Streaming Submodular Matching Meets the Primal-Dual Method

Roie Levin[*1] and David Wajc[†2]

[1]Carnegie Mellon University
[2]Stanford University

Abstract

We study streaming submodular maximization subject to matching/b-matching constraints (MSM/MSʙM), and present improved upper and lower bounds for these problems. On the upper bounds front, we give primal-dual algorithms achieving the following approximation ratios.

- $3 + 2\sqrt{2} \approx 5.828$ for monotone MSM, improving the previous best ratio of 7.75.
- $4 + 3\sqrt{2} \approx 7.464$ for non-monotone MSM, improving the previous best ratio of 9.899.
- $3 + \epsilon$ for maximum weight b-matching, improving the previous best ratio of $4 + \epsilon$.

On the lower bounds front, we improve on the previous best lower bound of $\frac{e}{e-1} \approx 1.582$ for MSM, and show ETH-based lower bounds of ≈ 1.914 for polytime monotone MSM streaming algorithms.

Our most substantial contributions are our algorithmic techniques. We show that the (randomized) primal-dual method, which originated in the study of maximum *weight* matching (MWM), is also useful in the context of MSM. To our knowledge, this is the first use of primal-dual based analysis for streaming submodular optimization. We also show how to reinterpret previous algorithms for MSM in our framework; hence, we hope our work is a step towards unifying old and new techniques for streaming submodular maximization, and that it paves the way for further new results.

1 Introduction

In this paper we study streaming maximum submodular matching (MSM) and related problems. That is, we study the problem of computing high-value matchings where the value is determined by a submodular function, i.e. a function which captures *diminishing returns*.

Submodular function maximization has a long history. For example, it has been known since the 70s that the greedy algorithm yields an $e/(e-1) \approx 1.582$ approximation for monotone submodular maximization subject to a cardinality constraint [60]. This is optimal among polytime algorithms with value oracle access [59], or assuming standard complexity-theoretic conjectures [21, 25, 56]. The same problem for *non-monotone* submodular functions is harder; it is hard to approximate to within a 2.037 factor [62]. Much work has been dedicated to improving the achievable approximation [13, 14, 22, 28, 38, 52, 62, 68]; the best currently stands at 2.597 [13].

Closer to our work is the study of submodular maximization subject to *matching* constraints. For this problem, the greedy algorithm has long been known to be 3-approximate for monotone functions [31]. Improved approximations have since been obtained [29, 38, 51, 53], with the current best being $(2 + \epsilon)$ and $(4 + \epsilon)$ for monotone and non-monotone MSM respectively [29]. The papers above studied rich families of constraints (e.g. matroid intersection, matchoids, exchange systems), some of which were motivated explicitly by matching constraints (see [29]). Beyond theoretical interest, the MSM problem also has great practical appeal, since many natural objectives exhibit diminishing returns behavior. Applications across different fields include: machine translation [55], Internet advertising [20, 49], combinatorial auctions more broadly [15, 54, 67], and any matching problem where the goal is a submodular notion of utility such as diversity [2, 3].

The proliferation of big-data applications such as those mentioned above has spurred a surge of interest in algorithms for the regime where the input is too large to even store in local memory. To this end, it is common to formulate problems in the *streaming* model. Here the input is presented element-by-element to an algorithm that is restricted to use $\tilde{O}(S)$ memory, where S is the maximum size of any feasible solution. We study MSM in this model.

For our problem when the objective is *linear*, a line of work [18, 23, 26, 33, 57, 63] has shown that a $(2 + \epsilon)$-approximation is possible in the streaming model [33, 63]. Meanwhile, for submodular objectives under *cardinality* constraints (which are a special case of MSM in complete bipartite graphs), a separate line of work [4, 7, 17, 27, 48, 58] has culminated in the same $(2 + \epsilon)$ approximation ratio, for both monotone and non-monotone functions (the latter taking exponential time, as is to be expected from the lower bound of [62]); moreover, this $(2 + \epsilon)$ bound was recently proven to be tight [4, 30, 61]. On the other hand, for fully general MSM, the gap between known upper and lower bounds remain frustratingly large. Chakrabarti and Kale [16] gave a 7.75-approximate algorithm for MSM with monotone functions. For non-monotone functions, Chekuri et al. [17] gave a $(12 + \epsilon)$-approximate algorithm, later improved by Feldman et al. [27] to $5 + 2\sqrt{6} \approx 9.899$. The only known lower bound for monotone MSM is $\frac{e}{e-1} \approx 1.582$ for streaming or polytime algorithms, implied respectively by [47] and [25, 59]. For non-monotone functions, [62] implies a hardness of 2.037. Closing these gaps, especially from the algorithmic side, seems to require new ideas.

