



# Semi-streaming Algorithms for Submodular Matroid Intersection

Paritosh Garg<sup>(✉)</sup>, Linus Jordan, and Ola Svensson

EPFL, Lausanne, Switzerland

{paritosh.garg, ola.svensson}@epfl.ch, linus.jordan@bluewin.ch

**Abstract.** While the basic greedy algorithm gives a semi-streaming algorithm with an approximation guarantee of 2 for the *unweighted* matching problem, it was only recently that Paz and Schwartzman obtained an analogous result for weighted instances. Their approach is based on the versatile local ratio technique and also applies to generalizations such as weighted hypergraph matchings. However, the framework for the analysis fails for the related problem of weighted matroid intersection and as a result, the approximation guarantee for weighted instances did not match the factor 2 achieved by the greedy algorithm for unweighted instances. Our main result closes this gap by developing a semi-streaming algorithm with an approximation guarantee of  $2 + \varepsilon$  for *weighted* matroid intersection, improving upon the previous best guarantee of  $4 + \varepsilon$ . Our techniques also allow us to generalize recent results by Levin and Wajc on submodular maximization subject to matching constraints to that of matroid-intersection constraints.

While our algorithm is an adaptation of the local ratio technique used in previous works, the analysis deviates significantly and relies on structural properties of matroid intersection, called kernels. Finally, we also conjecture that our algorithm gives a  $(k + \varepsilon)$  approximation for the intersection of  $k$  matroids but prove that new tools are needed in the analysis as the used structural properties fail for  $k \geq 3$ .

## 1 Introduction

For large problems, it is often not realistic that the entire input can be stored in random access memory so more memory efficient algorithms are preferable. A popular model for such algorithms is the (semi-)streaming model (see e.g. [13]): the elements of the input are fed to the algorithm in a stream and the algorithm is required to have a small memory footprint.

---

This research was supported by the Swiss National Science Foundation project 200021-184656 “Randomness in Problem Instances and Randomized Algorithms.”

© Springer Nature Switzerland AG 2021

M. Singh and D. P. Williamson (Eds.): IPCO 2021, LNCS 12707, pp. 208–222, 2021.

[https://doi.org/10.1007/978-3-030-73879-2\\_15](https://doi.org/10.1007/978-3-030-73879-2_15)

Consider the classic maximum matching problem in an undirected graph  $G = (V, E)$ . An algorithm in the semi-streaming model<sup>1</sup> is fed the edges one-by-one in a stream  $e_1, e_2, \dots, e_{|E|}$  and at any point of time the algorithm is only allowed  $O(|V| \text{polylog}(|V|))$  bits of storage. The goal is to output a large matching  $M \subseteq E$  at the end of the stream. Note that the allowed memory usage is sufficient for the algorithm to store a solution  $M$  but in general it is much smaller than the size of the input since the number of edges may be as many as  $|V|^2/2$ . Indeed, the intuitive difficulty in designing a semi-streaming algorithm is that the algorithm needs to discard many of the seen edges (due to the memory restriction) without knowing the future edges and still return a good solution at the end of the stream.

For the unweighted matching problem, the best known semi-streaming algorithm is the basic greedy approach:

Initially, let  $M = \emptyset$ . Then for each edge  $e$  in the stream, add it to  $M$  if  $M \cup \{e\}$  is a feasible solution, i.e., a matching; otherwise the edge  $e$  is discarded.

The algorithm uses space  $O(|V| \log |V|)$  and a simple proof shows that it returns a 2-approximate solution in the *unweighted* case, i.e., a matching of size at least half the size of an maximum matching. However, this basic approach fails to achieve any approximation guarantee for *weighted graphs*.

Indeed, for weighted matchings, it is non-trivial to even get a small constant-factor approximation. One way to do so is to replace edges if we have a much heavier edge. This is formalized in [6] who get a 6-approximation. Later, [12] improved this algorithm to find a 5.828-approximation; and, with a more involved technique, [4] provided a  $(4 + \varepsilon)$ -approximation.

It was only in recent breakthrough work [14] that the gap in the approximation guarantee between unweighted and weighted matchings was closed. Specifically, [14] gave a semi-streaming algorithm for weighted matchings with an approximation guarantee of  $2 + \varepsilon$  for every  $\varepsilon > 0$ . Shortly after, [9] came up with a simplified analysis of their algorithm, reducing the memory requirement from  $O_\varepsilon(|V| \log^2 |V|)$  to  $O_\varepsilon(|V| \log |V|)$ . These results for weighted matchings are tight (up to the  $\varepsilon$ ) in the sense that any improvement would also improve the state-of-the-art in the unweighted case, which is a long-standing open problem.

The algorithm of [14] is an elegant use of the local ratio technique ([1] ([2]) in the semi-streaming setting. While this technique is very versatile and it readily generalizes to weighted hypergraph matchings, it is much harder to use it for the related problem of weighted matroid intersection. This is perhaps surprising as many of the prior results for the matching problem also applies to the matroid intersection problem in the semi-streaming model (see Sect. 2 for definitions). Indeed, the greedy algorithm still returns a 2-approximate solution

<sup>1</sup> This model can also be considered in the multi-pass setting when the algorithm is allowed to take several passes over the stream. However, in this work we focus on the most basic and widely studied setting in which the algorithm takes a single pass over the stream.

in the unweighted case and the algorithm in [4] returns a  $(4 + \varepsilon)$ -approximate solution for weighted instances. So, prior to our work, the status of the matroid intersection problem was that of the matching problem *before* [14].

We now describe on a high-level the reason that the techniques from [14] are not easily applicable to matroid intersection and our approach for dealing with this difficulty. The approach in [14] works in two parts, first certain elements of the stream are selected and added to a set  $S$ , and then at the end of the stream a matching  $M$  is computed by the greedy algorithm that inspects the edges of  $S$  in the reverse order in which they were added. This way of constructing the solution  $M$  greedily by going backwards in time is a standard framework for analyzing algorithms based on the local ratio technique. Now in order to adapt their algorithm to matroid intersection, recall that the bipartite matching problem can be formulated as the intersection of two partition matroids. We can thus reinterpret their algorithm and analysis in this setting. Furthermore, after this reinterpretation, it is not too hard to define an algorithm that works for the intersection of any two matroids. However, bipartite matching is a *special* case of matroid intersection which captures a rich set of seemingly more complex problems. This added expressiveness causes the analysis and the standard framework for analyzing local ratio algorithms to fail. Specifically, we prove that a solution formed by running the greedy algorithm on  $S$  in the reverse order (as done for the matching problem) fails to give any constant-factor approximation guarantee for the matroid intersection problem. To overcome this and to obtain our main result, we make a connection to a concept called matroid kernels (see [7] for more details about kernels), which allows us to, in a more complex way, identify a subset of  $S$  with an approximation guarantee of  $2 + \varepsilon$ .

