

Semi-Streaming Algorithms for Submodular Function Maximization Under b -Matching, Matroid, and Matchoid Constraints

Chien-Chung Huang ✉

CNRS, DI ENS, École normale supérieure, Université PSL, France

François Sellier ✉

École polytechnique, Institut Polytechnique de Paris, France

Abstract

We consider the problem of maximizing a non-negative submodular function under the b -matching constraint, in the semi-streaming model. When the function is linear, monotone, and non-monotone, we obtain the approximation ratios of $2 + \varepsilon$, $3 + 2\sqrt{2} \approx 5.828$, and $4 + 2\sqrt{3} \approx 7.464$, respectively.

We also consider a generalized problem, where a k -uniform hypergraph is given, along with an extra matroid or a k' -matchoid constraint imposed on the edges, with the same goal of finding a b -matching that maximizes a submodular function. When the extra constraint is a matroid, we obtain the approximation ratios of $k + 1 + \varepsilon$, $k + 2\sqrt{k+1} + 2$, and $k + 2\sqrt{k+2} + 3$ for linear, monotone and non-monotone submodular functions, respectively. When the extra constraint is a k' -matchoid, we attain the approximation ratio $\frac{8}{3}k + \frac{64}{9}k' + O(1)$ for general submodular functions.

2012 ACM Subject Classification Theory of computation \rightarrow Streaming, sublinear and near linear time algorithms; Theory of computation \rightarrow Approximation algorithms analysis

Keywords and phrases Maximum weight matching, submodular function maximization, streaming, matroid, matchoid

Related Version A preliminary version of this article appears in APPROX 2021 [11].

1 Introduction

Let $G = (V, E)$ be a multi-graph (without self-loops) where each vertex $v \in V$ is associated with a *capacity* $b_v \in \mathbb{Z}_+$. A b -matching is a subset of edges $M \subseteq E$ where each vertex v has at most b_v incident edges contained in M .

In the maximum weight b -matching problem, edges are given weights $w : E \rightarrow \mathbb{R}_+$ and we need to compute a b -matching M so that $w(M) = \sum_{e \in M} w(e)$ is maximized. A generalization of the problem is that of maximizing a non-negative *submodular function* $f : 2^E \rightarrow \mathbb{R}_+$ under the b -matching constraint, namely, we look for a b -matching M so that $f(M)$ is maximized. Here we recall the definition of a submodular function:

$$\forall X \subseteq Y \subseteq E, \forall e \in E \setminus Y, f(X \cup \{e\}) - f(X) \geq f(Y \cup \{e\}) - f(Y).$$

Additionally f is called *monotone* if $\forall X \subseteq Y \subseteq E, f(X) \leq f(Y)$, otherwise it is *non-monotone*. Observe that the maximum weight matching is the special case where f is a linear sum of the weights associated with the edges. Note that throughout this paper, we implicitly assume that the submodular function f is always non-negative.

In the traditional offline setting, both problems are extensively studied. The maximum weight b -matching can be solved in polynomial time [16]; maximizing a non-negative monotone submodular function under b -matching constraint is NP-hard and the best approximation ratios so far are $2 + \varepsilon$ and $4 + \varepsilon$, for the monotone and non-monotone case, respectively [8].

In this work, we consider the problem in the semi-streaming model [14]. Here the edges in E arrive over time but we have only limited space (ideally proportional to the output size)

and cannot afford to store all edges in E —this rules out the possibility of applying known offline algorithms.

We also consider a generalized problem, where a matroid or a matchoid is imposed on the edges of an hypergraph. Specifically, here $G = (V, E)$ is a k -uniform hypergraph, where each edge $e \in E$ contains k vertices in V . In addition to the capacities b_v , a matroid or a matchoid \mathcal{M} (see the next section for the definitions) is imposed on the edges of E . A b -matching M is feasible only if M is independent in \mathcal{M} . The objective here is to find a feasible b -matching M that maximizes $f(M)$ for f a submodular function.

1.1 Our Contribution

We start with the maximum weight (b -)matching. For this problem, a long series of papers [4, 5, 6, 10, 12, 13, 15, 17] have proposed many semi-streaming algorithms, with progressively improved approximation ratios, culminating in the work of Paz and Schwartzman [15], where $2 + \varepsilon$ approximation is attained, for the simple matching. For the general b -matching, very recently, Levin and Wajc [12] gave a $3 + \varepsilon$ approximation algorithm. We close the gap between the simple matching and the general b -matching.

► **Theorem 1.** *For the maximum weight b -matching problem, we obtain a $2 + \varepsilon$ approximation algorithm using $O(\log_{1+\varepsilon}(W/\varepsilon) \cdot |M_{max}|)$ variables in memory, and another using $O(\log_{1+\varepsilon}(1/\varepsilon) \cdot |M_{max}| + \sum_{v \in V} b_v)$ variables, where M_{max} denotes the maximum cardinality b -matching and W denotes the maximum ratio between two non-zero weights.*

The details of this theorem can be found in Section 2.

Next we consider the general case of submodular functions, for whom approximation algorithms have been proposed in [2, 3, 12, 7]. The current best ratios obtained by Levin and Wajc [12] are $3 + 2\sqrt{2} \approx 5.828$ and $4 + 2\sqrt{3} \approx 7.464$, for the monotone and non-monotone functions, respectively. We propose an alternative algorithm to achieve the same bounds.

► **Theorem 2.** *To maximize a non-negative submodular function under the b -matching constraint, we obtain algorithms providing a $3 + 2\sqrt{2} \approx 5.828$ approximation for monotone functions and a $4 + 2\sqrt{3} \approx 7.464$ approximation for non-monotone functions, using $O(\log W \cdot |M_{max}|)$ variables, where M_{max} denotes the maximum cardinality b -matching and W denotes the maximum quotient $\frac{f(e|Y)}{f(e'|X)}$, for $X \subseteq Y \subseteq E$, $e, e' \in E$, $f(e'|X) > 0$.*

The details of this theorem can be found in Section 3.

It should be pointed out that in [12], for the case of non-monotone functions, their algorithm only works for the simple matching, and it is unclear how to generalize it to the general b -matching, while our algorithm lifts this restriction. Another interesting thing to observe is that even though the achieved ratios are the same and our analysis borrows ideas from [12], our algorithm is not really just the same algorithm disguised under a different form. See Appendix B for a concrete example where the two algorithms behave differently.

Extension: k -uniform hypergraph with a matroid or a matchoid constraint

Before we explain our results, let us recall some definitions.

A pair $\mathcal{M} = (E, \mathcal{I})$ 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 $e \in Y \setminus X$ so that $X \cup \{e\} \in \mathcal{I}$. The sets in \mathcal{I} are the *independent sets* and the *rank* $r_{\mathcal{M}}$ of the matroid \mathcal{M} is defined as $\max_{X \in \mathcal{I}} |X|$. A minimum (inclusion-wise) non-independent subset is called a *circuit*. For any subset $S \subseteq E$, its *rank* $r_{\mathcal{M}}(S)$ is defined as the maximum size of an

independent set in S . We will also use the notion of *span*, which is defined for a subset S of E as $\text{span}(S) = \{e \in E : r_{\mathcal{M}}(S) = r_{\mathcal{M}}(S \cup \{e\})\}$. These definitions are close to the ones for independent families in vector spaces, and their nomenclature is similar.

In the following, we will sometimes refer to particular types of matroids: uniform and partition matroids. A *uniform matroid* $\mathcal{M} = (E, \mathcal{I})$ of rank $r_{\mathcal{M}}$ is a matroid such that the independent sets are the sets of cardinality at most $r_{\mathcal{M}}$. A *partition matroid* $\mathcal{M} = (E, \mathcal{I})$ is defined by a partition E_1, \dots, E_l of E and bounds k_1, \dots, k_l : a set $M \subseteq E$ is independent in \mathcal{M} if for all $1 \leq i \leq l$, $|M \cap E_i| \leq k_i$.

A matchoid $\mathcal{M} = (\mathcal{M}_i = (E_i, \mathcal{I}_i))_{i=1}^s$ is a collection of matroids $\mathcal{M}_i = (E_i, \mathcal{I}_i)$. Specifically, a k' -matchoid is a matchoid with the additional condition that each element $e \in E = \cup_{i=1}^s E_i$ appears in at most k' of E_i s. A set $S \subseteq \cup_{i=1}^s E_i$ is independent in the matchoid \mathcal{M} only if $S \cap E_i$ is independent for all $1 \leq i \leq s$. For instance, a partition matroid can be seen as 1-matchoid made of uniform matroids.

In our generalization, a k -uniform hypergraph is given with an additional matroid or a k' -matchoid \mathcal{M} imposed. The goal is to find a b -matching M , which is also independent in \mathcal{M} , so as to maximize $f(M)$.

We notice that the problem of maximizing a submodular function under a p -matchoid constraint (without our additional b -matching constraint in the hypergraph) has already been studied by Chekuri et al. [3] and Feldman et al. [7], where the best approximation ratios for the maximizing a monotone and a non-monotone function are respectively $4p$ and $2p + 2\sqrt{p(p+1)} + 1$. It is easy to see that, if the constraint \mathcal{M} is a matroid, our problem is a special case of the $(k+1)$ -matchoid submodular function maximization, and if the constraint \mathcal{M} is a k' -matchoid, our problem is a special case of the $(k+k')$ -matchoid submodular function maximization¹.

Essentially in our next two theorems, we show that if the majority of matroids contained in the matchoid are simple uniform matroids, we can attain better approximation ratios than in [3, 7].

First consider the case where \mathcal{M} is a simple matroid.

► **Theorem 3.** *To maximize linear functions under the b -matching constraint with an additional matroid constraint, we obtain a $k + 1 + \varepsilon$ approximation algorithm, having a memory consumption of $O(\log_{1+\varepsilon}(W/\varepsilon) \cdot \min\{|M_{max}|, r_{\mathcal{M}}\})$ variables, where M_{max} denotes the maximum cardinality b -matching and W denotes the maximum ratio between two non-zero weights. To maximize non-negative submodular functions, we have an algorithm providing an approximation ratio of $k + 2\sqrt{k+1} + 1$ for monotone functions and $k + 2\sqrt{k+2} + 3$ for non-monotone functions, using $O(\log_{1+\varepsilon}(W/\varepsilon) \cdot \min\{|M_{max}|, r_{\mathcal{M}}\})$ variables in memory, where W denotes the maximum quotient $\frac{f(e|Y)}{f(e'|X)}$, for $X \subseteq Y \subseteq E$, $e, e' \in E$, such that $f(e'|X) > 0$.*

The details of this theorem can be found in Section 4.

Observe that when f is a monotone, or a non-monotone submodular function, our ratios are better than $4(k+1)$ and $2(k+1) + 2\sqrt{(k+1)(k+2)} + 1$ achieved in [3] and in [7].

Our result is of particular interest when f is a linear function. In fact, notice that if the given hypergraph is k -partite, our problem is the same as finding the maximum weight intersection of k partition matroids and an arbitrary matroid—this is a special case of finding a maximum weight intersection of $k+1$ arbitrary matroids, a problem recently studied by

¹ Indeed, one can observe that the capacity constraint associated with a vertex can be seen as a uniform matroid: each edge involves k such uniform matroids.

Garg et al. [9]. It can be seen that the set of stored elements in our algorithm is identical to that in the algorithm of [9]. Garg et al. have conjectured that there always exists a $k + 1$ approximation in their stored elements (and have proved this to be true when $k + 1 = 2$). Theorem 3 thus confirms their conjecture for the special case when all but one matroids are partition matroids.

We next consider the case where \mathcal{M} is a k' -matchoid.

► **Theorem 4.** *To maximize a non-negative submodular function under the b -matching constraint with an additional k' -matchoid constraint, we have an algorithm providing an approximation ratio better than $\frac{8}{3}k + \frac{64}{9}k' + O(1)$ for both monotone and non-monotone functions, using $O(\log W \cdot |M_{max}|)$ variables in memory, where M_{max} denotes the maximum cardinality b -matching and W denotes the maximum quotient $\frac{f(e|Y)}{f(e'|X)}$, for $X \subseteq Y \subseteq E$, $e, e' \in E$, such that $f(e'|X) > 0$.*

The details of this theorem can be found in Section 5.

The exact expressions for the approximation ratios are formally stated in Theorems 45 and 47. For a comparison, recall that using the techniques of [3, 7], the obtained ratio is $4(k + k')$ and $2(k + k') + 2\sqrt{(k + k' + 1)(k + k' + 2)} + 1$ when f is monotone and non-monotone, respectively. Our result shows that if k is at least roughly three times of k' , better approximation ratios can be achieved.