1.1 Our Contributions

We present a number of improved results for streaming maximum submodular matching (MSM) and related problems.

Our first result is an improvement on the 7.75 approximation of [16] for monotone MSM.

Theorem 1.1. *There exists a deterministic quasilinear-time streaming* MSM *algorithm for monotone functions which is* $3 + 2\sqrt{2} \approx 5.828$ *approximate.*

Our algorithm extends in various ways: First, it yields the same approximation ratio for submodular b-*matchings*, where each node v can be matched b_v times, improving on the previous best 8-approximations [17, 27]. For the special case of linear functions (MWM), our algorithm—with appropriate parameters—recovers the $(2 + \epsilon)$-approximate algorithm of [63]. For weighted b-matching (MWBM), a slight modification of our algorithm yields a $(3 + \epsilon)$-approximate algorithm, improving on the previous best $(4 + \epsilon)$-approximation [18].[1]

Next, we improve on the $5 + 2\sqrt{6} \approx 9.899$ approximation of [27] for non-monotone MSM.

Theorem 1.2. *There exists a randomized linear-time streaming* MSM *algorithm for non-monotone functions which is* $4 + 2\sqrt{3} \approx 7.464$ *approximate.*

Our non-monotone MSM algorithm's approximation ratio is better than the previous state-of-the-art 7.75-approximate *monotone* MSM algorithm [16]. Moreover, when applied to monotone functions, the algorithm of Theorem 1.2 yields the same approximation ratio as the deterministic algorithm of Theorem 1.1.

We turn to proving hardness for monotone MSM. As stated before, the previous best lower bounds for this problem were $\frac{e}{e-1} \approx 1.582$. These lower bounds applied to either space-bounded [47] or time-bounded algorithms [25, 59]. We show that the problem becomes harder for algorithms which are both space bounded and time bounded. This answers an open problem posed in the Bertinoro Workshop on Sublinear Algorithms 2014 [1], at least for time bounded algorithms.[2]

Theorem 1.3. *No polytime streaming* MSM *algorithm for monotone functions is better than* 1.914 *approximate.*

Finally, to demonstrate that our techniques have the potential for wider applicability, we also use them to provide an alternative and unified proof of the results of Chakrabarti and Kale [16] and Feldman et al. [27] for MSM, in Appendix A.

1.2 Our Techniques and Overview

Our starting point is the breakthrough result of Paz and Schwartzman [63] for a special case of our problem—maximum weight matching (MWM). They gave a $(2 + \epsilon)$-approximate streaming

---

[1]Independently, Mohsen Ghaffari (private communication) obtained the same $(3 + \epsilon)$-approximate for MWBM.
[2]We note briefly that such a bound does not follow from space lower bounds for cardinality constrained submodular maximization [30, 61] (a special case of our setting, with a complete bipartite graph on n and k nodes), since a bound for that problem cannot be superlinear in n.

algorithm by extending the local-ratio technique [8]. Subsequently, Ghaffari and Wajc [33] simplified and slightly improved the analysis of [63], by re-interpreting their algorithm in terms of the primal-dual method.[3] The primal-dual method is ubiquitous in the context of approximating linear objectives. In this paper, we show that this method is also useful in the context of streaming submodular optimization, where to the best of our knowledge, it has not yet been used. For our primal-dual analysis, we rely on the *concave-closure extension* for submodular functions which has a "configuration LP"-like formulation. In particular, using this extension, we find that a natural generalization of the MWM algorithm of [63] (described in Section 3) yields improved bounds for monotone MSM and its generalization to b-matchings. Our primal-dual analysis is robust in the sense that it allows for extensions and generalizations, as we now outline.