Finally, for the intersection of more than two matroids, the same approach in the analysis does not work, because the notion of matroid kernel does not generalize to more than two matroids. However, we conjecture that the subset  $S$  generated for the intersection of  $k$  matroids still contains a  $(k + \varepsilon)$ -approximation. Currently, the best approximation results are a  $(k^2 + \varepsilon)$ -approximation from [4] and a  $(2(k + \sqrt{k(k-1)}) - 1)$ -approximation from [3]. For  $k = 3$ , the former is better, giving a  $(9 + \varepsilon)$ -approximation. For  $k > 3$ , the latter is better, giving an  $O(k)$ -approximation.

*Generalization to Submodular Functions.* Very recently, Levin and Wajc [11] obtained improved approximation ratios for matching and b-matching problems in the semi-streaming model with respect to submodular functions. Specifically, they get a  $(3 + 2\sqrt{2})$ -approximation for monotone submodular b-matching,  $(4 + 3\sqrt{2})$ -approximation for non-monotone submodular matching, and a  $(3 + \varepsilon)$ -approximation for maximum weight (linear) b-matching. In our paper, we are able to extend our algorithm for weighted matroid intersection to work with submodular functions by combining our and their ideas. In fact, we are able to generalize all their results to the case of matroid intersection with better or equal<sup>2</sup> approximation ratios: we get  $(3 + 2\sqrt{2} + \delta)$ -approximation for monotone

---

<sup>2</sup> One can get rid of the  $\delta$  factor if we assume that the function value is polynomially bounded by  $|E|$ , an assumption made by [11].

submodular matroid intersection,  $(4 + 3\sqrt{2} + \delta)$ -approximation for non-monotone submodular matroid intersection and  $(2 + \varepsilon)$ -approximation for maximum weight (linear) matroid intersection. Due to space limitations, we refer the reader to the full version [8] of our paper for this generalization.

*Outline.* In Sect. 2, we introduce basic matroid concepts and we formally define the weighted matroid intersection problem in the semi-streaming model. Section 3 is devoted to our main result. Here, we adapt the algorithm of [14] without worrying about the memory requirements, show why the standard analysis fails, and then give our new analysis to get a 2-approximation. We then make the obtained algorithm memory efficient in Sect. 4. Finally, in Sect. 5, we discuss the case of more than two matroids.

## 2 Preliminaries

*Matroids.* We define and give a brief overview of the basic concepts related to matroids that we use in this paper. For a more comprehensive treatment, we refer the reader to [15]. A *matroid* is a tuple  $M = (E, I)$  consisting of a finite ground set  $E$  and a family  $I \subseteq 2^E$  of subsets of  $E$  satisfying:

- if  $X \subseteq Y, Y \in I$ , then  $X \in I$ ; and
- if  $X \in I, Y \in I$  and  $|Y| > |X|$ , then  $\exists e \in Y \setminus X$  such that  $X \cup \{e\} \in I$ .

The elements in  $I$  (that are subsets of  $E$ ) are referred to as the *independent sets* of the matroid and the set  $E$  is referred to as the *ground set*. With a matroid  $M = (E, I)$ , we associate the *rank function*  $\text{rank}_M : 2^E \rightarrow \mathbb{N}$  and the *span function*  $\text{span}_M : 2^E \rightarrow 2^E$  defined as follows for every  $E' \subseteq E$ ,

$$\begin{aligned} \text{rank}_M(E') &= \max\{|X| \mid X \subseteq E' \text{ and } X \in I\}, \\ \text{span}_M(E') &= \{e \in E \mid \text{rank}_M(E' \cup \{e\}) = \text{rank}_M(E')\}. \end{aligned}$$

We simply write  $\text{rank}(\cdot)$  and  $\text{span}(\cdot)$  when the matroid  $M$  is clear from the context. In words, the rank function equals the size of the largest independent set when restricted to  $E'$  and the span function equals the elements in  $E'$  and all elements that cannot be added to a maximum cardinality independent set of  $E'$  while maintaining independence. The *rank of the matroid* equals  $\text{rank}(E)$ , i.e., the size of the largest independent set.

*The Weighted Matroid Intersection Problem in the Semi-Streaming Model.* In the *weighted matroid intersection problem*, we are given two matroids  $M_1 = (E, I_1), M_2 = (E, I_2)$  on a common ground set  $E$  and a non-negative weight function  $w : E \rightarrow \mathbb{R}_{\geq 0}$  on the elements of the ground set. The goal is to find a subset  $X \subseteq E$  that is independent in both matroids, i.e.,  $X \in I_1$  and  $X \in I_2$ , and whose weight  $w(X) = \sum_{e \in X} w(e)$  is maximized.

In seminal work [5], Edmonds gave a polynomial-time algorithm for solving the weighted matroid intersection problem to optimality in the classic model of

computation when the whole input is available to the algorithm throughout the computation. In contrast, the problem becomes significantly harder and tight results are still eluding us in the semi-streaming model where the memory footprint of the algorithm and its access pattern to the input are restricted. Specifically, in the *semi-streaming* model the ground set  $E$  is revealed in a stream  $e_1, e_2, \dots, e_{|E|}$  and at time  $i$  the algorithm gets access to  $e_i$  and can perform computation based on  $e_i$  and its current memory but without knowledge of future elements  $e_{i+1}, \dots, e_{|E|}$ . The algorithm has independence-oracle access to the matroids  $M_1$  and  $M_2$  restricted to the elements stored in the memory, i.e., for a set of such elements, the algorithm can query whether the set is independent in each matroid. The goal is to design an algorithm such that (i) the memory usage is near-linear  $O((r_1 + r_2) \text{polylog}(r_1 + r_2))$  at any time, where  $r_1$  and  $r_2$  denote the ranks of the input matroids  $M_1$  and  $M_2$ , respectively, and (ii) at the end of the stream the algorithm should output a feasible solution  $X \subseteq E$ , i.e., a subset  $X$  that satisfies  $X \in I_1$  and  $X \in I_2$ , of large weight  $w(X)$ . We remark that the memory requirement  $O((r_1 + r_2) \text{polylog}(r_1 + r_2))$  is natural as  $r_1 + r_2 = |V|$  when formulating a bipartite matching problem as the intersection of two matroids<sup>3</sup>.