Comparison with the conference version of this work.

Compared with our previous results of [11], Theorems 3 and 4 are significant improvements. In fact, Theorem 3 allows us to get approximation ratios of $k + 1 + \varepsilon$, $k + 2\sqrt{k + 1} + 1$ and $k + 2\sqrt{k + 2} + 3$ for linear, monotone, and non-monotone submodular functions, respectively, meanwhile in [11] the approximation ratios are $k + 2\sqrt{k} + 1$, $\frac{8}{3}k + O(1)$, and $\frac{8}{3}k + O(1)$. Moreover, Theorem 4 is a generalization of the technique used for the matroids in [11] to matchoids.

1.2 Our Technique

We use a local-ratio technique to decide to retain or to discard a newly arrived edge during the streaming phase. After this phase, a greedy algorithm, according to the reverse edge arrival order, is then applied to add edges one by one into the solution while guaranteeing feasibility.

This is in fact the same framework used in [12, 15]. Our main technical innovation is to introduce a data structure, which takes the form of a set of queues. Such a set of queues is associated with each vertex (and with the imposed matroid). Every edge, if retained, will appear as an element² in one of these queues for each of its endpoints (and for the imposed matroid). These queues will guide the greedy algorithm to make better choices and are critical in our improvement over previous results. Here we give some intuition behind these queues. Consider the maximum weight b -matching problem. Similar to [9], we compute, for every edge, a *gain*. The sum of the gains of the retained edges can be shown to be at least half of the real weight of the unknown optimal matching. The question then boils down to how to “extract” a matching whose real weight is large compared to these gains of the retained edges. In our queues, the elements are stacked in such a way that the weight of an

² In this article, we will often use “edge” and “element” interchangeably.

element e is the sum of the gains of all the elements preceding e in any queue containing e . This suggests that if e is taken by the greedy algorithm, we can as well ignore all elements that are underneath e in the queues, as their gains are already “paid for” by e .

2 Maximum Weight b -Matching

2.1 Description of the Algorithm

For ease of description, we explain how to achieve 2 approximation, ignoring the issue of space complexity for the moment. We will explain how a slight modification can ensure the desired space complexity, at the expense of an extra ε term in the approximation ratio (see Appendix A).

The formal algorithm for the streaming phase is shown in Algorithm 1. We give an informal description here. Let S , initially empty, be the set of edges that have been stored so far. For each vertex $v \in V$, a set $Q_v = \{Q_{v,1}, \dots, Q_{v,b_v}\}$ of queues are maintained. These queues contain the edges incident to v that are stored in S and respect the arrival order of edges (newer edges are higher up in the queues). Each time a new edge e arrives, we compute its *gain* $g(e)$ (see Lines 5 and 8). Edge e is added into S only if its gain is strictly positive. If this is the case, for each endpoint u of e , we put e in one of u 's queues (see Lines 6 and 13) and define a *reduced weight* $w_u(e)$ (Line 11). It should be noted that $w_u(e)$ will be exactly the sum of the gains of the edges preceding (and including) e in the queue. We refer to the last element inserted in a queue Q as the top element of that queue, denoted $Q.top()$. To insert an element e on top of a queue Q , we use the instruction $Q.push(e)$. By convention, for an empty queue Q we have $Q.top() = \perp$. We also set $w_u(\perp) = 0$. Notice that each element e also has, for each endpoint $v \in e$, a pointer $r_v(e)$ to indicate its immediate predecessor in the queue of v , where it appears.

Algorithm 1 Streaming phase for weighted matching

```

1:  $S \leftarrow \emptyset$ 
2:  $\forall v \in V : Q_v \leftarrow (Q_{v,1} = \emptyset, \dots, Q_{v,b_v} = \emptyset)$   $\triangleright b_v$  queues for a vertex  $v$ 
3: for  $e = e_t, 1 \leq t \leq |E|$  an edge from the stream do
4:   for  $u \in e$  do
5:      $w_u^*(e) \leftarrow \min\{w_u(Q_{u,q}.top()) : 1 \leq q \leq b_u\}$ 
6:      $q_u(e) \leftarrow q$  such that  $w_u(Q_{u,q}.top()) = w_u^*(e)$ 
7:   if  $w(e) > \sum_{u \in e} w_u^*(e)$  then
8:      $g(e) \leftarrow w(e) - \sum_{u \in e} w_u^*(e)$ 
9:      $S \leftarrow S \cup \{e\}$ 
10:    for  $u \in e$  do
11:       $w_u(e) \leftarrow w_u^*(e) + g(e)$ 
12:       $r_u(e) \leftarrow Q_{u,q_u(e)}.top()$   $\triangleright r_u(e)$  is the element below  $e$  in the queue
13:       $Q_{u,q_u(e)}.push(e)$   $\triangleright$  add  $e$  on the top of the smallest queue

```

After the streaming phase, our greedy algorithm, formally described in Algorithm 2, constructs a b -matching based on the stored set S .

The greedy proceeds based on the reverse edge arrival order—but with an important modification. Once an edge e is taken as part of the b -matching, all edges preceding e that are stored in the same queue as e will be subsequently ignored by the greedy algorithm. The variables z_e are used to mark this fact.

Algorithm 2 Greedy construction phase

```

1:  $M \leftarrow \emptyset$ 
2:  $\forall e \in S : z_e \leftarrow 1$ 
3: for  $e \in S$  in reverse order do
4:   if  $z_e = 0$  then continue ▷ skip edge  $e$  if it is marked
5:    $M \leftarrow M \cup \{e\}$ 
6:   for  $u \in e$  do
7:      $c \leftarrow e$ 
8:     while  $c \neq \perp$  do
9:        $z_c \leftarrow 0$  ▷ mark elements below  $e$  in each queue
10:       $c \leftarrow r_u(c)$ 
11: return  $M$ 

```

2.2 Analysis for Maximum Weight b -Matching

For analysis, for each discarded element $e \in E \setminus S$, we set $g(e) = 0$ and $w_u(e) = w_u^*(e)$ for each $u \in e$. The *weight* of a queue, $w_u(Q_{u,i})$, is defined as the reduced weight of its top element, namely, $w_u(Q_{u,i}.top())$. Let $w_u(Q_u) = \sum_{i=1}^{b_u} w_u(Q_{u,i})$. We write $S^{(t)}$ as the value of S at the end of the iteration t of the streaming phase, and by convention $S^{(0)} = \emptyset$. This notation $^{(t)}$ will also be used for other sets such as Q_u and $Q_{u,i}$. Through this paper, M^{opt} will always refer to the best solution for the considered problem.

The following proposition follows easily by induction.

- **Proposition 5.** (i) For all $v \in V$ we have $g(\delta(v)) = g(\delta(v) \cap S) = w_v(Q_v)$.
(ii) The set $\{Q_{v,q}.top() : 1 \leq q \leq b_v\}$ contains the b_v heaviest elements of $S \cap \delta(v)$ in terms of reduced weights.

- **Lemma 6.** At the end of Algorithm 1, for all b -matching M' and for all $v \in V$, we have $w_v(Q_v) \geq w_v(M' \cap \delta(v))$.

Proof. By Proposition 5(ii), $w_v(Q_v)$ is exactly the sum of the reduced weights of the b_v heaviest elements in $S \cap \delta(v)$ (which are on top of the queues of Q_v). If we can show that for each element $e = e_t \in M' \setminus S$, $w_u(e_t) \leq \min\{w_v(Q_{v,q}^{[E]}) : 1 \leq q \leq b_v\}$, the proof will follow. Indeed, as e_t is discarded, we know that $w_v(e_t) = \min\{w_v(Q_{v,q}^{(t-1)}) : 1 \leq q \leq b_v\} \leq \min\{w_v(Q_{v,q}^{[E]}) : 1 \leq q \leq b_v\}$, where the inequality holds because the weight of a queue is monotonically increasing. ◀

- **Lemma 7.** $2g(S) \geq w(M^{opt})$.

Proof. It is clear that for $e = \{u, v\}$ we have $w_u(e) + w_v(e) \geq w(e)$. Therefore

$$\begin{aligned}
w(M^{opt}) &\leq \sum_{e=\{u,v\} \in M^{opt}} w_u(e) + w_v(e) = \sum_{u \in V} w_u(M^{opt} \cap \delta(u)) \\
&\leq \sum_{u \in V} w_u(Q_u) = \sum_{u \in V} g(S \cap \delta(u)) = 2g(S),
\end{aligned}$$

where the second inequality follows from Lemma 6 and the subsequent equality from Proposition 5(i). The last equality comes from the fact that an edge is incident to 2 vertices. ◀

Recall that $q_v(e)$ refers to the index of the particular queue in Q_v where a new edge e will be inserted (Line 6 of Algorithm 1).

► **Lemma 8.** *Algorithm 2 outputs a feasible b -matching M with weight $w(M) \geq g(S)$.*

Proof. By an easy induction, we know that for a given $e = e_t \in S$ and $v \in e$, we have:

$$w_v(e) = w_v^*(e) + g(e) = \sum_{e' \in Q_{v, q_v(e)}^{(t)}} g(e') \quad \text{and} \quad w(e) = g(e) + \sum_{u \in e} \sum_{e' \in Q_{u, q_u(e)}^{(t-1)}} g(e'). \quad (1)$$

Moreover, observe that $S \cap \delta(v)$ can be written as a disjoint union of the $Q_{v, q}$ for $1 \leq q \leq b_v$: $S \cap \delta(v) = \bigcup_{1 \leq q \leq b_v} Q_{v, q}$. One can also observe that Algorithm 2 takes at most one element in each queue $Q_{v, i}$. In fact, an element can be added only if no element above it in any of the queues where it appears has already been added into M ; and no element below it in the queues can be already part of M because S is read in the reverse arrival order. Consequently M respects the capacity constraint and is thus a feasible b -matching. We now make a critical claim from which the correctness of the lemma follows easily.

► **Claim 9.** Given an edge $e \in S$, either $e \in M$, or there exists another edge e' arriving later than e , such that $e' \in M$ and there exists a queue belonging to a common endpoint of e and e' , which contains both of them.

Observe that if the claim holds, by (1), the gain $g(e)$ of any edge $e \in S$ will be “paid” for by some edge $e' \in M$ and the proof will follow.

To prove the claim, let $e = \{u, v\}$ and assume that $e \notin M$. Consider the two queues $Q_{u, q_u(e)}$ and $Q_{v, q_v(e)}$. The edges stored above e in these two queues must have arrived later than e in S and have thus already been considered by Algorithm 2. The only reason that $e \notin M$ must be that $z_e = 0$ when e is processed, implying that one of these edges was already part of M . Hence the claim follows. ◀

Lemmas 7 and 8 give the following theorem:

► **Theorem 10.** *Algorithms 1 and 2 provide a 2 approximation for the maximum weight b -matching problem.*

We refer the readers to Appendix A for the details on how to handle the memory consumption of the algorithm.

► **Remark 11.** It is straightforward to extend our algorithm to a k -uniform hypergraph, where we can get an approximation ratio of k . Notice that if the k -uniform hypergraph is also k -partite, then the problem becomes that of finding a maximum weight intersection of k partition matroids. It can be shown that our stored edge set is exactly identical to the one stored by the algorithm of Garg et al. [9]. They have conjectured that for k arbitrary general matroids, their stored edge set always contains a k approximation. Our result thus proves their conjecture to be true when all matroids are partition matroids.