Our approach in a nutshell (Sections 3+4). Our approach is to keep monotone dual solutions (initially zero), and whenever an edge arrives, discard it if its dual constraint is already satisfied. Edges whose dual constraint is not satisfied are added to a stack S, and relevant dual variables are increased, so as to satisfy their dual constraint. Finally, we unwind the stack S, constructing a matching M greedily. The intuition here is that the latter edge in the stack incident on a common edge have higher marginal gain than earlier such edges in the stack. More formally, we show that this matching M has value at least some constant times the dual objective cost. Weak LP duality and the choice of LP imply that $f(M) \geq \frac{1}{\alpha} \cdot f(S \cup OPT)$ for some $\alpha > 1$, which implies our algorithm is α-approximate for monotone MSM.

Extension 1 (Section 5). Extending our approach, which gives $f(M) \geq \frac{1}{\alpha} \cdot f(S \cup OPT)$, to non-monotone functions f seems challenging, since for such functions $f(S \cup OPT)$ can be arbitrarily smaller than $f(OPT)$. To overcome this challenge, we note that our dual updates over-satisfy dual constraints of edges in S. We can therefore afford to randomly discard edges whose dual is not satisfied on arrival (and not add them to S), resulting in these edges' dual constraints holding *in expectation*. This allows us to argue, via a generalization of the randomized primal-dual method of Devanur et al. [19] (on which we elaborate in Section 2), that $\mathbb{E}[f(M)] \geq \frac{1}{\alpha} \cdot \mathbb{E}[f(S \cup OPT)]$. As S contains each element with probability at most some q, a classic lemma of [14] allows us to show that $\mathbb{E}[f(S \cup OPT)] \geq (1 - q) \cdot f(OPT)$, from which we get our results for non-monotone MSM.[4] Given the wide success of the randomized-primal dual method of [19] in recent years [24, 35, 40, 41, 42, 43, 44, 45, 65], we believe that our extension of this method in the context of submodular optimization will likely find other applications.

Extension 2 (Section 6). For maximum weight b-*matching* (MWBM), the dual updates when adding an edge to the stack are not high enough to satisfy this edge's dual constraint. However, since we do cover each edge *outside* the stack S, weak duality implies that a maximum-weight b-matching M in the stack S has value at least as high as $f(M) \geq \frac{1}{2+\epsilon} \cdot f(OPT \setminus S)$, and trivially at least as high as $f(M) \geq f(OPT \cap S)$. Combining these lower bounds on $f(M)$ imply our improved $(3 + \epsilon)$ approximation ratio for MWBM. This general approach seems fairly general, and could find uses for other sub-additive objectives subject to downward-closed constraints.

---

[3]An equivalence between the local-ratio and primal-dual methods was established in [9], though it does not capture the extension of the local ratio method in [63].

[4]Incidentally, for monotone functions, for which $E[f(M)] \geq \frac{1}{\alpha} \cdot E[f(S \cup OPT)] \geq \frac{1}{\alpha} \cdot f(OPT)$, this algorithm is α approximate. This is somewhat surprising, as this algorithm runs an α-approximate monotone algorithm (and this analysis is tight, by Appendix B) on a random q-fraction of the input, suggesting an α/q approximation. Nonetheless, we show that for q not too small in terms of α, we retain the same approximation ratio even after this sub-sampling.

Unifying Prior Work (Appendix A). To demonstrate the usefulness of our primal-dual analysis, we also show that this (randomized) primal-dual approach gives an alternative, unified way to analyze the MSM algorithms of [16, 27].

Lower bound (Section 7). Our lower bound instance makes use of two sources of hardness: computational hardness under ETH ([21, 25]) and information-theoretic hardness resulting form the algorithm not knowing the contents or order of the stream in advance ([34]). In particular, our proof embeds a submodular optimization problem (specifically, set cover) in parts of the linear instance of [34], and hence exploits the submodularity in the MSM objective. Interestingly, our lower bound of 1.914 is higher than any convex combination of the previous hardness results we make use of, both of which imply a lower bound no higher than $e/(e-1)$.