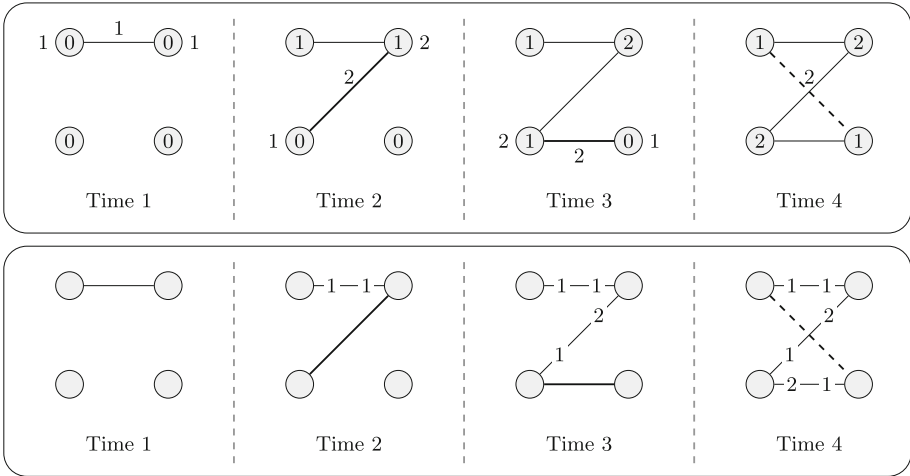
The difficulty in designing a good semi-streaming algorithm is that the memory requirement is much smaller than the size of the ground set  $E$  and thus the algorithm must intuitively discard many of the elements without knowledge of the future and without significantly deteriorating the weight of the final solution  $X$ . The quality of the algorithm is measured in terms of its approximation guarantee: an algorithm is said to have an *approximation guarantee* of  $\alpha$  if it is guaranteed to output a solution  $X$ , no matter the input and the order of the stream, such that  $w(X) \geq \text{OPT}/\alpha$  where  $\text{OPT}$  denotes the weight of an optimal solution to the instance. As aforementioned, our main result in this paper is a semi-streaming algorithm with an approximation guarantee of  $2 + \varepsilon$ , for every  $\varepsilon > 0$ , improving upon the previous best guarantee of  $4 + \varepsilon$  [4].

### 3 The Local Ratio Technique for Weighted Matroid Intersection

In this section, we first present the local ratio algorithm for the weighted matching problem that forms the basis of the semi-streaming algorithm in [14]. We then adapt it to the weighted matroid intersection problem. While the algorithm is fairly natural to adapt to this setting, we give an example in Sect. 3.2 that shows that the same techniques as used for analyzing the algorithm for

<sup>3</sup> The considered problem can also be formulated as the problem of finding an independent set in one matroid, say  $M_1$ , and maximizing a submodular function which would be the (weighted) rank function of  $M_2$ . For that problem, [10] recently gave a streaming algorithm with an approximation guarantee of  $(2 + \varepsilon)$ . However, the space requirement of their algorithm is exponential in the rank of  $M_1$  (which would correspond to be exponential in  $|V|$  in the matching case) and thus it does not provide a meaningful guarantee for our setting.

matchings does not work for matroid intersection. Instead, our analysis, which is presented in Sect. 3.3, deviates from the standard framework for analyzing local ratio algorithms and it heavily relies on a structural property of matroid intersection known as kernels. We remark that the algorithms considered in this section do not have a small memory footprint.



**Fig. 1.** The top part shows an example execution of the local ratio technique for weighted matchings. The bottom part shows how to adapt this (bipartite) example to the language of weighted matroid intersection (Algorithm 1).

### 3.1 Local-Ratio Technique for Weighted Matching

The local ratio algorithm for the weighted matching problem is as follows. The algorithm maintains vertex potentials  $w(u)$  for every vertex  $u$ , a set  $S$  of selected edges, and an auxiliary weight function  $g : S \rightarrow \mathbb{R}_{\geq 0}$  of the selected edges. Initially the vertex potentials are set to 0 and the set  $S$  is empty. When an edge  $e = \{u, v\}$  arrives, the algorithm computes how much it gains compared to the previous edges, by taking its weight minus the weight/potential of its endpoints ( $g(e) = w(e) - w(u) - w(v)$ ). If the gain is positive, then we add the edge to  $S$ , and add the gain to the weight of the endpoints, that is, we set  $w(u) = w(u) + g(e)$  and  $w(v) = w(v) + g(e)$ . At the end, we return a maximum weight matching  $M$  among the edges stored on the stack  $S$ .

For a better intuition of the algorithm, consider the example depicted on the top of Fig. 1. The stream consists of four edges  $e_1, e_2, e_3, e_4$  with weights  $w(e_1) = 1$  and  $w(e_2) = w(e_3) = w(e_4) = 2$ . At each time step  $i$ , we depict the arriving edge  $e_i$  in thick along with its weight; the vertex potentials before the algorithm considers this edge is written on the vertices, and the updated vertex potentials (if any) after considering  $e_i$  are depicted next to the incident vertices. The edges that are added to  $S$  are solid and those that are not added to  $S$  are dashed.

---

**Algorithm 1.** Local ratio for matroid intersection

---

**Input:** A stream of the elements of the common ground set of matroids  $M_1 = (E, I_1), M_2 = (E, I_2)$ .**Output:** A set  $X \subseteq E$  that is independent in both matroids. $S \leftarrow \emptyset$ **for** element  $e$  in the stream **do**    calculate  $w_i^*(e) = \max(\{0\} \cup \{\theta : e \in \text{span}_{M_i}(\{f \in S \mid w_i(f) \geq \theta\})\})$  for  $i \in \{1, 2\}$ .    **if**  $w(e) > w_1^*(e) + w_2^*(e)$  **then**         $g(e) \leftarrow w(e) - w_1^*(e) - w_2^*(e)$          $w_1(e) \leftarrow w_1^*(e) + g(e)$          $w_2(e) \leftarrow w_2^*(e) + g(e)$          $S \leftarrow S \cup \{e\}$     **end if****end for****return** a maximum weight set  $T \subseteq S$  that is independent in  $M_1$  and  $M_2$ 

---

At the arrival of the first edge of weight  $w(e_1) = 1$ , both incident vertices have potential 0 and so the algorithm adds this edge to  $S$  and increases the incident vertex potentials with the gain  $g(e_1) = 1$ . For the second edge of weight  $w(e_2) = 2$ , the sum of incident vertex potentials is 1 and so the gain of  $e_2$  is  $g(e_2) = 2 - 1$ , which in turn causes the algorithm to add this edge to  $S$  and to increase the incident vertex potentials by 1. The third time step is similar to the second. At the last time step, edge  $e_4$  arrives of weight  $w(e_4) = 2$ . As the incident vertex potentials sum up to 2 the gain of  $e_4$  is not strictly positive and so this edge is *not* added to  $S$  and no vertex potentials are updated. Finally, the algorithm returns the maximum weight matching in  $S$  which in this case consists of edges  $\{e_1, e_3\}$  and has weight 3. Note that the optimal matching of this instance had weight 4 and we thus found a  $4/3$ -approximate solution.