3 Submodular Function Maximization

3.1 Description of the Algorithm

For submodular function maximization, the streaming algorithm, formally described in Algorithm 3, is quite similar to the one for the weighted b -matching in the preceding section. Here notice that the element weight $w(e)$ is replaced by the marginal value $f(e | S)$ (see Lines 7 and 10). We use a similar randomization method to that of Levin and Wajc [12] for non-monotone functions (adding an element to S only with probability p , see Lines 8-9), and

Algorithm 3 Streaming phase for submodular function maximization

```

1:  $S \leftarrow \emptyset$ 
2:  $\forall v \in V : Q_v \leftarrow (Q_{v,1} = \emptyset, \dots, Q_{v,b_v} = \emptyset)$ 
3: for  $e = e_t, 1 \leq t \leq |E|$  an edge from the stream do
4:   for  $u \in e$  do
5:      $w_u^*(e) \leftarrow \min\{w_u(Q_{u,q}.top()) : 1 \leq q \leq b_u\}$ 
6:      $q_u(e) \leftarrow q$  such that  $w_u(Q_{u,q}.top()) = w_u^*(e)$ 
7:   if  $f(e | S) > \alpha \sum_{u \in e} w_u^*(e)$  then
8:      $\pi \leftarrow$  a random variable equal to 1 with probability  $p$  and 0 otherwise
9:     if  $\pi = 0$  then continue ▷ skip edge  $e$  with probability  $1 - p$ 
10:     $g(e) \leftarrow f(e | S) - \sum_{u \in e} w_u^*(e)$ 
11:     $S \leftarrow S \cup \{e\}$ 
12:    for  $u \in e$  do
13:       $w_u(e) \leftarrow w_u^*(e) + g(e)$ 
14:       $r_u(e) \leftarrow Q_{u,q_u(e)}.top()$ 
15:       $Q_{u,q_u(e)}.push(e)$ 

```

our analysis will bear much similarity to theirs. The greedy algorithm to build a solution M from S is still Algorithm 2.

Algorithm 3 uses, for $\alpha = 1 + \varepsilon$, $O(\log_{1+\varepsilon}(W/\varepsilon) \cdot |M_{max}|)$ variables, where M_{max} denotes the maximum cardinality b -matching and W denotes the maximum quotient $\frac{f(e|Y)}{f(e'|X)}$, for $X \subseteq Y \subseteq E$, $e, e' \in E$, $f(e'|X) > 0$ (in Appendix A we explain how to guarantee such space complexity when f is linear—the general case of a submodular function follows similar ideas).

3.2 Analysis for Monotone Submodular Function Maximization

Let $\alpha = 1 + \varepsilon$. In this section, $p = 1$ (so we have actually a deterministic algorithm for the monotone case). The following two lemmas relate the total gain $g(S)$ with the marginal values $f(S | \emptyset)$ and $f(M^{opt} | S)$.

► **Lemma 12.** *It holds that $g(S) \geq \frac{\varepsilon}{1+\varepsilon} f(S | \emptyset)$.*

Proof. For an element $e = e_t \in S$ we have $f(e | S^{(t-1)}) \geq (1 + \varepsilon) \sum_{u \in e} w_u^*(e)$ so

$$g(e) = f(e | S^{(t-1)}) - \sum_{u \in e} w_u^*(e) \geq f(e | S^{(t-1)}) \left(1 - \frac{1}{1 + \varepsilon}\right),$$

implying that

$$g(S) = \sum_{e \in S} g(e) \geq \sum_{e=e_t \in S} f(e | S^{(t-1)}) \left(1 - \frac{1}{1 + \varepsilon}\right) = \frac{\varepsilon}{1 + \varepsilon} f(S | \emptyset).$$

◀

As in the previous section, if an edge e is discarded, we assume that $w_v^*(e) = w_v(e)$ for each $v \in e$.

► **Lemma 13.** *It holds that $2(1 + \varepsilon)g(S) \geq f(M^{opt} | S)$.*

Proof. The only elements $e = e_t$ missing in S are the ones satisfying the inequality $f(e | S^{(t-1)}) \leq (1 + \varepsilon) \sum_{u \in e} w_u^*(e)$. So by submodularity,

$$\begin{aligned} f(M^{opt} | S) &\leq \sum_{e \in M^{opt} \setminus S} f(e | S) \leq \sum_{e = e_t \in M^{opt} \setminus S} f(e | S^{(t-1)}) \\ &\leq \sum_{e \in M^{opt} \setminus S} (1 + \varepsilon) \sum_{u \in e} w_u^*(e) = (1 + \varepsilon) \sum_{e \in M^{opt} \setminus S} \sum_{u \in e} w_u(e) \\ &= (1 + \varepsilon) \sum_{u \in V} w_u((M^{opt} \setminus S) \cap \delta(u)) \leq (1 + \varepsilon) \sum_{u \in V} w_u(Q_u) \\ &\leq 2(1 + \varepsilon)g(S), \end{aligned}$$

similar to the proof of Lemma 7. ◀

► **Lemma 14.** *Algorithm 2 outputs a feasible b -matching with $f(M) \geq g(S) + f(\emptyset)$.*

Proof. As argued in the proof of Lemma 8, M respects the capacities and so is feasible. Now, suppose that $M = \{e_{t_1}, \dots, e_{t_{|M|}}\}$, $t_1 < \dots < t_{|M|}$. Then

$$f(M) = f(\emptyset) + \sum_{i=1}^{|M|} f(e_{t_i} | \{e_{t_1}, \dots, e_{t_{i-1}}\}) \geq f(\emptyset) + \sum_{i=1}^{|M|} f(e_{t_i} | S^{(t_i-1)}) \geq f(\emptyset) + g(S),$$

as the values $f(e_{t_i} | S^{(t_i-1)})$ play the same role as the weights in Lemma 8. ◀

► **Theorem 15.** *Algorithms 3 and 2 provide a $3 + 2\sqrt{2}$ approximation if we set $\varepsilon = \frac{1}{\sqrt{2}}$.*

Proof. By Lemmas 12 and 13, we derive $(2 + 2\varepsilon + \frac{1+\varepsilon}{\varepsilon})g(S) \geq f(M^{opt} | S) + f(S | \emptyset) = f(M^{opt} \cup S | \emptyset) \geq f(M^{opt} | \emptyset)$, where the last inequality is due to the monotonicity of f . By Lemma 14, the output b -matching M guarantees that $f(M) \geq g(S) + f(\emptyset)$. As a result, $(3 + 2\varepsilon + \frac{1}{\varepsilon})f(M) \geq f(M^{opt} | \emptyset) + f(\emptyset) = f(M^{opt})$. Setting $\varepsilon = \frac{1}{\sqrt{2}}$ gives the result. ◀

► **Remark 16.** When $b_v = 1$ for all $v \in V$ (i.e. simple matching), our algorithm behaves exactly the same as the algorithm of Levin and Wajc [12]. Therefore their tight example also applies to our algorithm. In other words, our analysis of approximation ratio is tight.

3.3 Analysis for Non-Monotone Submodular Function Maximization

In this section, we suppose that $\frac{1}{3+2\varepsilon} \leq p \leq \frac{1}{2}$.

► **Lemma 17.** *It holds that*

$$\left(2(1 + \varepsilon) + \frac{1 + \varepsilon}{\varepsilon}\right) \mathbb{E}[g(S)] \geq \mathbb{E}[f(S \cup M^{opt} | \emptyset)].$$

Proof. From Lemma 12 we have that for any execution of the algorithm (a realization of randomness), the inequality $\frac{1+\varepsilon}{\varepsilon}g(S) \geq f(S | \emptyset)$ holds, so it is also true in expectation. We will try to prove in the following that $2(1 + \varepsilon)\mathbb{E}[g(S)] \geq \mathbb{E}[f(M^{opt} | S)]$, which is the counterpart of Lemma 13.

First, we show that for any $e \in M^{opt}$:

$$(1 + \varepsilon)\mathbb{E}\left[\sum_{u \in e} w_u(e)\right] \geq \mathbb{E}[f(e | S)] \tag{2}$$

We will use a conditioning similar to the one used in [12]. Let $e = e_t \in M^{opt}$. We consider the event $A_e = [f(e | S^{(t-1)}) \leq (1 + \varepsilon) \sum_{u \in e} w_u^*(e)]$. Notice that if A_e holds, e is not part of S and $w_v^*(e) = w_v(e)$ for each $v \in e$. Now by submodularity,

$$\begin{aligned} \mathbb{E}[f(e | S) | A_e] &\leq \mathbb{E}[f(e | S^{(t-1)}) | A_e] \leq \mathbb{E} \left[(1 + \varepsilon) \sum_{u \in e} w_u^*(e) | A_e \right] \\ &= (1 + \varepsilon) \mathbb{E} \left[\sum_{u \in e} w_u(e) | A_e \right] \end{aligned}$$

Next we consider the condition $\overline{A_e}$ (where the edge e should be added into S with probability p). As $p \leq \frac{1}{2}$, and for $e = e_t = \{u, v\}$ we have the inequality $w_u(e) + w_v(e) = 2f(e | S^{(t-1)}) - w_u^*(e) - w_v^*(e)$ when e is added to S , we get

$$\begin{aligned} \mathbb{E} \left[\sum_{u \in e} w_u(e) | \overline{A_e} \right] &= p \cdot \mathbb{E} \left[2f(e | S^{(t-1)}) - \sum_{u \in e} w_u^*(e) | \overline{A_e} \right] + (1 - p) \cdot \mathbb{E} \left[\sum_{u \in e} w_u^*(e) | \overline{A_e} \right] \\ &= 2p \cdot \mathbb{E} \left[f(e | S^{(t-1)}) | \overline{A_e} \right] + (1 - 2p) \cdot \mathbb{E} \left[\sum_{u \in e} w_u^*(e) | \overline{A_e} \right] \\ &\geq 2p \cdot \mathbb{E} \left[f(e | S^{(t-1)}) | \overline{A_e} \right]. \end{aligned}$$

As a result, for $p \geq \frac{1}{3+2\varepsilon}$,

$$\begin{aligned} (1 + \varepsilon) \mathbb{E} \left[\sum_{u \in e} w_u(e) | \overline{A_e} \right] &\geq 2p(1 + \varepsilon) \cdot \mathbb{E} \left[f(e | S^{(t-1)}) | \overline{A_e} \right] \\ &\geq (1 - p) \cdot \mathbb{E} \left[f(e | S^{(t-1)}) | \overline{A_e} \right] \\ &\geq \mathbb{E} \left[f(e | S) | \overline{A_e} \right], \end{aligned}$$

where the last inequality holds because with probability p we have $f(e | S) = 0$ (as $e \in S$) and with probability $1 - p$, $f(e | S) \leq f(e | S^{(t-1)})$ (by submodularity).

So we have proven inequality (2) and it follows that

$$\begin{aligned} \mathbb{E} [f(M^{opt} | S)] &\leq \sum_{e \in M^{opt}} \mathbb{E} [f(e | S)] \leq (1 + \varepsilon) \sum_{e \in M^{opt}} \mathbb{E} \left[\sum_{u \in e} w_u(e) \right] \\ &= (1 + \varepsilon) \sum_{u \in V} \sum_{e \in M^{opt} \cap \delta(u)} \mathbb{E} [w_u(e)] \leq (1 + \varepsilon) \sum_{u \in V} \mathbb{E} [w_u(Q_u)] \\ &= 2(1 + \varepsilon) \mathbb{E} [g(S)], \end{aligned}$$

where in the last inequality we use the fact that Lemma 6 holds for every realization of randomness.

Now the bounds on $\mathbb{E} [f(M^{opt} | S)]$ and the bound on $\mathbb{E} [f(S | \emptyset)]$ argued in the beginning give the proof of the lemma. \blacktriangleleft

Then we will use the following lemma, due to Buchbinder et al. [1]:

► Lemma 18 (Lemma 2.2 in [1]). *Let $h : 2^N \rightarrow \mathbb{R}_+$ be a non-negative submodular function, and let B be a random subset of N containing every element of N with probability at most p (not necessarily independently), then $\mathbb{E}[h(B)] \geq (1 - p)h(\emptyset)$.*

► **Theorem 19.** *Algorithm 3 run with $p = \frac{1}{3+2\varepsilon}$ provides a set S , upon which Algorithm 2 outputs a b -matching M satisfying:*

$$\left(\frac{4\varepsilon^2 + 8\varepsilon + 3}{2\varepsilon}\right) \mathbb{E}[f(M)] \geq f(M^{opt}).$$

This ratio is optimized when $\varepsilon = \frac{\sqrt{3}}{2}$, which gives a $4 + 2\sqrt{3}$ approximation.