2 Preliminaries

A set function $f : 2^N \to \mathbb{R}$ is *submodular* if the marginal gains of adding elements to sets, denoted by $f_S(e) := f(S \cup \{e\}) - f(S)$, satisfy $f_S(e) \geq f_T(e)$ for $e \notin T$ and $S \subseteq T \subseteq N$. We say f is *monotone* if $f(S) \leq f(T)$ for all $S \subseteq T \subseteq N$. Throughout this paper we assume oracle access to the submodular function. The maximum submodular matching (MSM) problem is defined by a non-negative submodular function $f : 2^E \to \mathbb{R}_{\geq 0}$, where E is the edge-set of some n-node graph $G = (V, E)$, and feasible sets are matchings in G. The more general maximum submodular b-matching (MSBM) problem has as feasible sets subgraphs in which the degree of each vertex v does not exceed b_v, for some input vector $\vec{b}$. Our objective is to design algorithms with low approximation ratio $\alpha \geq 1$, that is algorithms producing solutions M such that $\mathbb{E}[f(M)] \geq \frac{1}{\alpha} \cdot f(OPT)$ for the smallest possible value of α, where OPT is an optimal solution.

For *streaming* MSM, edges of E are presented one at a time, and we are tasked with computing a matching in G at the end of the stream, using little memory. In particular, we give algorithms using space within the output-sensitive bound of $\tilde{O}(M_{\max})$, where $M_{\max}$ is the maximum size of a b-matching (compared to the entire graph size, which can be $\Omega(n^2)$). We note that for matchings, this is a finer requirement than the usual $\tilde{O}(n)$ space bound, since $M_{\max} = O(n)$. We note also that for monotone objectives, an optimal b-matching is maximal, and so $M_{\max} \leq 2|OPT|$. Consequently, such space is a trivial lower bound for outputting constant-approximate solutions.

On a technical note, we will only allow the algorithm to query the value oracle for f on subsets currently stored in memory. We assume the range of f is polynomially bounded (it is common to assume bounds on this quantity, e.g., [37, 70]). More precisely, we assume that $f_{\max}/f_{\min} = n^{O(1)}$ (these max and min marginals are defined below). This implies in particular that we can store a value $f_S(e)$ using $O(\log n)$ bits.

Useful Notation Throughout this paper we will rely on the following notation. First, we denote by $e^{(1)}, e^{(2)}, \ldots$, the edges in the stream, in order. For edges $e = e^{(i)}, e' = e^{(j)}$, we write $e < e'$ if and only if $i < j$, i.e., if e arrived before e'. Similarly to [17, 27], we will also use $f(e : S) := f_{S \cap \{e' < e\}}(e)$ as shorthand for the marginal gain from adding e to the set of elements which arrived before e in S. One simple yet useful property of this notation is that $\sum_{e \in S} f(e : S) = f(S)$ ([17, Lemma 1].) Other properties of this notation we will make use of, both easy consequences of submodularity, are $f(e : S) \leq f_S(e)$, as well as monotonicity of $f(e : S)$ in S, i.e., $f(e : A) \geq f(e : B)$ for $A \subseteq B$. Define the max and min marginals of f to be $f_{\max} := \max\{f(e) \mid e \in E\}$ and

$f_{\min} := \min\{f(e \mid S) \mid e \in E, S \subseteq E, \ f(e \mid S) \neq 0\}$. Finally, for any edge e, we denote by $N(e) := \{e' \mid e' \cap e \neq \emptyset\}$ the set of neighboring edges of e.

2.1 The Primal-Dual Method in Our Setting

As discussed in Section 1.2, the main workhorse of our algorithms is the primal-dual method. In this method, we consider some linear program (LP) relaxation, and its dual LP. We then design an algorithm which computes a (primal) solution of value P, and a feasible solution of value D, and show that $P \geq \frac{1}{\alpha} \cdot D$, which implies an approximation ratio of α, by weak duality, since

$$P \geq \frac{1}{\alpha} \cdot D \geq \frac{1}{\alpha} \cdot f(OPT).$$

For linear objectives, the first step of the primal-dual method—obtaining an LP relaxation—is often direct: write some integer linear program for the problem and drop the integrality constraints. For submodular objective functions, which are only naturally defined over vertices of the hypercube, $\vec{x} \in \{0,1\}^E$, and are not defined over fractional points $\vec{x} \in [0,1]^E \setminus \{0,1\}^E$, the first step of defining a relaxation usually requires extending f to real vectors. For this, we use the *concave closure* (see e.g.[66] for a survey of its history and further properties).