In general, the algorithm has an approximation guarantee of 2. This is proved using a common framework to analyze algorithms based on the local ratio technique: We ignore the weights and greedily construct a matching  $M$  by inspecting the edges in  $S$  in reverse order, i.e., we first consider the edges that were added last. An easy proof (see e.g. [9]) then shows that the matching  $M$  constructed in this way has weight at least half the optimum weight.

In the next section, we adapt the above described algorithm to the context of matroid intersections. We also give an example that the above framework for the analysis fails to give any constant-factor approximation guarantee. Our alternative (tight) analysis of this algorithm is then given in Sect. 3.3.

### 3.2 Adaptation to Weighted Matroid Intersection

When adapting the local ratio algorithm for weighted matching to matroid intersection to obtain Algorithm 1, the first problem we encounter is the fact that matroids do not have a notion of vertices, so we cannot keep a weight/potential for each vertex. To describe how we overcome this issue, it is helpful to consider

the case of bipartite matching and in particular the example depicted in Fig. 1. It is well known that the weighted matching problem on a bipartite graph with edge set  $E$  and bipartition  $V_1, V_2$  can be modelled as a weighted matroid intersection problem on matroids  $M_1 = (E, I_1)$  and  $M_2 = (E, I_2)$  where for  $i \in \{1, 2\}$

$$I_i = \{E' \subseteq E \mid \text{each vertex } v \in V_i \text{ is incident to at most one vertex in } E'\}.$$

Instead of keeping a weight for each vertex, we will maintain two weight functions  $w_1$  and  $w_2$ , one for each matroid. These weight functions will be set so that the following holds in the special case of bipartite matching: on the arrival of a new edge  $e$ , let  $T_i \subseteq S$  be an independent set in  $I_i$  of selected edges that maximizes the weight function  $w_i$ . Then we have that

$$\min_{f \in T_i: T_i \setminus \{f\} \cup \{e\} \in I_i} w_i(f) \quad \text{if } T_i \cup \{e\} \notin I_i \text{ and } 0 \text{ otherwise} \tag{1}$$

equals the vertex potential of the incident vertex  $V_i$  when running the local ratio algorithm for weighted matching. It is well-known (e.g. by the optimality of the greedy algorithm for matroids) that the cheapest element  $f$  to remove from  $T_i$  to make  $T_i \setminus \{f\} \cup \{e\}$  an independent set equals the largest weight  $\theta$  so that the elements of weight at least  $\theta$  spans  $e$ . We thus have that (1) equals

$$\max (\{0\} \cup \{\theta : e \in \text{span}_{M_i} (\{f \in S \mid w_i(f) \geq \theta\})\})$$

and it follows that the quantities  $w_1^*(e)$  and  $w_2^*(e)$  in Algorithm 1 equal the incident vertex potentials in  $V_1$  and  $V_2$  of the local ratio algorithm in the special case of bipartite matching. To see this, let us return to our example in Fig. 1 and let  $V_1$  be the two vertices on the left and  $V_2$  be the two vertices on the right. In the bottom part of the figure, the weight functions  $w_1$  and  $w_2$  are depicted (at the corresponding side of the edge) after the arrival of each edge. At time step 1,  $e_1$  does not need to replace any elements in any of the matroids and so  $w_1^*(e_1) = w_2^*(e_1) = 0$ . We therefore have that its gain is  $g(e_1) = 1$  and the algorithm sets  $w_1(e_1) = w_2(e_1) = 1$ . At time 2, edge  $e_2$  of weight 2 arrives. It is not spanned in the first matroid whereas it is spanned by edge  $e_1$  of weight 1 in the second matroid. It follows that  $w_1^*(e_2) = 0$  and  $w_2^*(e_2) = w_2(e_1) = 1$  and so  $e_2$  has positive gain  $g(e_2) = 1$  and it sets  $w_1(e_2) = 1$  and  $w_2(e_2) = w_2(e_1) + 1 = 2$ . The third time step is similar to the second. At the last time step,  $e_4$  of weight 2 arrives. However, since it is spanned by  $e_1$  with  $w_1(e_1) = 1$  in the first matroid and by  $e_3$  with  $w_2(e_3) = 1$  in the second matroid, its gain is 0 and it is thus not added to the set  $S$ . Note that throughout this example, and in general for bipartite graphs, Algorithm 1 is identical to algorithm for weighted matching. One may therefore expect that the analysis of the latter also generalizes to Algorithm 1. We explain next that this is not the case for general matroids.

**Counter Example to Same Approach in Analysis.** We give a simple example showing that the greedy selection (as done in the analysis for local ratio algorithm for weighted matching) does not work for matroid intersection. Still, it turns out that the set  $S$  generated by Algorithm 1 always contains a 2-approximation but the selection process is more involved.



**Lemma 1.** *There exist two matroids  $M_1 = (E, I_1)$  and  $M_2 = (E, I_2)$  on a common ground set  $E$  and a weight function  $w : E \rightarrow \mathbb{R}_{\geq 0}$  such that a greedy algorithm that considers the elements in the set  $S$  in the reverse order of when they were added by Algorithm 1 does not provide any constant-factor approximation.*

*Proof.* The example consists of the ground set  $E = \{a, b, c, d\}$  with weights  $w(a) = 1, w(b) = 1 + \varepsilon, w(c) = 2\varepsilon, w(d) = 3\varepsilon$  for a small  $\varepsilon > 0$  (the approximation guarantee will be at least  $\Omega(1/\varepsilon)$ ). The matroids  $M_1 = (E, I_1)$  and  $M_2 = (E, I_2)$  are defined by

- a subset of  $E$  is in  $I_1$  if and only if it does not contain  $\{a, b\}$ ; and
- a subset of  $E$  is in  $I_2$  if and only if it contains at most two elements.