Proof. Combining Lemma 14 and Lemma 17,

$$\left(2(1 + \varepsilon) + \frac{1 + \varepsilon}{\varepsilon}\right) \mathbb{E}[f(M)] \geq \mathbb{E}[f(S \cup M^{opt})].$$

Now we can apply Lemma 18 by defining $h : 2^E \rightarrow \mathbb{R}_+$ as, for any $X \subseteq E$, $h(X) = f(X \cup M^{opt})$ (trivially h is non-negative and submodular). As any element of E has the probability of at most p to appear in S , we derive $\mathbb{E}[f(S \cup M^{opt})] = \mathbb{E}[h(S)] \geq (1-p)h(\emptyset) = (1-p)f(M^{opt})$. Therefore,

$$\left(3 + 2\varepsilon + \frac{1}{\varepsilon}\right) \mathbb{E}[f(M)] \geq \mathbb{E}[f(S \cup M^{opt})] \geq (1-p)f(M^{opt}).$$

As $p = \frac{1}{3+2\varepsilon}$, we have

$$\left(\frac{4\varepsilon^2 + 8\varepsilon + 3}{2\varepsilon}\right) \cdot \mathbb{E}[f(M)] \geq f(M^{opt}).$$

This ratio is optimized when $\varepsilon = \frac{\sqrt{3}}{2}$, which gives a $4 + 2\sqrt{3} \approx 7.464$ approximation. ◀

4 Matroid-constrained Maximum Submodular b -Matching

In this section we consider the more general case of a b -matching on a k -uniform hypergraph and we impose a matroid constraint $\mathcal{M} = (E, \mathcal{I})$. A matching $M \subseteq E$ is feasible only if it respects the capacities of the vertices and is an independent set in the matroid \mathcal{M} .

4.1 Description of the Algorithm

For the streaming phase, our algorithm, formally described in Algorithm 4, is a generalization of Algorithm 3 in the last section. We let $\alpha = 1 + \varepsilon \geq 1$. For the matroid \mathcal{M} , we maintain a set of queues $Q_{\mathcal{M}} = \{Q_{\mathcal{M},1}, \dots, Q_{\mathcal{M},r_{\mathcal{M}}}\}$, where $r_{\mathcal{M}}$ is the rank of \mathcal{M} , to store the elements of S (so if an edge e is part of S , it appears in a total of $k+1$ queues, k of them corresponding to the vertices in e , and the remaining one corresponding to the matroid). To facilitate the presentation, we write $Top(Q_{\mathcal{M}})$ to denote the set of the elements on top of the queues of $Q_{\mathcal{M}}$. Lines 8-13 will guarantee that $Top(Q_{\mathcal{M}})$ is an independent set at any time (in fact a maximum weight independent set among all elements arrived so far, according to the reduced weights—see Lemma 21). Notice that in the end of the algorithm, $Top(Q_{\mathcal{M}})$ is not always fully-ranked (*i.e.* a base of $\mathcal{M} = (E, \mathcal{I})$), but it is always fully-ranked in the restricted matroid $\mathcal{M}|S = (S, \mathcal{I}_S)$, where $\mathcal{I}_S = \{I \subseteq S : I \in \mathcal{I}\}$.

Differently from the previous two sections where we have applied the greedy algorithm based on the reverse edge arrival order (Algorithm 2), here we introduce a new idea to build a feasible b -matching. Observe that after the streaming phase, $Top(Q_{\mathcal{M}})$ is an independent set (more precisely, a base of $\mathcal{M}|S$). Let $M = Top(Q_{\mathcal{M}})$. If M is a b -matching in the hypergraph, we are done. If not, some vertex v must have more than b_v incident edges in M . We discard

one such edge e in M (the choice of e will depend on the edge arrival order, see below) and replace it with another edge e' that arrived earlier than e (sometimes e is not replaced at all, as we will explain below). We guarantee $M \cup \{e'\} \setminus \{e\}$ remains independent in \mathcal{M} . The same procedure is repeated iteratively, where we replace newer elements of the independent set M by the older elements, until M becomes a feasible b -matching in the hypergraph. A formal description of this procedure is provided in Algorithm 5.

We now give a more detailed description. In Algorithm 5, an element e is *dominated* in the hypergraph by an element $e_d \in M$ if there exists $v \in V$ and $1 \leq i \leq b_v$ such that e and e_d are both in $Q_{v,i}$ and e_d arrived later than e in S . The algorithm starts with the maximum weight independent set $M = \text{Top}(Q_{\mathcal{M}})$ in $\mathcal{M}|S$, such that $w_{\mathcal{M}}(M) = g(S)$. Then, at each step, if the independent family M is not a b -matching, then some edge in M must be dominated by some other edge in M . We take among these dominated elements the latest one that arrived in S , and we replace it by an element e' of $C_e \setminus \{e\}$ that is not spanned by $M \setminus \{e\}$ (where C_e is defined as the circuit that is created by e in $\text{Top}(Q_{\mathcal{M}})$ when it is inserted, see Line 11 in Algorithm 4; if $C_e = \emptyset$ then we remove e without replacing it). Such an element e' always exists, otherwise e would have been spanned by $M \setminus \{e\}$ (see Lemma 24). Moreover this element satisfies $w_{\mathcal{M}}(e') \geq w_{\mathcal{M}}^*(e)$ (because $w_{\mathcal{M}}^*(e)$ is the minimum reduced weight among the reduced weights of the elements in $C_e \setminus \{e\}$) and we know that $g(e)$ is already “paid for” by the element e_d which dominates e (if $C_e = \emptyset$ then $w_{\mathcal{M}}^*(e) = 0$ and there is no need to add any element e' to compensate the loss). The element e' that replaces e arrived earlier than e in S because it is part of the circuit that was created when e arrived, so this newly arrived element e' cannot dominate e_d . As e was the latest dominated element, all future added elements will be elements that arrived earlier than e in S , so e_d will never be dominated in the next steps and $g(e)$ will be “paid for” by e_d until the end of the algorithm.

Regarding the memory consumption of the algorithm, it is easy to see that the number of variables used will be $O(\log_{1+\varepsilon}(W/\varepsilon) \cdot \min\{|M_{max}|, r_{\mathcal{M}}\})$, where $|M_{max}|$ denotes the maximum cardinality matching and W denotes the maximum quotient $\frac{f(e|Y)}{f(e'|X)}$, for $X \subseteq Y \subseteq E$, $e, e' \in E$, $f(e'|X) > 0$ (the idea is similar to the one in Appendix A). In fact, we do not need to actually store the circuits C_e , as we can get these circuits back during the construction phase by going back in time in the structure of the queues $Q_{\mathcal{M}}$.

4.2 Analysis for Linear Function Maximization

In this section, $p = 1$. For each discarded elements $e \in E \setminus S$, similarly as before, we set $w_{\mathcal{M}}(e) = w_{\mathcal{M}}^*(e)$. Moreover, for $e = e_t$ we set $w(e) = f(e|\emptyset) = f(e|S^{(t)})$ the weight of an element.

We introduce some basic facts in matroid theory, e.g., see [16].

► **Proposition 20.** *Given a matroid $\mathcal{M} = (E, \mathcal{I})$ with weight $w : E \rightarrow \mathbb{R}_+$, then*

- (i) *An independent set $I \in \mathcal{I}$ is a maximum weight base if and only if, for every element $e \in E \setminus I$, $I \cup \{e\}$ contains a circuit and $w(e) \leq \min_{e' \in C \setminus \{e\}} w(e')$.*
- (ii) *If $I \in \mathcal{I}$, $I \cup \{e\}$ contains a circuit C_1 and $I \cup \{e'\}$ contains a circuit C_2 and C_1 and C_2 contain a common element $e'' \in I$, then there exists another circuit $C_3 \subseteq (C_1 \cup C_2) \setminus \{e''\}$.*
- (iii) *If $X, Y \subseteq E$, $z \in E$ are such that $X \subseteq \text{span}(Y)$ and $z \in \text{span}(X)$, then $z \in \text{span}(Y)$.*

► **Lemma 21.** *Let $\{e_1, \dots, e_t\}$ be the set of edges arrived so far. Then the top elements $\text{Top}(Q_{\mathcal{M}}) = \text{Top}(Q_{\mathcal{M}}^{(t)})$ forms a maximum weight base in $\{e_1, \dots, e_t\}$ with regard to the reduced weight $w_{\mathcal{M}}$.*

Algorithm 4 Streaming phase for Matroid-constrained Maximum Submodular b -Matching

```

1:  $S \leftarrow \emptyset$ 
2:  $Q_{\mathcal{M}} \leftarrow (Q_{\mathcal{M},1} = \emptyset, \dots, Q_{\mathcal{M},r_{\mathcal{M}}} = \emptyset)$ 
3:  $\forall v \in V : Q_v \leftarrow (Q_{v,1} = \emptyset, \dots, Q_{v,b_v} = \emptyset)$ 
4: for  $e = e_t, 1 \leq t \leq |E|$  an edge from the stream do
5:   for  $u \in e$  do
6:      $w_u^*(e) \leftarrow \min\{w_u(Q_{u,q}.top()) : 1 \leq q \leq b_u\}$ 
7:      $q_u(e) \leftarrow q$  such that  $w_u(Q_{u,q}.top()) = w_u^*(e)$ 
8:   if  $Top(Q_{\mathcal{M}}) \cup \{e\} \in \mathcal{I}$ , i.e.  $C_e = \emptyset$  then
9:      $w_{\mathcal{M}}^*(e) \leftarrow 0$ 
10:     $q_{\mathcal{M}}(e) \leftarrow q$  such that  $Q_{\mathcal{M},q}$  is empty
11:   if  $Top(Q_{\mathcal{M}}) \cup \{e\}$  contains a circuit  $C_e$  then
12:      $w_{\mathcal{M}}^*(e) \leftarrow \min_{e' \in C_e \setminus \{e\}} w_{\mathcal{M}}(e')$ 
13:      $q_{\mathcal{M}}(e) \leftarrow q$  such that  $w_{\mathcal{M}}(Q_{\mathcal{M},q}.top())$  is equal to  $\min_{e' \in C_e \setminus \{e\}} w_{\mathcal{M}}(e')$  and
      $Q_{\mathcal{M},q}.top() \in C$ 
14:   if  $f(e|S) > \alpha(\sum_{u \in e} w_u^*(e) + w_{\mathcal{M}}^*(e))$  then
15:      $\pi \leftarrow$  a random variable equal to 1 with probability  $p$  and 0 otherwise
16:     if  $\pi = 0$  then continue  $\triangleright$  skip edge  $e$  with probability  $1 - p$ 
17:      $g(e) \leftarrow f(e|S) - \sum_{u \in e} w_u^*(e) - w_{\mathcal{M}}^*(e)$ 
18:      $S \leftarrow S \cup \{e\}$ 
19:     for  $u \in e$  do
20:        $w_u(e) \leftarrow w_u^*(e) + g(e)$ 
21:        $r_u(e) \leftarrow Q_{u,q_u(e)}.top()$ 
22:        $Q_{u,q_u(e)}.push(e)$ 
23:      $w_{\mathcal{M}}(e) \leftarrow w_{\mathcal{M}}^*(e) + g(e)$ 
24:      $r_{\mathcal{M}}(e) \leftarrow Q_{\mathcal{M},q_{\mathcal{M}}(e)}.top()$ 
25:      $Q_{\mathcal{M},q_{\mathcal{M}}(e)}.push(e)$ 

```

Algorithm 5 Construction phase for Matroid-constrained Maximum Submodular b -Matching

```

1:  $M \leftarrow Top(Q_{\mathcal{M}})$ 
2: while some element in  $M$  is dominated in the hypergraph by another element in  $M$  do
3:   let  $e$  be the latest dominated element in  $M$ 
4:   if  $C_e = \emptyset$  then
5:      $M \leftarrow M \setminus \{e\}$ 
6:   else
7:     let  $e'$  be an element in  $C_e \setminus (\text{span}(M \setminus \{e\}) \cup \{e\})$ 
8:      $M \leftarrow (M \cup \{e'\}) \setminus \{e\}$ 
return  $M$ 

```

Proof. This can be easily proved by induction on the number of edges arrived so far and Proposition 20(i) and (ii). ◀

► **Corollary 22.** *At the end of the algorithm, $w_{\mathcal{M}}(Q_{\mathcal{M}}) \geq w_{\mathcal{M}}(M^{opt})$.*

► **Lemma 23.** *It holds that $(1 + \varepsilon)(k + 1)g(S) \geq w(M^{opt})$.*

Proof. Using the fact that $w(e) \leq (1 + \varepsilon)(\sum_{u \in e} w_u(e) + w_{\mathcal{M}}(e))$ (observe that it is true when e is stored in S as well as when it is not) we get:

$$\begin{aligned} w(M^{opt}) &\leq (1 + \varepsilon) \sum_{e \in M^{opt}} \left(\sum_{u \in e} w_u(e) + w_{\mathcal{M}}(e) \right) \\ &= (1 + \varepsilon) \sum_{v \in V} w_v(M^{opt} \cap \delta(v)) + (1 + \varepsilon)w_{\mathcal{M}}(M^{opt}) \\ &\leq (1 + \varepsilon) \sum_{v \in V} g(S \cap \delta(v)) + (1 + \varepsilon)w_{\mathcal{M}}(Q_{\mathcal{M}}) \\ &= (1 + \varepsilon)(kg(S) + g(S)). \end{aligned}$$

For the last inequality we use Corollary 22 and Proposition 5. ◀

Now we turn our attention to Algorithm 5. First, we check that this algorithm performs legitimate operations and terminates.

