap V' = S$ (just like in the proof of Theorem 14 where we keep track of these values for pairs in V'). For instance, for a uniform, a partition, or a laminar matroid, we could get a $(1 - \varepsilon)$ approximation using $O((\frac{2\eta \cdot k}{\varepsilon})^\eta)$ variables. This shows that for the special matroids that we studied of rank k, a weighted coverage function with bounded frequency η can be $(1 - \varepsilon)$ approximated in streaming. This extends a result of [29] to matroids.