To see that  $M_1$  and  $M_2$  are matroids, note that  $M_1$  is a partition matroid with partitions  $\{a, b\}, \{c\}, \{d\}$ , and  $M_2$  is the 2-uniform matroid (alternatively, one can easily check that  $M_1$  and  $M_2$  satisfy the definition of a matroid). Now consider the execution of Algorithm 1 when given the elements of  $E$  in the order  $a, b, c, d$ :

- Element  $a$  has weight 1, and  $\{a\}$  is independent both in  $M_1$  and  $M_2$ , so we set  $w_1(a) = w_2(a) = g(a) = 1$  and  $a$  is added to  $S$ .
- Element  $b$  is spanned by  $a$  in  $M_1$  and not spanned by any element in  $M_2$ . So we get  $g(b) = w(b) - w_1^*(b) - w_2^*(b) = 1 + \varepsilon - 1 - 0 = \varepsilon$ . As  $\varepsilon > 0$ , we add  $b$  to  $S$ , and set  $w_1(b) = w_1(a) + \varepsilon = 1 + \varepsilon$  and  $w_2(b) = \varepsilon$ .
- Element  $c$  is not spanned by any element in  $M_1$  but is spanned by  $\{a, b\}$  in  $M_2$ . As  $b$  has the smallest  $w_2$  weight,  $w_2^*(c) = w_2(b) = \varepsilon$ . So we have  $g(c) = 2\varepsilon - w_1^*(c) - w_2^*(c) = 2\varepsilon - 0 - \varepsilon = \varepsilon > 0$ , and we set  $w_1(c) = \varepsilon$  and  $w_2(c) = 2\varepsilon$  and add  $c$  to  $S$ .
- Element  $d$  is similar to  $c$ . We have  $g(d) = 3\varepsilon - 0 - 2\varepsilon = \varepsilon > 0$  and so we set  $w_1(d) = \varepsilon$  and  $w_2(d) = 3\varepsilon$  and add  $d$  to  $S$ .

As the algorithm selected all the elements, we have  $S = E$ . It follows that the greedy algorithm on  $S$  (in the reverse order of when elements were added) will select  $d$  and  $c$ , after which the set is a maximal independent set in  $M_2$ . This gives a weight of  $5\varepsilon$ , even though  $a$  and  $b$  both have weight at least 1, which shows that this algorithm does not guarantee any constant factor approximation.  $\square$

### 3.3 Analysis of Algorithm 1

We prove that Algorithm 1 has an approximation guarantee of 2.

**Theorem 1.** *Let  $S$  be the subset generated by Algorithm 1 on a stream  $E$  of elements, matroids  $M_1 = (E, I_1), M_2 = (E, I_2)$  and weight function  $w : E \rightarrow \mathbb{R}_{\geq 0}$ . Then there exists a subset  $T \subseteq S$  independent in  $M_1$  and in  $M_2$  whose weight  $w(T)$  is at least  $w(S^*)/2$ , where  $S^*$  denotes an optimal solution to the weighted matroid intersection problem.*

Throughout the analysis we fix the input matroids  $M_1 = (E, I_1), M_2 = (E, I_2)$ , the weight function  $w : R \rightarrow \mathbb{R}_{\geq 0}$ , and the order of the elements in the stream. While Algorithm 1 only defines the weight functions  $w_1$  and  $w_2$  for the elements added to the set  $S$ , we extend them in the analysis by, for  $i \in \{1, 2\}$ , letting  $w_i(e) = w_i^*(e)$  for the elements  $e$  not added to  $S$ .

We now prove Theorem 1 by showing that  $g(S) \geq w(S^*)/2$  (Lemma 3) and that there is a solution  $T \subseteq S$  such that  $w(T) \geq g(S)$  (Lemma 4). In the proof of both these lemmas, we use the following properties of the computed set  $S$ .

**Lemma 2.** *Let  $S$  be the set generated by Algorithm 1 and  $S' \subseteq S$  any subset. Consider one of the matroids  $M_i$  with  $i \in \{1, 2\}$ . There exists a subset  $T' \subseteq S'$  that is independent in  $M_i$ , i.e.,  $T' \in I_i$ , and  $w_i(T') \geq g(S')$ . Furthermore, the maximum weight independent set in  $M_i$  over the whole ground set  $E$  can be selected to be a subset of  $S$ , i.e.  $T_i \subseteq S$ , and it satisfies  $w_i(T_i) = g(S)$ .*

*Proof.* Consider matroid  $M_1$  (the proof is identical for  $M_2$ ) and fix  $S' \subseteq S$ . The set  $T'_1 \subseteq S'$  that is independent in  $M_1$  and that maximizes  $w_1(T'_1)$  satisfies

$$w_1(T'_1) = \int_0^\infty \text{rank}(\{e \in T'_1 \mid w_1(e) \geq \theta\}) d\theta = \int_0^\infty \text{rank}(\{e \in S' \mid w_1(e) \geq \theta\}) d\theta.$$

The second equality follows from the fact that the greedy algorithm that considers the elements in decreasing order of weight is optimal for matroids and thus we have  $\text{rank}(\{e \in T'_1 \mid w_1(e) \geq \theta\}) = \text{rank}(\{e \in S' \mid w_1(e) \geq \theta\})$  for any  $\theta \in \mathbb{R}$ .

Now index the elements of  $S' = \{e_1, e_2, \dots, e_\ell\}$  in the order they were added to  $S$  by Algorithm 1 and let  $S'_j = \{e_1, \dots, e_j\}$  for  $j = 0, 1, \dots, \ell$  (where  $S'_0 = \emptyset$ ). By the above equalities and by telescoping,

$$w_1(T'_1) = \sum_{i=1}^{\ell} \int_0^\infty (\text{rank}(\{e \in S'_i \mid w_1(e) \geq \theta\}) - \text{rank}(\{e \in S'_{i-1} \mid w_1(e) \geq \theta\})) d\theta.$$

We have that  $\text{rank}(\{e \in S'_i \mid w_1(e) \geq \theta\}) - \text{rank}(\{e \in S'_{i-1} \mid w_1(e) \geq \theta\})$  equals 1 if  $w(e_i) \geq \theta$  and  $e_i \notin \text{span}(\{e \in S'_{i-1} \mid w_1(e) \geq \theta\})$  and it equals 0 otherwise. Therefore, by the definition of  $w_1^*(\cdot)$ , the gain  $g(\cdot)$  and  $w_1(e_i) = w_1^*(e_i) + g(e_i)$  in Algorithm 1 we have

$$\begin{aligned} w_1(T'_1) &= \sum_{i=1}^{\ell} [w_1(e_i) - \max(\{0\} \cup \{\theta : e_i \in \text{span}(\{f \in S'_{i-1} \mid w_i(f) \geq \theta\})\})] \\ &\geq \sum_{i=1}^{\ell} g(e_i) = g(S'). \end{aligned}$$