► **Lemma 24.** (i) *The element e' defined at Line 7 in Algorithm 5 is well-defined.*

(ii) *Through the execution of the algorithm, M remains an independent set in \mathcal{M} .*

Proof. To prove (i), suppose e' does not exist. If such an element does not exist, then it means that we have the inclusion $C_e \setminus \{e\} \subseteq \text{span}(M \setminus \{e\})$. In addition, C_e is a circuit so $e \in \text{span}(C_e \setminus \{e\})$. As a result, we should have $e \in \text{span}(M \setminus \{e\})$ (see Proposition 20(iii)), in contradiction with the fact that M is independent in \mathcal{M} .

For the point (ii), observe that, at the beginning, M is an independent set because of Lemma 21. Then, if M is an independent set, then so is $M \setminus \{e\}$. Moreover, if e is replaced by an element $e' \in C_e \setminus (\text{span}(M \setminus \{e\}) \cup \{e\})$, the set $(M \setminus \{e\}) \cup \{e'\}$ is still an independent set, as e' is not in the span of $M \setminus \{e\}$. ◀

► **Lemma 25.** *Algorithm 5 terminates and outputs a feasible b -matching.*

Proof. The algorithm terminates because an element is either not replaced or replaced by an element that arrived earlier in S . By Lemma 24(ii), through the execution of the algorithm, M remains an independent set in the matroid. In addition, when there is no domination relation between elements in M , then M must be a b -matching (as there is at most one element per queue) in the hypergraph. ◀

Let us define the set D of elements that were removed so far from M during the execution of Algorithm 5 (because they were dominated in the hypergraph).

► **Lemma 26.** *At any time during the execution of Algorithm 5, the elements in D are dominated by some elements in M .*

Proof. When an element e is added to D , it means that at that point it is the latest dominated element in M , and that it is dominated by some other element e_d that arrived later. As elements are always replaced by other elements that arrived earlier, and because all the subsequent dominated elements considered by the algorithm will be edges that arrived earlier than e , this edge e_d will not be removed from M until the end of the execution the algorithm. ◀

► **Lemma 27.** *At any time during the execution of Algorithm 5, it holds that $w(M) \geq w_{\mathcal{M}}(M) + g(D)$.*

Proof. We have

$$\begin{aligned} w(M) &= w_{\mathcal{M}}(M) + \sum_{e \in M} \sum_{v \in e} g(\{e' \text{ dominated by } e \text{ in } Q_{v, q_v(e)}\}) \\ &\geq w_{\mathcal{M}}(M) + g(D), \end{aligned}$$

because, by the previous lemma, D is a subset of the elements that are actually dominated in the hypergraph (and an element can be dominated multiple times in the real computation of $w(M)$). ◀

► **Lemma 28.** *At any time during the execution of Algorithm 5, $w_{\mathcal{M}}(M) + g(D) \geq g(S)$.*

Proof. We prove by induction on the number of times the while loop is executed. At the beginning, $M = \text{Top}(Q_{\mathcal{M}})$ so $w(M) \geq w_{\mathcal{M}}(Q_{\mathcal{M}}) = g(S)$, and $D = \emptyset$. For the induction step, suppose that we have to remove $e \in M$ from M because it is dominated by $e_d \in M$. If $C_e = \emptyset$ it means that $w_{\mathcal{M}}(e) = g(e)$, then $w_{\mathcal{M}}((M \setminus \{e\})) + g(D \cup \{e\}) = w_{\mathcal{M}}(M) + g(D) \geq g(S)$. If $C_e \neq \emptyset$, then the replacing element e' guarantees that $(M \setminus \{e\}) \cup \{e'\} \in \mathcal{I}$ and $w_{\mathcal{M}}(e') \geq w_{\mathcal{M}}^*(e)$ so the new value $w_{\mathcal{M}}((M \setminus \{e\}) \cup \{e'\}) + g(D \cup \{e\}) = w_{\mathcal{M}}(M) - g(e) - w_{\mathcal{M}}^*(e) + w_{\mathcal{M}}(e') + g(D) + g(e) \geq w_{\mathcal{M}}(M) + g(D) \geq g(S)$ also respects the inequality. ◀

► **Lemma 29.** *Algorithm 5 returns a feasible b -matching M such that $w(M) \geq g(S)$.*

Proof. This follows easily from the previous lemmas. ◀

Therefore combining the previous results we get a $(1 + \varepsilon)(k + 1)$ approximation streaming algorithm for the matroid-constrained maximum weight b -matching problem:

► **Theorem 30.** *Algorithm 4 combined with Algorithm 5 provides a $(1 + \varepsilon)(k + 1)$ approximation for the maximum weight b -matching problem under a matroid constraint.*

► **Remark 31.** When the hypergraph is k -partite, our algorithm builds the set S exactly as the algorithm by Garg et al. [9] does. As a result, it proves the conjecture they formulated in the case where we have k partition matroids and one general matroid: we can extract from S a $k + 1$ approximation in polynomial time, if ε is set to 0.

4.3 Analysis for Monotone Submodular Function Maximization

In this section, $p = 1$ and $\varepsilon > 0$. The next two lemmas relate the total gain $g(S)$ with $f(S | \emptyset)$ and $f(M^{\text{opt}} | S)$.

► **Lemma 32.** *It holds that $g(S) \geq \frac{\varepsilon}{1 + \varepsilon} f(S | \emptyset)$.*

Proof. Same proof as for Lemma 12. ◀

► **Lemma 33.** *It holds that $(1 + \varepsilon)(k + 1)g(S) \geq f(M^{\text{opt}} | S)$.*

Proof. By the same argument as in the proof of Lemma 13, we have

$$\sum_{e \in M^{\text{opt}} \setminus S} (1 + \varepsilon) \sum_{u \in e} w_u^*(e) \leq (1 + \varepsilon)k \cdot g(S).$$

Moreover, Corollary 22 shows that $g(S) = w_{\mathcal{M}}(Q_{\mathcal{M}}) \geq w_{\mathcal{M}}(M^{opt})$ and we know that

$$w_{\mathcal{M}}(M^{opt}) \geq \sum_{e \in M^{opt} \setminus S} w_{\mathcal{M}}(e) = \sum_{e \in M^{opt} \setminus S} w_{\mathcal{M}}^*(e).$$

As a result, we obtain

$$\begin{aligned} f(M^{opt} | S) &\leq \sum_{e \in M^{opt} \setminus S} f(e | S) \leq \sum_{e=e_t \in M^{opt} \setminus S} f(e | S^{(t-1)}) \\ &\leq (1 + \varepsilon) \left(\sum_{e \in M^{opt} \setminus S} \sum_{u \in e} w_u^*(e) + w_{\mathcal{M}}^*(e) \right) \\ &\leq (1 + \varepsilon)(k + 1)g(S). \end{aligned}$$

◀

► **Lemma 34.** *It holds that $((1 + \varepsilon)(k + 1) + 1 + \frac{1}{\varepsilon})g(S) \geq f(M^{opt} | \emptyset)$.*

Proof. By Lemmas 32 and 33, we get $((1 + \varepsilon)(k + 1) + 1 + \frac{1}{\varepsilon})g(S) \geq f(M^{opt} | S) + f(S | \emptyset) = f(M^{opt} \cup S | \emptyset) \geq f(M^{opt} | \emptyset)$ because f is monotone. ◀

► **Lemma 35.** *With S as input, Algorithm 5 returns a feasible b -matching M satisfying $f(M) \geq g(S) + f(\emptyset)$.*

Proof. Proof similar to the one for Lemma 14 (using Lemma 29 instead of Lemma 8). ◀

As a result, we get the following theorem (the same way we obtained Theorem 15):

► **Theorem 36.** *For non-negative monotone submodular functions, Algorithm 4 with $p = 1$ combined with Algorithm 5 provides a feasible b -matching such that*

$$\left((1 + \varepsilon)(k + 1) + 1 + \frac{1}{\varepsilon} \right) f(M) \geq f(M^{opt}).$$

By setting $\varepsilon = \frac{1}{\sqrt{k+1}}$, we attain the approximation ratio of $k + 2\sqrt{k+1} + 2$ for all k .

4.4 Analysis for Non-Monotone Submodular Function Maximization

In this section, we assume $\frac{1}{1+(k+1)(1+\varepsilon)} \leq p \leq \frac{1}{k+1}$. The following lemma is the counterpart of Lemma 17, whose proof again uses the technique of conditioning (in a more general form).

► **Lemma 37.** *It holds that $((1 + \varepsilon)(k + 1) + \frac{1+\varepsilon}{\varepsilon})\mathbb{E}[g(S)] \geq \mathbb{E}[f(S \cup M^{opt} | \emptyset)]$.*

Proof. By Lemma 32, for any realization on randomness, we have $\frac{1+\varepsilon}{\varepsilon}g(S) \geq f(S | \emptyset)$, so the inequality also holds in expectation.

We next show that, for any $e \in M^{opt}$ we have

$$\mathbb{E}[f(e | S)] \leq (1 + \varepsilon)\mathbb{E}\left[\sum_{u \in e} w_u(e) + w_{\mathcal{M}}(e)\right]. \quad (3)$$

Let $e \in M^{opt}$. Conditioning on $A_e = [f(e | S^{(t-1)}) \leq (1 + \varepsilon)(\sum_{u \in e} w_u^*(e) + w_{\mathcal{M}}^*(e))]$ (i.e. e cannot be part of S), we have

$$\mathbb{E}[f(e | S) | A_e] \leq \mathbb{E}[f(e | S^{(t-1)}) | A_e]$$

$$\begin{aligned}
&\leq \mathbb{E} \left[(1 + \varepsilon) \left(\sum_{u \in e} w_u^*(e) + w_{\mathcal{M}}^*(e) \right) \mid A_e \right] \\
&= (1 + \varepsilon) \mathbb{E} \left[\sum_{u \in e} w_u(e) + w_{\mathcal{M}}(e) \mid A_e \right].
\end{aligned}$$

For the condition $\overline{A_e}$, recall that it means that with probability p , the edge is added into S and with probability $1 - p$, it is not. Thereby we obtain

$$\begin{aligned}
&\mathbb{E} \left[\sum_{u \in e} w_u(e) + w_{\mathcal{M}}(e) \mid \overline{A_e} \right] \\
&= p \cdot \mathbb{E} \left[(k + 1)f(e \mid S^{(t-1)}) - k \left(\sum_{u \in e} w_u^*(e) + w_{\mathcal{M}}^*(e) \right) \mid \overline{A_e} \right] \\
&\quad + (1 - p) \cdot \mathbb{E} \left[\sum_{u \in e} w_u^*(e) + w_{\mathcal{M}}^*(e) \mid \overline{A_e} \right] \\
&= (k + 1) \cdot p \cdot \mathbb{E} \left[f(e \mid S^{(t-1)}) \mid \overline{A_e} \right] \\
&\quad + (1 - (k + 1) \cdot p) \cdot \mathbb{E} \left[\sum_{u \in e} w_u^*(e) + w_{\mathcal{M}}^*(e) \mid \overline{A_e} \right] \\
&\geq (k + 1) \cdot p \cdot \mathbb{E} \left[f(e \mid S^{(t-1)}) \mid \overline{A_e} \right],
\end{aligned}$$

where in the last inequality we use the fact that $p \leq \frac{1}{k+1}$.

Now as $p \geq \frac{1}{1+(k+1)(1+\varepsilon)}$, we have

$$\begin{aligned}
(1 + \varepsilon) \mathbb{E} \left[\sum_{u \in e} w_u(e) + w_{\mathcal{M}}(e) \mid \overline{A_e} \right] &\geq (1 + \varepsilon)(k + 1) \cdot p \cdot \mathbb{E} \left[f(e \mid S^{(t-1)}) \mid \overline{A_e} \right] \\
&\geq (1 - p) \cdot \mathbb{E} \left[f(e \mid S^{(t-1)}) \mid \overline{A_e} \right] \\
&\geq \mathbb{E} \left[f(e \mid S) \mid \overline{A_e} \right],
\end{aligned}$$

and we have established (3).