Definition 2.1. *The* concave closure $f^+ : [0,1]^E \to \mathbb{R}$ *of a set function* $f : 2^E \to \mathbb{R}$ *is given by*

$$f^+(\vec{x}) := \max \left\{ \sum_{T \subseteq E} \alpha_T \cdot f(T) \ \middle| \ \sum_{T \subseteq E} \alpha_T = 1, \ \alpha_T \geq 0 \ \forall T \subseteq E, \sum_{T \ni e} \alpha_T = x_e \ \forall e \in E \right\}.$$

In words, the concave closure is the maximum expected f-value of a random subset $T \subseteq E$, where the maximum is taken over all distributions matching the marginal probabilities given by $\vec{x}$. This is indeed an extension of set functions (and in particular submodular functions) to real-valued vectors, as this distribution must be deterministic for all $\vec{x} \in \{0,1\}^E$. Consequently, for any set $P \subseteq [0,1]^E$ containing the characteristic vector $\vec{x}_{OPT}$ of an optimal solution OPT, we have that $\max_{\vec{x}^* \in P} f^+(\vec{x}) \geq f^+(\vec{x}_{OPT}) = f(OPT)$.

Now, to define an LP relaxation for submodular maximization of some function g subject to some linear constraints $A\vec{x} \leq \vec{c}$, we simply consider $\max\left\{ g^+(\vec{x}) \mid A\vec{x} \leq \vec{c} \right\}$. For MSBM, we obtain the primal and dual programs given in Figure 1.

Primal (P)		Dual (D)	
max	$\sum_{T \subseteq E} \alpha_T \cdot g(T)$	min	$\mu + \sum_{v \in V} b_v \cdot \phi_v$
subject to		subject to	
$\forall T \subseteq E:$	$\sum_{T \ni e} \alpha_T = x_e$	$\forall T \subseteq E:$	$\mu + \sum_{e \in T} \lambda_e \geq g(T)$
	$\sum_{T \subseteq E} \alpha_T = 1$	$\forall e \in E:$	$\sum_{v \in e} \phi_v \geq \lambda_e$
$\forall v \in V:$	$\sum_{e \ni v} x_e \leq b_v$		
$\forall e \in E, T \subseteq E:$	$x_e, \alpha_T \geq 0$	$\forall v \in V:$	$\phi_v \geq 0$

Figure 1: The LP relaxation of the MSBM problem and its dual

2.2 Non-Monotone MSM: Extending the Randomized Primal-Dual Method

To go from monotone to non-monotone function maximization, we make use of our dual updates resulting in dual solutions which over-satisfy (some) dual constraints. This allows us to randomly sub-sample edges with probability q when deciding whether to insert them into S, and still have a dual solution which is feasible *in expectation* over the choice of S. This is akin to the randomized primal-dual method of Devanur et al. [19], who introduced this technique in the context of maximum *cardinality* and *weighted* matching. However, unlike in [19] (and subsequent work [24, 35, 40, 41, 42, 43, 44, 45, 65]), for our problem the LP is not fixed. Specifically, we consider a different submodular function in our LP based on S, denoted by $g^S(T) := f(T \cup S)$. This results in *random* primal and dual LPs, depending on the random set S. We show that our (randomized) dual solution is feasible for the obtained (randomized) dual LP in expectation over S. Consequently, our expected solution's value is at least as high as some multiple of an expected solution to the dual LP, implying

$$\mathbb{E}_S[f(M)] \geq \frac{1}{\alpha} \cdot \mathbb{E}_S[D] \geq \frac{1}{\alpha} \cdot \mathbb{E}_S[f(S \cup OPT)]. \tag{1}$$

Equation (1) retrieves our bound for monotone functions, for which $\mathbb{E}_S[f(S \cup OPT)] \geq f(OPT)$. To obtain bounds for non-monotone functions, we show that $\mathbb{E}_S[f(S \cup OPT)] \geq (1-q) \cdot f(OPT)$, by relying on the following lemma, due to Buchbinder et al. [14, Lemma 2.2].