The inequality holds because  $S'_{i-1}$  is a subset of the set  $S$  at the time when Algorithm 1 considers element  $e_i$ . Moreover, if  $S' = S$ , then  $S'_{i-1}$  equals the set  $S$  at that point and so we then have

$$w_1^*(e_i) = \max(\{0\} \cup \{\theta : e_i \in \text{span}(\{f \in S'_{i-1} \mid w_i(f) \geq \theta\})\}).$$

This implies that the above inequality holds with equality in that case. We can thus also conclude that a maximum weight independent set  $T_1 \subseteq S$  satisfies

$w_1(T_1) = g(S)$ . Finally, we can observe that  $T_1$  is also a maximum weight independent set over the whole ground set since we have  $\text{rank}(\{e \in S \mid w_1(e) \geq \theta\}) = \text{rank}(\{e \in E \mid w_1(e) \geq \theta\})$  for every  $\theta > 0$ , which holds because, by the extension of  $w_1$ , an element  $e \notin S$  satisfies  $e \in \text{span}(\{f \in S : w_1(f) \geq w_1(e)\})$ .  $\square$

We can now relate the gain of the elements in  $S$  with the weight of an optimal solution.

**Lemma 3.** *Let  $S$  be the subset generated by Algorithm 1. Then  $g(S) \geq w(S^*)/2$ .*

*Proof.* We first observe that  $w_1(e) + w_2(e) \geq w(e)$  for every element  $e \in E$ . Indeed, for an element  $e \in S$ , we have by definition  $w(e) = g(e) + w_1^*(e) + w_2^*(e)$ , and  $w_i(e) = g(e) + w_i^*(e)$ , so  $w_1(e) + w_2(e) = 2g(e) + w_1^*(e) + w_2^*(e) = w(e) + g(e) > w(e)$ . In the other case, when  $e \notin S$  then  $w_1^*(e) + w_2^*(e) \geq w(e)$ , and  $w_i(e) = w_i^*(e)$ , so automatically,  $w_1(e) + w_2(e) \geq w(e)$ .

The above implies that  $w_1(S^*) + w_2(S^*) \geq w(S^*)$ . On the other hand, by Lemma 2, we have  $w_i(T_i) \geq w_i(S^*)$  (since  $T_i$  is a max weight independent set in  $M_i$  with respect to  $w_i$ ) and  $w_i(T_i) = g(S)$ , thus  $g(S) \geq w_i(S^*)$  for  $i = 1, 2$ .  $\square$

We finish the proof of Theorem 1 by proving that there is a  $T \subseteq S$  independent in both  $M_1$  and  $M_2$  such that  $w(T) \geq g(S)$ . As described in Sect. 3.2, we cannot select  $T$  using the greedy method. Instead, we select  $T$  using the concept of kernels studied in [7].

**Lemma 4.** *Let  $S$  be the subset generated by Algorithm 1. Then there exists a subset  $T \subseteq S$  independent in  $M_1$  and in  $M_2$  such that  $w(T) \geq g(S)$ .*

*Proof.* Consider one of the matroids  $M_i$  with  $i \in \{1, 2\}$  and define a total order  $<_i$  on  $E$  such that  $e <_i f$  if  $w_i(e) > w_i(f)$  or if  $w_i(e) = w_i(f)$  and  $e$  appeared later in the stream than  $f$ . The pair  $(M_i, <_i)$  is known as an ordered matroid. We further say that a subset  $E'$  of  $E$  dominates element  $e$  of  $E$  if  $e \in E'$  or there is a subset  $C_e \subseteq E'$  such that  $e \in \text{span}(C_e)$  and  $c < e$  for all elements  $c$  of  $C_e$ . The set of elements dominated by  $E'$  is denoted by  $D_{M_i}(E')$ . Note that if  $E'$  is an independent set, then the greedy algorithm that considers the elements of  $D_{M_i}(E')$  in the order  $<_i$  selects exactly the elements  $E'$ .

Theorem 2 in [7] says that for two ordered matroids  $(M_1, <_1), (M_2, <_2)$  there always is a set  $K \subseteq E$ , which is referred to as a  $M_1 M_2$ -kernel, such that

- $K$  is independent in both  $M_1$  and in  $M_2$ ; and
- $D_{M_1}(K) \cup D_{M_2}(K) = E$ .

We use the above result on  $M_1$  and  $M_2$  restricted to the elements in  $S$ . Specifically we select  $T \subseteq S$  to be the kernel such that  $D_{M_1}(T) \cup D_{M_2}(T) = S$ . Let  $S_1 = D_{M_1}(T)$  and  $S_2 = D_{M_2}(T)$ . By Lemma 2, there exists a set  $T' \subseteq S_1$  independent in  $M_1$  such that  $w_1(T') \geq g(S_1)$ . As noted above, the greedy algorithm that considers the element of  $S_1$  in the order  $<_i$  (decreasing weights) selects exactly the elements in  $T$ . It follows by the optimality of the greedy

algorithm for matroids that  $T$  is optimal for  $S_1$  in  $M_1$  with weight function  $w_1$ , which in turn implies  $w_1(T) \geq g(S_1)$ . In the same way, we also have  $w_2(T) \geq g(S_2)$ . By definition, for any  $e \in S$ , we have  $w(e) = w_1(e) + w_2(e) - g(e)$ . Together, we have  $w(T) = w_1(T) + w_2(T) - g(T) \geq g(S_1) + g(S_2) - g(T)$ . As elements from  $T$  are in both  $S_1$  and  $S_2$ , and all other elements are in at least one of both sets, we have  $g(S_1) + g(S_2) \geq g(S) + g(T)$ , and thus  $w(T) \geq g(S)$ .  $\square$

## 4 Making the Algorithm Memory Efficient

We now modify Algorithm 1 to only select elements with a significant gain, parametrized by  $\alpha > 1$ , and delete elements if we have too many in memory, parametrized by a real number  $y$ . Let us call this algorithm **EMI**. If  $\alpha$  is close enough to 1 and  $y$  is large enough, then EMI is very close to Algorithm 1, and allows for a similar analysis. This method is very similar to the one used in [14] and [9], but our analysis is quite different.