Similar to the proof of Lemma 17 we derive

$$\begin{aligned}
\mathbb{E} [f(M^{opt} \mid S)] &\leq \sum_{e \in M^{opt}} \mathbb{E} [f(e \mid S)] \\
&\leq (1 + \varepsilon) \sum_{e \in M^{opt}} \mathbb{E} \left[\sum_{u \in e} w_u(e) \right] + (1 + \varepsilon) \mathbb{E} [w_{\mathcal{M}}(M^{opt})] \\
&\leq k(1 + \varepsilon) \mathbb{E} [g(S)] + (1 + \varepsilon) \mathbb{E} [w_{\mathcal{M}}(M^{opt})].
\end{aligned}$$

By Lemma 22 we know that, for any realization of randomness, we have $w_{\mathcal{M}}(M^{opt}) \leq w_{\mathcal{M}}(S_f) = g(S)$. Thus we get $\mathbb{E} [f(M^{opt} \mid S)] \leq (k + 1)(1 + \varepsilon) \mathbb{E} [g(S)]$.

Now the bound on $\mathbb{E} [f(M^{opt} \mid S)]$ and the bound on $\mathbb{E} [f(S \mid \emptyset)]$ (argued in the beginning of the proof) give us the lemma. \blacktriangleleft

Finally, using Lemma 18, we obtain the following theorem.

► **Theorem 38.** *For non-negative submodular functions, Algorithm 4 executed with $p = \frac{1}{1+(k+1)(1+\varepsilon)}$ combined with Algorithm 2 provides a b -matching M independent in \mathcal{M} such that:*

$$\frac{1 + (k+1)(1+\varepsilon)}{(k+1)(1+\varepsilon)} \left((1+\varepsilon)(k+1) + 1 + \frac{1}{\varepsilon} \right) \mathbb{E}[f(M)] \geq f(M^{opt}).$$

This ratio is optimized when $\varepsilon = \frac{\sqrt{k+2}}{k+1}$, providing a $k + 2\sqrt{k+2} + 3$ approximation.

5 Matchoid-constrained Maximum Submodular b -Matching

In this section we consider an even more general case of a b -matching on a k -uniform hypergraph on which we impose a k' -matchoid constraint $\mathcal{M} = (\mathcal{M}_i = (E_i, \mathcal{I}_i))_{i=1}^s$. A matching $M \subseteq E$ is feasible only if it respects the capacities of the vertices and $E_i \cap M$ is an independent set in \mathcal{M}_i for all i .

5.1 Description of the Algorithm

For the streaming phase, our algorithm, formally described in Algorithm 6, is an adaptation of Algorithm 3. We let $\alpha = 1 + \varepsilon$ and $\gamma > 1$. Here S as before denotes the set of elements stored so far. Additionally, we maintain a set $S_f \subseteq S$, an independent set in \mathcal{M} . The set S_f is, in a sense, equivalent to the set of top elements $Top(Q_{\mathcal{M}})$ in the previous section when \mathcal{M} was a simple matroid. Note that here we do not use any special data structures associated with the matchoid \mathcal{M} . For a newly-arrived element e , its reduced weight in the matchoid \mathcal{M} is determined by the set of circuits contained in $S_f \cup \{e\}$ (see Lines 8-11). If the new element e is to be stored (see Lines 8-11, 17 and 23), it replaces the lightest elements of the circuits in $S_f \cup \{e\}$.³ In the end, the set S_f (instead of all stored elements S) is given to the simple greedy Algorithm 2 as in Sections 2 and 3 to produce a feasible b -matching.

Here we give some intuition. We retain only the elements in S_f (as input to the greedy algorithm) because they are independent in every matroid \mathcal{M}_i (hence any subset of them chosen by the greedy algorithm), releasing us from the worry that the output is not independent in \mathcal{M} . We use $\gamma > 1$ to decide the relative importance of the matchoid \mathcal{M} (see Line 13). A larger γ guarantees that the gains of new edges grows quickly when S_f is updated. By doing this, S_f , by itself, contributes to a significant fraction of all gains in $g(S)$ (see Lemma 42). However, an overly large γ causes us to throw away too many edges, thus hurting the final approximation ratio. To optimize, we need to choose γ carefully.

Regarding the memory consumption of the algorithm, it is easy to see that the number of variables used would be a $O(\log_{1+\varepsilon}(W/\varepsilon) \cdot |M_{max}|)$, where M_{max} denotes the maximum cardinality b -matching in the graph and W denotes the maximum quotient $\frac{f(e|Y)}{f(e'|X)}$, for $X \subseteq Y \subseteq E$, $e, e' \in E$, $f(e'|X) > 0$.

5.2 Analysis for Monotone Submodular Function Maximization

In this section, $p = 1$. For each discarded element $e \in E \setminus S$, similarly as before, we set $w_{\mathcal{M}}(e) = w_{\mathcal{M}}^*(e)$. Moreover, we define an auxiliary reduced weight $w'_{\mathcal{M}}$ such that $w'_{\mathcal{M}}(e) = w_{\mathcal{M}}(e)$ if $e \in S$ and $w'_{\mathcal{M}}(e) = \gamma \cdot (1 + \varepsilon) \cdot w_{\mathcal{M}}(e)$ otherwise.

³ We note that how the elements in S_f are updated here are based on the same idea as in [2, 3] when constraint is only a p -matchoid without the hypergraph.

Algorithm 6 Streaming phase for Matchoid-constrained Maximum Submodular b -Matching

```

1:  $S \leftarrow \emptyset$ 
2:  $S_f \leftarrow \emptyset$ 
3:  $\forall v \in V : Q_v \leftarrow (Q_{v,1} = \emptyset, \dots, Q_{v,b_v} = \emptyset)$ 
4: for  $e = e_t, 1 \leq t \leq |E|$  an edge from the stream do
5:   for  $u \in e$  do
6:      $w_u^*(e) \leftarrow \min\{w_u(Q_{u,q}.top()) : 1 \leq q \leq b_u\}$ 
7:      $q_u(e) \leftarrow q$  such that  $w_u(Q_{u,q}.top()) = w_u^*(e)$ 
8:    $C_e \leftarrow \emptyset$ 
9:   for  $i$  such that  $e \in E_i$  do
10:    if  $(S_f \cap E_i) \cup \{e\}$  contains a circuit  $X$  in  $\mathcal{M}_i$  then
11:       $C_e \leftarrow C_e \cup \{\arg \min_{e' \in X \setminus \{e\}} w_{\mathcal{M}}(e')\}$ 
12:     $w_{\mathcal{M}}^*(e) \leftarrow w_{\mathcal{M}}(C_e)$ 
13:    if  $f(e|S) > \alpha(\sum_{u \in e} w_u^*(e) + \gamma \cdot w_{\mathcal{M}}^*(e))$  then
14:       $\pi \leftarrow$  a random variable equal to 1 with probability  $p$  and 0 otherwise
15:      if  $\pi = 0$  then continue ▷ skip edge  $e$  with probability  $1 - p$ 
16:       $g(e) \leftarrow f(e|S) - \sum_{u \in e} w_u^*(e) - w_{\mathcal{M}}^*(e)$ 
17:       $S \leftarrow S \cup \{e\}$ 
18:      for  $u \in e$  do
19:         $w_u(e) \leftarrow w_u^*(e) + g(e)$ 
20:         $r_u(e) \leftarrow Q_{u,q_u(e)}.top()$ 
21:         $Q_{u,q_u(e)}.push(e)$ 
22:       $w_{\mathcal{M}}(e) \leftarrow w_{\mathcal{M}}^*(e) + g(e)$ 
23:       $S_f \leftarrow (S_f \cup \{e\}) \setminus C_e$ 
24: update the values of  $r_v$  for  $v \in V$  as necessary so that only the elements in  $S_f$  are
    considered
  
```

The way S_f is built can be seen as an execution of the algorithm of Chekuri et al. [3] where elements of E are given weights $w'_{\mathcal{M}}$. Using their results we know that:

► **Proposition 39.** *Let $M' \subseteq E$ satisfying $M' \cap E_i \in \mathcal{I}_i$ for all i . Then we have*

$$\frac{(\gamma \cdot (1 + \varepsilon))^2}{\gamma \cdot (1 + \varepsilon) - 1} k' \cdot w_{\mathcal{M}}(S_f) = \frac{(\gamma \cdot (1 + \varepsilon))^2}{\gamma \cdot (1 + \varepsilon) - 1} k' \cdot w'_{\mathcal{M}}(S_f) \geq w'_{\mathcal{M}}(M')$$

and (this inequality is from Lemma 11 of the arXiv version of [3])

$$\left(\frac{(\gamma \cdot (1 + \varepsilon))^2}{\gamma \cdot (1 + \varepsilon) - 1} k' - \frac{\gamma \cdot (1 + \varepsilon)}{\gamma \cdot (1 + \varepsilon) - 1} \right) \cdot w_{\mathcal{M}}(S_f) \geq \gamma \cdot (1 + \varepsilon) \cdot w_{\mathcal{M}}(M' \setminus S).$$

The next two lemmas relate the total gain $g(S)$ with $f(S|\emptyset)$ and $f(M^{opt}|S)$.

► **Lemma 40.** *It holds that $g(S) \geq \frac{\varepsilon}{1+\varepsilon} f(S|\emptyset)$.*

Proof. Same proof as for Lemma 12. ◀

► **Lemma 41.** *It holds that*

$$\left((1 + \varepsilon)k + \frac{(\gamma \cdot (1 + \varepsilon))^2}{\gamma \cdot (1 + \varepsilon) - 1} k' - \frac{\gamma \cdot (1 + \varepsilon)}{\gamma \cdot (1 + \varepsilon) - 1} \right) g(S) \geq f(M^{opt}|S).$$

Proof. By the same argument as in the proof of Lemma 13, we have

$$\sum_{e \in M^{opt} \setminus S} (1 + \varepsilon) \sum_{u \in e} w_u^*(e) \leq (1 + \varepsilon)k \cdot g(S).$$

Moreover, we know that $g(S) = w_{\mathcal{M}}(S_f)$ and by Proposition 39:

$$\left(\frac{(\gamma \cdot (1 + \varepsilon))^2}{\gamma \cdot (1 + \varepsilon) - 1} k' - \frac{\gamma \cdot (1 + \varepsilon)}{\gamma \cdot (1 + \varepsilon) - 1} \right) w_{\mathcal{M}}(S_f) \geq \gamma \cdot (1 + \varepsilon) \cdot w_{\mathcal{M}}^*(M^{opt} \setminus S).$$

As a result, we obtain

$$\begin{aligned} f(M^{opt} | S) &\leq \sum_{e \in M^{opt} \setminus S} f(e | S) \leq \sum_{e = e_t \in M^{opt} \setminus S} f(e | S^{(t-1)}) \\ &\leq (1 + \varepsilon) \left(\sum_{e \in M^{opt} \setminus S} \sum_{u \in e} w_u^*(e) + \gamma \cdot w_{\mathcal{M}}^*(e) \right) \\ &\leq \left((1 + \varepsilon)k + \frac{(\gamma \cdot (1 + \varepsilon))^2}{\gamma \cdot (1 + \varepsilon) - 1} k' - \frac{\gamma \cdot (1 + \varepsilon)}{\gamma \cdot (1 + \varepsilon) - 1} \right) g(S). \end{aligned}$$

◀

The following lemma states that S_f retains a reasonably large fraction of the gains compared to S . For this lemma, we use the notion of *erasing tree*. The erasing tree T_e of an element $e \in S$ is defined recursively: it is made of the elements e' of C_e that were evicted from S_f when e was inserted and of the elements of the erasing trees $T_{e'}$ associated with these evicted elements.

► **Lemma 42.** *It holds that $\left(1 + \frac{1}{\gamma \cdot (1 + \varepsilon) - 1}\right) g(S_f) \geq g(S)$.*

Proof. We have, for all element $e \in S_f$,

$$g(e) \geq (\gamma \cdot (1 + \varepsilon) - 1) w_{\mathcal{M}}^*(e) = (\gamma \cdot (1 + \varepsilon) - 1) \sum_{e' \in T_e} g(e').$$

where T_e denotes the elements that are in the erasing tree of e , then

$$g(S) = \sum_{e \in S_f} \left(g(e) + \sum_{e' \in T_e} g(e') \right) \leq \sum_{e \in S_f} \left(1 + \frac{1}{\gamma \cdot (1 + \varepsilon) - 1} \right) g(e),$$

and the proof follows. ◀

► **Lemma 43.** *It holds that*

$$\begin{aligned} \left(1 + \frac{1}{\gamma \cdot (1 + \varepsilon) - 1} \right) \cdot \left((1 + \varepsilon)k + \frac{(\gamma \cdot (1 + \varepsilon))^2}{\gamma \cdot (1 + \varepsilon) - 1} k' - \frac{\gamma \cdot (1 + \varepsilon)}{\gamma \cdot (1 + \varepsilon) - 1} + \frac{1 + \varepsilon}{\varepsilon} \right) g(S_f) \\ \geq f(M^{opt} | \emptyset). \end{aligned}$$

Proof. By Lemmas 40 and 41, we have that

$$\begin{aligned} \left(1 + \frac{1}{\gamma \cdot (1 + \varepsilon) - 1} \right) \cdot \left((1 + \varepsilon)k + \frac{(\gamma \cdot (1 + \varepsilon))^2}{\gamma \cdot (1 + \varepsilon) - 1} k' - \frac{\gamma \cdot (1 + \varepsilon)}{\gamma \cdot (1 + \varepsilon) - 1} + 1 + \frac{1}{\varepsilon} \right) g(S_f) \\ \geq f(M^{opt} | S) + f(S | \emptyset) = f(M^{opt} \cup S | \emptyset) \geq f(M^{opt} | \emptyset) \end{aligned}$$

because f is monotone. Then we use Lemma 42. ◀

► **Lemma 44.** *With S_f as input, Algorithm 2 returns a feasible b -matching M with $f(M) \geq g(S_f) + f(\emptyset)$.*

Proof. As argued in Lemma 8, M respects the capacities. Furthermore, as S_f is by construction an independent set in \mathcal{M} and $M \subseteq S_f \in \mathcal{I}_i$ for all i , we have $M \in \mathcal{I}_i$ for all i . So M is a feasible b -matching. Finally, using an analysis similar to the one in the proof of Lemma 14, we have $f(M) \geq g(S_f) + f(\emptyset)$ (the only difference being that now the "weight" $f(e_t | S^{(t-1)})$ of an element can be higher than the sum of the gains of the elements below it in the queues, which is not an issue for the analysis). ◀

As a result, we get the following theorem (the same way we obtained Theorem 15):

► **Theorem 45.** *For non-negative monotone submodular functions, Algorithm 6 with $p = 1$ combined with Algorithm 2 provides a feasible b -matching such that*

$$\left(1 + \frac{1}{\gamma \cdot (1 + \varepsilon) - 1}\right) \cdot \left((1 + \varepsilon)k + \frac{(\gamma \cdot (1 + \varepsilon))^2}{\gamma \cdot (1 + \varepsilon) - 1}k' - \frac{\gamma \cdot (1 + \varepsilon)}{\gamma \cdot (1 + \varepsilon) - 1} + 1 + \frac{1}{\varepsilon}\right) f(M) \geq f(M^{opt}).$$

By setting $\varepsilon = 1$ and $\gamma = 2$, we attain the approximation ratio of $\frac{8}{3}k + \frac{64}{9}k' + \frac{8}{9}$ for all k and k' .

It is possible to obtain better ratios for fixed k and k' by a more careful choice of the parameters ε and γ .

5.3 Analysis for Non-Monotone Submodular Function Maximization

In this section, we assume $\frac{1}{2+k(1+\varepsilon)} \leq p \leq \frac{1}{k+1}$. The following lemma is the counterpart of Lemma 17, whose proof again uses the technique of conditioning (in a more general form).

► **Lemma 46.** *We have $\left((1 + \varepsilon)k + \frac{(\gamma \cdot (1 + \varepsilon))^2}{\gamma \cdot (1 + \varepsilon) - 1}k' + \frac{1 + \varepsilon}{\varepsilon}\right) \mathbb{E}[g(S)] \geq \mathbb{E}[f(S \cup M^{opt} | \emptyset)]$.*

Proof. By Lemma 40, for any realization on randomness, we have $\frac{1 + \varepsilon}{\varepsilon}g(S) \geq f(S | \emptyset)$, so the inequality also holds in expectation.

We next show that, for any $e \in M^{opt}$ we have

$$\mathbb{E}[f(e | S)] \leq (1 + \varepsilon) \mathbb{E} \left[\sum_{u \in e} w_u(e) + \frac{w'_{\mathcal{M}}(e)}{1 + \varepsilon} \right]. \quad (4)$$

Let $e \in M^{opt}$. Conditioning on the event $A_e = [f(e | S^{(t-1)}) \leq (1 + \varepsilon)(\sum_{u \in e} w_u^*(e) + \gamma \cdot w_{\mathcal{M}}^*(e))]$ (i.e. e cannot be part of S), we have

$$\begin{aligned} \mathbb{E}[f(e | S) | A_e] &\leq \mathbb{E}[f(e | S^{(t-1)}) | A_e] \\ &\leq \mathbb{E} \left[(1 + \varepsilon) \left(\sum_{u \in e} w_u^*(e) + \gamma \cdot w_{\mathcal{M}}^*(e) \right) | A_e \right] \\ &= (1 + \varepsilon) \mathbb{E} \left[\sum_{u \in e} w_u(e) + \frac{1}{1 + \varepsilon} \cdot w'_{\mathcal{M}}(e) | A_e \right]. \end{aligned}$$

For the condition $\overline{A_e}$, recall that it means that with probability p , the edge is added into S and with probability $1 - p$, it is not. So

$$\begin{aligned}
& \mathbb{E} \left[\sum_{u \in e} w_u(e) + \frac{1}{1+\varepsilon} \cdot w'_{\mathcal{M}}(e) \mid \overline{A_e} \right] \\
&= p \cdot \mathbb{E} \left[\left(k + \frac{1}{1+\varepsilon} \right) f(e \mid S^{(t-1)}) - \left(k + \frac{1}{1+\varepsilon} - 1 \right) \sum_{u \in e} w_u^*(e) + k w'_{\mathcal{M}}(e) \mid \overline{A_e} \right] \\
&\quad + (1-p) \cdot \mathbb{E} \left[\sum_{u \in e} w_u^*(e) + \gamma \cdot w'_{\mathcal{M}}(e) \mid \overline{A_e} \right] \\
&\geq p \cdot \mathbb{E} \left[\left(k + \frac{1}{1+\varepsilon} \right) f(e \mid S^{(t-1)}) - k \sum_{u \in e} w_u^*(e) - k \cdot \gamma \cdot w'_{\mathcal{M}}(e) \mid \overline{A_e} \right] \\
&\quad + (1-p) \cdot \mathbb{E} \left[\sum_{u \in e} w_u^*(e) + \gamma \cdot w'_{\mathcal{M}}(e) \mid \overline{A_e} \right] \\
&= \left(k + \frac{1}{1+\varepsilon} \right) \cdot p \cdot \mathbb{E} \left[f(e \mid S^{(t-1)}) \mid \overline{A_e} \right] \\
&\quad + (1 - (k+1) \cdot p) \cdot \mathbb{E} \left[\sum_{u \in e} w_u^*(e) + \gamma \cdot w'_{\mathcal{M}}(e) \mid \overline{A_e} \right] \\
&\geq \left(k + \frac{1}{1+\varepsilon} \right) \cdot p \cdot \mathbb{E} \left[f(e \mid S^{(t-1)}) \mid \overline{A_e} \right],
\end{aligned}$$

where in the second step we use the fact that $\gamma > 1$ and in the last inequality that $p \leq \frac{1}{k+1}$. Now as $p \geq \frac{1}{2+k(1+\varepsilon)}$, we have

$$\begin{aligned}
(1+\varepsilon) \mathbb{E} \left[\sum_{u \in e} w_u(e) + \frac{w'_{\mathcal{M}}(e)}{1+\varepsilon} \mid \overline{A_e} \right] &\geq (1+\varepsilon) \left(k + \frac{1}{1+\varepsilon} \right) \\
&\quad \cdot p \cdot \mathbb{E} \left[f(e \mid S^{(t-1)}) \mid \overline{A_e} \right] \\
&\geq (1-p) \cdot \mathbb{E} \left[f(e \mid S^{(t-1)}) \mid \overline{A_e} \right] \\
&\geq \mathbb{E} \left[f(e \mid S) \mid \overline{A_e} \right],
\end{aligned}$$

and we have established (4).

Similar to the proof of Lemma 17 we get

$$\begin{aligned}
\mathbb{E} [f(M^{opt} \mid S)] &\leq \sum_{e \in M^{opt}} \mathbb{E} [f(e \mid S)] \\
&\leq (1+\varepsilon) \sum_{e \in M^{opt}} \mathbb{E} \left[\sum_{u \in e} w_u(e) \right] + \mathbb{E} [w'_{\mathcal{M}}(M^{opt})] \\
&\leq k(1+\varepsilon) \mathbb{E} [g(S)] + \mathbb{E} [w'_{\mathcal{M}}(M^{opt})].
\end{aligned}$$

By Proposition 39 we know that for any realization of randomness,

$$w'_{\mathcal{M}}(M^{opt}) \leq \frac{(\gamma \cdot (1+\varepsilon))^2}{\gamma \cdot (1+\varepsilon) - 1} k' w'_{\mathcal{M}}(S_f) = \frac{(\gamma \cdot (1+\varepsilon))^2}{\gamma \cdot (1+\varepsilon) - 1} k' \cdot g(S).$$

Thus we get $\mathbb{E} [f(M^{opt} \mid S)] \leq \left((1+\varepsilon)k + \frac{(\gamma \cdot (1+\varepsilon))^2}{\gamma \cdot (1+\varepsilon) - 1} k' \right) \mathbb{E} [g(S)]$.

Now the bound on $\mathbb{E} [f(M^{opt} \mid S)]$ and the bound on $\mathbb{E} [f(S \mid \emptyset)]$ (argued in the beginning of the proof) give us the lemma. \blacktriangleleft

Finally, using Lemma 18 we obtain:

► **Theorem 47.** *For non-negative submodular functions, Algorithm 6 with $p = \frac{1}{2+k(1+\varepsilon)}$ combined with Algorithm 2 provides a b -matching M independent in \mathcal{M} such that:*

$$\frac{2+k(1+\varepsilon)}{1+k(1+\varepsilon)} \cdot \left(1 + \frac{1}{\gamma \cdot (1+\varepsilon) - 1}\right) \cdot \left((1+\varepsilon)k + \frac{(\gamma \cdot (1+\varepsilon))^2}{\gamma \cdot (1+\varepsilon) - 1}k' + 1 + \frac{1}{\varepsilon}\right) \mathbb{E}[f(M)] \geq f(M^{opt}).$$

Setting $\varepsilon = 1$ and $\gamma = 2$ we obtain the ratio of $\frac{2k+2}{2k+1} \cdot \left(\frac{8}{3}k + \frac{64}{9}k' + \frac{8}{3}\right)$ for all k and k' .

As in the last section, it is possible to obtain better ratios for fixed k and k' by a more careful choice of the parameters ε and γ .

References

- 1 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proc. 25th SODA*, pages 1433–1452, 2014.
- 2 Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Math. Program.*, 154(1-2):225–247, 2015.
- 3 Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *Proc. 42nd ICALP*, pages 318–330, 2015.
- 4 Michael Crouch and D.M. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. *Leibniz International Proceedings in Informatics, LIPIcs*, 28:96–104, 09 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.96.
- 5 Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM Journal on Discrete Mathematics*, 25, 07 2009. doi:10.4230/LIPIcs.STACS.2010.2476.
- 6 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348:207–216, 12 2005. doi:10.1016/j.tcs.2005.09.013.
- 7 Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: Streaming submodular maximization with subsampling. In *NeurIPS*, pages 730–740, 2018.
- 8 Moran Feldman, Joseph (Seffi) Naor, Roy Schwartz, and Justin Ward. Improved approximations for k -exchange systems. In *Proc. 19th ESA*, pages 784–798, 2011. URL: <http://portal.acm.org/citation.cfm?id=2040572.2040658&coll=DL&dl=GUIDE&CFID=95852163&CFTOKEN=76709828>.
- 9 Paritosh Garg, Linus Jordan, and Ola Svensson. Semi-streaming algorithms for submodular matroid intersection. In *IPCO*, 2021.
- 10 Mohsen Ghaffari and David Wajc. Simplified and Space-Optimal Semi-Streaming $(2+\varepsilon)$ -Approximate Matching. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*, volume 69 of *OpenAccess Series in Informatics (OASICS)*, pages 13:1–13:8, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: <http://drops.dagstuhl.de/opus/volltexte/2018/10039>, doi:10.4230/OASICS.SOSA.2019.13.
- 11 Chien-Chung Huang and François Sellier. Semi-Streaming Algorithms for Submodular Function Maximization Under b -Matching Constraint. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*, volume 207 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: <https://drops.dagstuhl.de/opus/volltexte/2021/14707>, doi:10.4230/LIPIcs.APPROX/RANDOM.2021.14.