More precisely, we take an element  $e$  only if  $w(e) > \alpha(w_1^*(e) + w_2^*(e))$  instead of  $w(e) > w_1^*(e) + w_2^*(e)$ , and we delete all elements  $e'$  in the current stack  $S$  for which the ratio between the  $g$  weight and the maximum  $g$  weight i.e.,  $g_{max} = \max_{e \in S} g(e)$  exceeds  $y$  (i.e.,  $\frac{g_{max}}{g(e')} > y$ ). For technical purposes, we also need to keep independent sets  $T_1$  and  $T_2$  which maximize the weight functions  $w_1$  and  $w_2$  respectively. If an element with small  $g$  weight is in  $T_1$  or  $T_2$ , we do not delete it, as this would modify the  $w_i$ -weights and selection of coming elements. We show that this algorithm is a semi-streaming algorithm with an approximation guarantee of  $(2 + \varepsilon)$  for an appropriate selection of the parameters (see Lemma 6 for the space requirement and theorem2 for the approximation guarantee).

**Lemma 5.** *Let  $S$  be the subset generated by EMI with  $\alpha \geq 1$  and  $y = \infty$ . Then  $w(S^*) \leq 2\alpha g(S)$ .*

*Proof.* We define  $w_\alpha : E \rightarrow \mathbb{R}$  by  $w_\alpha(e) = w(e)$  if  $e \in S$  and  $w_\alpha(e) = \frac{w(e)}{\alpha}$  otherwise. By construction, EMI and Algorithm 1 give the same set  $S$ , and the same weight function  $g$  for this modified weight function. By Lemma 3,  $w_\alpha(S^*) \leq 2g(S)$ . On the other hand,  $w(S^*) \leq \alpha w_\alpha(S^*)$ .  $\square$

**Lemma 6.** *Let  $S$  be the subset generated generated by EMI with  $\alpha = 1 + \varepsilon$  and  $y = \frac{\min(r_1, r_2)}{\varepsilon^2}$  and  $S^*$  be a maximum weight independent set, where  $r_1$  and  $r_2$  are the ranks of  $M_1$  and  $M_2$  respectively. Then  $w(S^*) \leq 2(1 + 2\varepsilon + o(\varepsilon))g(S)$ . Furthermore, at any point of time, the size of  $S$  is at most  $r_1 + r_2 + \min(r_1, r_2) \log_\alpha(\frac{y}{\varepsilon})$ .*

*Proof.* We first prove that the generated set  $S$  satisfies  $w(S^*) \leq 2(1 + 2\varepsilon + o(\varepsilon))g(S)$  and we then verify the space requirement of the algorithm, i.e., that it is a semi-streaming algorithm.

Let us call  $S'$  the set of elements selected by EMI, including the elements deleted later. By Lemma 5, we have  $2\alpha g(S') \geq w(S^*)$ , so all we have to prove

is that  $g(S') - g(S) \leq \varepsilon g(S)$ . We set  $i \in \{1, 2\}$  to be the index of the matroid with smaller rank.

In our analysis, it will be convenient to think that the algorithm maintains the maximum weight independent set  $T_i$  of  $M_i$  throughout the stream. We have, at the arrival of an element  $e$  that is added to  $S$ , that the set  $T_i$  is updated as follows. If  $T_i \cup \{e\} \in I_i$  then  $e$  is simply added to  $T_i$ . Otherwise, before updating  $T_i$ , there is an element  $e^* \in T_i$  such that  $w_i(e^*) = w_i^*(e)$  and  $T_i \setminus \{e^*\} \cup \{e\}$  is maximum weight independent set in  $M_i$  with respect to  $w_i$ . Thus we can speak of elements which are *replaced* be another element in  $T_i$ . By construction, if  $e$  replaces  $f$  in  $T_i$ , then  $w_i(e) > \alpha w_i(f)$ .

We can now divide the elements of  $S'$  into stacks in the following way: If  $e$  replaces an element  $f$  in  $T_i$ , then we add  $e$  on top of the stack containing  $f$ , otherwise we create a new stack containing only  $e$ . At the end of the stream, each element  $e \in T_i$  is in a different stack, and each stack contains exactly one element of  $T_i$ , so let us call  $S'_e$  the stack containing  $e$  whenever  $e \in T_i$ . We define  $S_e$  to be the restriction of  $S'_e$  to  $S$ . In particular, each element from  $S'$  is in exactly one  $S'_e$  stack, and each element from  $S$  is in exactly one  $S_e$  stack. For each stack  $S'_e$ , we set  $e_{del}(S'_e)$  to be the highest element of  $S'_e$  which was removed from  $S$ . By construction,  $g(S'_e) - g(S_e) \leq w_i(e_{del}(S'_e))$ . On the other hand,  $w_i(f) < \frac{1}{\varepsilon}g(f)$  for any element  $f \in S'$  (otherwise we would not have selected it), so  $g(S'_e) - g(S_e) < \frac{1}{\varepsilon}g(e_{del}(S'_e))$ . As  $e_{del}(S'_e)$  was removed from  $S$ , we have  $g(e_{del}(S'_e)) < \frac{g_{max}}{y}$ . As there are exactly  $r_i$  stacks, we get  $g(S') - g(S) < r_i \frac{g_{max}\varepsilon^2}{r_i\varepsilon} = \varepsilon g_{max} \leq \varepsilon g(S)$ .

We now have to prove that the algorithm fits the semi-streaming criteria. In fact, the size of  $S$  never exceeds  $r_1 + r_2 + r_i \log_\alpha(\frac{y}{\varepsilon})$ . By the pigeonhole principle, if  $S$  has at least  $r_i \log_\alpha(\frac{y}{\varepsilon})$  elements, then there is at least one stack  $S_e$  which has at least  $\log_\alpha(\frac{y}{\varepsilon})$  elements. By construction, the  $w_i$  weight increases by a factor of at least  $\alpha$  each time we add an element on the same stack, so the  $w_i$  weight difference between the lowest and highest element on the biggest stack would be at least  $\frac{y}{\varepsilon}$ . As  $w_i(f) < \frac{1}{\varepsilon}g(f)$ , the  $g$  weight difference would be at least  $y$ , and we would remove the lowest element, unless it was in  $T_1$  or  $T_2$ .  $\square$