- 12 Roie Levin and David Wajc. Streaming submodular matching meets the primal-dual method. In *SODA*, pages 1914–1933, 2021.
- 13 Andrew McGregor. Finding graph matchings in data streams. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 170–181, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- 14 S. Muthukrishnan. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 1(2):117–236, August 2005. doi:10.1561/0400000002.
- 15 Ami Paz and Gregory Schwartzman. A $(2+\varepsilon)$ -approximation for maximum weight matching in the semi-streaming model. In *Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2153–2161, 2017. URL: <https://epubs.siam.org/doi/abs/10.1137/1.9781611974782.140>, arXiv:<https://epubs.siam.org/doi/pdf/10.1137/1.9781611974782.140>, doi:10.1137/1.9781611974782.140.
- 16 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.
- 17 Mariano Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62:1–20, 02 2008. doi:10.1007/s00453-010-9438-5.

A Making Algorithm 1 Memory-Efficient

We explain how to guarantee the space requirement promised in Theorem 1. In this section, w_{min} denotes the minimum non-zero value of the weight of an edge, and w_{max} the maximum weight of an edge. Moreover, we set $W = w_{max}/w_{min}$. We also define M_{max} as a given maximum cardinality b -matching.

Let $\alpha = (1 + \varepsilon) > 1$. In Algorithm 7 we add an edge e to S only if $w(e) > \alpha \sum_{u \in e} w_u^*(e)$ (Line 7). For the moment we set $d = 0$ in our analysis, ignoring Lines 14-16 of the algorithm.

Algorithm 7 Streaming phase for weighted matching, memory-efficient

```

1:  $S \leftarrow \emptyset$ 
2:  $\forall v \in V : Q_v \leftarrow (Q_{v,1} = \emptyset, \dots, Q_{v,b_v} = \emptyset)$ 
3: for  $e = e_t, 1 \leq t \leq |E|$  an edge from the stream do
4:   for  $u \in e$  do
5:      $w_u^*(e) \leftarrow \min\{w_u(Q_{u,q}.top()) : 1 \leq q \leq b_u\}$ 
6:      $q_u(e) \leftarrow q$  such that  $w_u(Q_{u,q}.top()) = w_u^*(e)$ 
7:     if  $w(e) > \alpha \sum_{u \in e} w_u^*(e)$  then ▷ stricter condition here
8:        $g(e) \leftarrow w(e) - \sum_{u \in e} w_u^*(e)$ 
9:        $S \leftarrow S \cup \{e\}$ 
10:    for  $u \in e$  do
11:       $w_u(e) \leftarrow w_u^*(e) + g(e)$ 
12:       $r_u(e) \leftarrow Q_{u,q_u(e)}.top()$ 
13:       $Q_{u,q_u(e)}.push(e)$ 
14:      if  $d = 1$  and  $Q_{u,q_u(e)}.length() > \beta$  then ▷ remove some small element
15:        let  $e'$  be the  $\beta + 1$ -th element from the top of  $Q_{u,q_u(e)}$ 
16:        mark  $e'$  as erasable, so that when it will no longer be on the top of any
        queue, it will be removed from  $S$  and from all the queues it appears in

```

► **Lemma 48.** *The set S obtained at the end of algorithm 7 when $d = 0$ guarantees that $2\alpha g(S) \geq w(M^{opt})$.*

Proof. We proceed as in [9]. Let $w_\alpha : E \rightarrow \mathbb{R}$ such that $w_\alpha(e) = w(e)$ for $e \in S$ and $w_\alpha(e) = \frac{w(e)}{\alpha}$ for $e \in E \setminus S$. We can observe that with the weights w_α , Algorithm 1 gives the same set S as Algorithm 7 with the weights w . We deduce that $w_\alpha(M^{opt}) \leq 2g(S)$ and then, as $w \leq \alpha w_\alpha$, we get $2\alpha g(S) \geq w(M^{opt})$. \blacktriangleleft

Hence, using same arguments as in [10, 12] we obtain the following:

► **Theorem 49.** *Algorithm 7 (with $d = 0$) combined with Algorithm 2 gives a $2 + \varepsilon$ approximation algorithm by using $O(\log_{1+\varepsilon}(W/\varepsilon) \cdot |M_{max}|)$ variables.*

Proof. In a given queue, the minimum non-zero value that can be attained is $\frac{\varepsilon}{1+\varepsilon}w_{min}$ and the maximum value that can be attained is w_{max} . As the value of the top element of the queue increases at least by a factor $1 + \varepsilon$ for each inserted element, a given queue contains at most $\log_{1+\varepsilon}(W/\varepsilon) + 1$ edges. Hence, a vertex $v \in V$ contains at most $\min\{|\delta(v)|, b_v \cdot (\log_{1+\varepsilon}(W/\varepsilon) + 1)\}$ elements of S at the end of the algorithm.

Then let $U \subseteq V$ be the set of *saturated* vertices of V by M_{max} , i.e. the set of vertices $v \in V$ such that $|\delta(v) \cap M_{max}| = \min\{|\delta(v)|, b_v\}$. By construction, U is a vertex cover and $\sum_{v \in U} \min\{|\delta(v)|, b_v\} \leq 2|M_{max}|$. As all edges of S have at least one endpoint in U , we get:

$$\begin{aligned} |S| &\leq \sum_{v \in U} |\delta(v) \cap S| \leq \sum_{v \in U} \min\{|\delta(v)|, b_v \cdot (\log_{1+\varepsilon}(W/\varepsilon) + 1)\} \\ &\leq \sum_{v \in U} \min\{|\delta(v)|, b_v\} \cdot (\log_{1+\varepsilon}(W/\varepsilon) + 1) \leq 2(\log_{1+\varepsilon}(W/\varepsilon) + 1) \cdot |M_{max}|, \end{aligned}$$

so the memory consumption of the algorithm is $O(\log_{1+\varepsilon}(W/\varepsilon) \cdot |M_{max}|)$ \blacktriangleleft

Modifying an idea from Ghaffari and Wajc [10] we can further improve the memory consumption of the algorithm, especially if W is not bounded. For that, set $\beta = 1 + \frac{2 \log(1/\varepsilon)}{\log(1+\varepsilon)} = 1 + \log_{1+\varepsilon}(1/\varepsilon^2)$ in Algorithm 7 as the maximum size of a queue, and set $d = 1$ (so that we now consider the whole code).

Consider an endpoint u of the newly-inserted edge e . If the queue $Q_{u, q_u(e)}$ becomes too long (more than β elements), it means the gain $g(e')$ of the $\beta + 1$ -th element from the top of the queue (we will call that element e') is very small compared to $g(e)$, so we then can “potentially” remove e' from S and from the queues without hurting too much $g(S)$. In the code, we will mark this edge e' as *erasable*, so that when e' will no longer be on top of any queue, it will be removed from S and all the queues it appears in. To be able to do these eviction operations, the queues have to be implemented with doubly linked lists.

If an edge $e = \{u, v\}$ is marked as erasable by Algorithm 7 ($d = 1$) because an edge $e' = \{u, v'\}$ is added to S , then we say that e' *evicted* e (and that e was *evicted* by e').

► **Lemma 50.** *If $e = \{u, v\}$ is evicted by $e' = \{u, v'\}$, then $g(e') \geq g(e)/\varepsilon$.*

Proof. We have $g(e') \geq \varepsilon(w_u^*(e') + w_{v'}^*(e')) \geq \varepsilon w_u^*(e') \geq \varepsilon(1 + \varepsilon)^{\beta-1}g(e) \geq g(e)/\varepsilon$ because after $e = e_t$ is added to $Q_{u, q_u(e)}$ we have $w_u(Q_{u, q_u(e)}^{(t)}) \geq g(e)$ and each time an element is added to $Q_{u, q_u(e)}$ the value $w_u(Q_{u, q_u(e)})$ is multiplied at least by $(1 + \varepsilon)$. \blacktriangleleft

► **Theorem 51.** *For $\varepsilon \leq \frac{1}{4}$, Algorithm 7 with $d = 1$ combined with Algorithm 2 gives a $2 + \varepsilon$ approximation algorithm by using $O(\log_{1+\varepsilon}(1/\varepsilon) \cdot |M_{max}| + \sum_{v \in V} b_v)$ variables.*

Proof. For an element e that was not evicted from S in Algorithm 7, denote by \mathcal{E}_e the elements that were evicted by e directly or indirectly (in a chain of evictions). This set \mathcal{E}_e contains at most 2 elements that were directly evicted when e was inserted in S , and their

associated gain is at most $\varepsilon g(e)$ for each, and at most 4 elements indirectly evicted by e when these 2 evicted elements were inserted in S , and their associated gain is at most equal to $\varepsilon^2 g(e)$ for each, and so on. Then, as $\varepsilon \leq \frac{1}{4}$,

$$\sum_{e' \in \mathcal{E}_e} g(e') \leq \sum_{i=1}^{\infty} (2\varepsilon)^i g(e) \leq 2\varepsilon g(e) \sum_{i=0}^{\infty} (1/2)^i = 4\varepsilon g(e)$$

Therefore, if S_0 denotes the set S obtained by Algorithm 7 when $d = 0$ and S_1 denotes the set S obtained by Algorithm 7 when $d = 1$, we get:

$$g(S_0) - g(S_1) \leq 4\varepsilon g(S_1)$$

because the elements inserted in S are exactly the same for the two algorithms, the only difference being that some elements are missing in S_1 (but these elements were removed when they no longer had any influence on the values of w^* and thereby no influence on the choice of the elements inserted in S afterwards). We have then:

$$w(M^{opt}) \leq 2(1 + \varepsilon)g(S_0) \leq 2(1 + \varepsilon)(1 + 4\varepsilon)g(S_1) \leq 2(1 + 6\varepsilon)g(S_1)$$

and Algorithm 2 will provide a $2(1 + 6\varepsilon)$ approximation of the optimal b -matching. In fact, the analysis of Algorithm 2 with S_1 as input is almost the same as in the proof of Lemma 8, the only difference being that now the weight of an element is no longer necessarily equal to the sum of the gains of elements below it but can be higher (which is not an issue).

Regarding the memory consumption of the algorithm, one can notice that, using the same notation U for the set of the elements saturated by M_{max} as previously, and because there are only up to $\sum_{v \in V} b_v$ elements on top of the queues that cannot be deleted (and thus these edges are the ones making some queues in U exceed the limit β), we obtain:

$$|S| \leq \sum_{v \in V} b_v + \sum_{v \in U} \beta \cdot \min\{|\delta(v)|, b_v\} \leq \sum_{v \in V} b_v + 2\beta \cdot |M_{max}|,$$

and therefore the algorithm uses $O(\sum_{v \in V} b_v + \log_{1+\varepsilon}(1/\varepsilon) \cdot |M_{max}|)$ variables. \blacktriangleleft

► **Remark 52.** Ideas presented here for the maximum weight b -matching can be easily extended to submodular function maximization, for hypergraphs under a matroid constraint or under matchoid constraints, namely for the algorithms presented in Sections 3, 4, and 5.

B Example of Different Behavior Compared to [12]

Here is an example to show the difference of behavior between our algorithm and the one proposed in [12]. Consider a set of four vertices $V = \{v_1, v_2, v_3, v_4\}$. We set $b_{v_1} = 2$ and $b_{v_i} = 1$ for $2 \leq i \leq 4$. Let $E = \{e_1, e_2, e_3\}$ with $e_1 = \{v_1, v_2\}$ and $w(e_1) = 2$, $e_2 = \{v_1, v_3\}$ and $w(e_2) = 7$, $e_3 = \{v_1, v_4\}$ and $w(e_3) = 4$. Using only one dual variable for each vertex, the algorithm of Levin and Wajc [12] takes e_1 and e_2 but discards e_3 because the dual variable ϕ_{v_1} is equal to 4 when e_3 is processed. On the other hand, our algorithm, when processing e_3 , compares $w(e_3)$ with $w_{v_1}(e_1) = 2$. Therefore, e_3 is added into S and our algorithm provides a b -matching of weight 11 instead of 9.