**Theorem 2.** *Let  $S$  be the subset generated by running EMI with  $\alpha = 1 + \varepsilon$  and  $y = \frac{\min(r_1, r_2)}{\varepsilon^2}$ . Then there exists a subset  $T \subseteq S$  independent in  $M_1$  and in  $M_2$  such that  $w(T) \geq g(S)$ . Furthermore,  $T$  is a  $2(1 + 2\varepsilon + o(\varepsilon))$ -approximation for the intersection of two matroids.*

*Proof.* Let  $S^*$  be a maximum weight independent set. By Lemma 6, we have  $2(1 + 2\varepsilon + o(\varepsilon))g(S) \geq w(S^*)$ . By Lemma 4 we can select an independent set  $T$  with  $w(T) \geq g(S)$  if the algorithm does not delete elements. Let  $S'$  be the set of elements selected by EMI, including the elements deleted later. As long as we do not delete elements from  $T_1$  or  $T_2$ , Algorithm 1 restricted to  $S'$  will select the same elements, with the same weights, so we can consider  $S'$  to be generated by Algorithm 1. We now observe that all the arguments used in Lemma 4 also work for a subset of  $S'$ , in particular, it is also true for  $S$  that we can find an independent set  $T \subseteq S$  such that  $w(T) \geq g(S)$ .  $\square$

*Remark 1.* EMI is not the most efficient possible in terms of memory, but is aimed to be simpler instead. Using the notion of stacks introduced in the proof of Lemma 6, it is possible to modify the algorithm and reduce the memory requirement by a factor  $\log(\min(\text{rank}(M_1), \text{rank}(M_2)))$ .

*Remark 2.* The techniques of this section can also be used in the case when the ranks of the matroids are unknown. Specifically, the algorithm can maintain the stacks created in the proof of Lemma 6 and allow for an error  $\varepsilon$  in the first two stacks created, an error of  $\varepsilon/2$  in the next 4 stacks, and in general an error of  $\varepsilon/2^i$  in the next  $2^i$  stacks.

*Remark 3.* It is easy to construct examples where the set  $S$  only contains a  $2\alpha$ -approximation (for example with bipartite graphs), so up to a factor  $\varepsilon$  our analysis is tight.

## 5 More Than Two Matroids

We can easily extend the algorithm EMI in the previous section to the intersection of  $k$  matroids. Now for any element  $e$ , if  $w(e) > \alpha \sum_{i=1}^k w_i^*(e)$ , we add it to our stack and update the weight functions  $w_1, \dots, w_k$  similarly as EMI. The only part which does not work is the selection of the independent set from  $S$ . Indeed, matroid kernels are very specific to two matroids. We refer the reader to the full version [8] to see why a similar approach fails and that a counter-example can arise. Thus, any attempt to find a  $k$  approximation using our techniques must bring some fundamentally new idea. Still, we conjecture that the generated set  $S$  contains such an approximation.

**Acknowledgements.** The authors thank Moran Feldman for pointing us to the recent paper [11].

## References

1. Bar-Yehuda, R., Even, S.: A local-ratio theorem for approximating the weighted vertex cover problem. In: Ausiello, G., Lucertini, M. (eds.) Analysis and Design of Algorithms for Combinatorial Problems, North-Holland Mathematics Studies, vol. 109, pp. 27–45. North-Holland (1985). [https://doi.org/10.1016/S0304-0208\(08\)73101-3](https://doi.org/10.1016/S0304-0208(08)73101-3), <http://www.sciencedirect.com/science/article/pii/S0304020808731013>
2. Bar-Yehuda, R., Bendel, K., Freund, A., Rawitz, D.: Local ratio: A unified framework for approximation algorithms. in memoriam: Shimon even 1935–2004. ACM Comput. Surv. (CSUR) **36**, 422–463 (2004). <https://doi.org/10.1145/1041680.1041683>
3. Chakrabarti, A., Kale, S.: Submodular maximization meets streaming: Matchings, matroids, and more. CoRR abs/1309.2038 (2013). <http://arxiv.org/abs/1309.2038>
4. Crouch, M., Stubbs, D.: Improved streaming algorithms for weighted matching, via unweighted matching. In: Leibniz International Proceedings in Informatics, LIPIcs, vol. 28, pp. 96–104 (2014). <https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2014.96>

5. Edmonds, J.: Matroid intersection. In: Discrete Optimization I, Annals of Discrete Mathematics, vol. 4, pp. 39–49. Elsevier (1979)
6. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. *Theor. Comput. Sci.* **348**(2–3), 207–216 (2005)
7. Fleiner, T.: A matroid generalization of the stable matching polytope. In: Aardal, K., Gerards, B. (eds.) IPCO 2001. LNCS, vol. 2081, pp. 105–114. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-45535-3\\_9](https://doi.org/10.1007/3-540-45535-3_9)
8. Garg, P., Jordan, L., Svensson, O.: Semi-streaming algorithms for submodular matroid intersection. arXiv preprint [arXiv:2102.04348](https://arxiv.org/abs/2102.04348) (2021)
9. Ghaffari, M., Wajc, D.: Simplified and Space-Optimal Semi-Streaming  $(2+\epsilon)$ -Approximate Matching. In: Fineman, J.T., Mitzenmacher, M. (eds.) 2nd Symposium on Simplicity in Algorithms (SOSA 2019). OpenAccess Series in Informatics (OASICS), vol. 69, pp. 13:1–13:8. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018). <https://doi.org/10.4230/OASICS.SOSA.2019.13>, <http://drops.dagstuhl.de/opus/volltexte/2018/10039>
10. Huang, C., Kakimura, N., Mauras, S., Yoshida, Y.: Approximability of monotone submodular function maximization under cardinality and matroid constraints in the streaming model. CoRR abs/2002.05477 (2020)
11. Levin, R., Wajc, D.: Streaming submodular matching meets the primal-dual method. arXiv preprint [arXiv:2008.10062](https://arxiv.org/abs/2008.10062) (2020)
12. McGregor, A.: Finding graph matchings in data streams. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX/RANDOM -2005. LNCS, vol. 3624, pp. 170–181. Springer, Heidelberg (2005). [https://doi.org/10.1007/11538462\\_15](https://doi.org/10.1007/11538462_15)
13. Muthukrishnan, S.: Data streams: algorithms and applications. *Found. Trends Theor. Comput. Sci.* **1**(2), 117–236 (2005). <https://doi.org/10.1561/0400000002>
14. Paz, A., Schwartzman, G.: A  $(2+\epsilon)$ -approximation for maximum weight matching in the semi-streaming model. CoRR abs/1702.04536 (2017), <http://arxiv.org/abs/1702.04536>
15. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Algorithms and Combinatorics (2003)