Lemma 2.2 ([14]). *Let $h : 2^N \to R_{\geq 0}$ be a non-negative submodular function, and let B be a random subset of N containing every element of N with probability at most q (not necessarily independently), then $\mathbb{E}[h(B)] \geq (1-q) \cdot h(\emptyset)$.*

3 Our Basic Algorithm

In this section we describe our monotone submodular b-matching algorithm, which we will use with slight modifications and different parameter choices in coming sections.

The algorithm maintains a stack of edges S, initially empty, as well as vertex potentials $\vec{\phi} \in \mathbb{R}^{|V|}$. To avoid storing zero-valued potentials (increasing space to $\tilde{O}(n)$, rather than $\tilde{O}(M_{\max})$), we implicitly represent these by keeping a mapping from vertices with non-zero potentials to their potentials, by using e.g., a hash table or balanced binary search tree. When an edge e arrives, we compare the marginal value of this arriving edge with respect to the stack to the sum of vertex potentials of the edge's endpoints times a slack parameter C. If $C \cdot \sum_{v \in e} \phi_v$ is larger, we continue to the next edge. Otherwise, with probability q we add the edge to the stack and increment the endpoint vertex potentials. At the end of the stream, we construct a b-matching greedily by unwinding the stack *in reverse order*. The pseudocode is given in Algorithm 1.

Algorithm 1 clearly outputs a feasible b-matching. In subsequent sections we analyze this algorithm for various values of the parameters C and q. Before doing so, we note that this algorithm when run with $C = 1 + \Omega(1)$ is indeed a streaming algorithm, i.e., its space usage is as follows.

Lemma 3.1. *For any constant $\epsilon > 0$, Algorithm 1 run with $C = 1 + \epsilon$ uses $\tilde{O}(M_{\max})$ space.*

The proof relies on each vertex v only having $\tilde{O}(b_v)$ edges in the stack, since every edge incident on vertex v inserted to the stack increases ϕ_v by a multiplicative factor of $(C-1)/b_v$, while the minimum and maximum non-zero values which ϕ_v can take are polynomially bounded in each other, due to f being polynomially bounded. Since this bound holds in particular for vertices of a

Algorithm 1 The MSbM Algorithm

___

Initialization

1: $S \leftarrow$ emptystack
2: $\forall v \in V : \phi_v^{(0)} \leftarrow 0$ (implicitly)

Loop

3: **for** $t \in \{1, \dots, |E|\}$ **do**
4:      $e \leftarrow e^{(t)}$
5:      $\forall v \in V : \phi_v^{(t)} \leftarrow \phi_v^{(t-1)}$
6:      **if** $C \cdot \sum_{v \in e} \phi_v^{(t-1)} \geq f(e : S)$ **then**
7:          **continue**   ▷ skip edge e
8:      **else**
9:          **with** probability q **do**
10:             $S.push(e)$
11:             **for** $v \in e$ **do**
12:                 $w_{ev} \leftarrow \frac{f(e:S) - \sum_{v \in e} \phi_v^{(t-1)}}{b_v}$
13:             **for** $v \in e$ **do**
14:                 $\phi_v^{(t)} \leftarrow \phi_v^{(t-1)} + w_{ev}$

Post-Processing

15: $M \leftarrow \emptyset$
16: **while** $S \neq$ emptystack **do**
17:     $e \leftarrow S.pop()$
18:     **if** $|M \cap N(e)| < b_v$ for all $v \in e$ **then**
19:         $M \leftarrow M \cup \{e\}$
20: **return** M

___

minimum vertex cover, the bound follows due to minimum vertex covers having size at most twice that of maximal b-matchings in the same graph. See Appendix C for the complete proof.

We further note that Algorithm 1 runs in time (quasi)linear in $|E|$, times evaluation time of f.

Lemma 3.2. *A randomized (deterministic) implementation of Algorithm 1 requires $O(1)$ $(O(\log n))$ operations and $O(1)$ function evaluations per arrival, followed by $\tilde{O}(M_{\max})$ time post-processing. Using $\tilde{O}(n)$ space, the deterministic time per arrival can be decreased to $O(1)$, by storing all ϕ_v.*

4   Monotone MSbM

In this section we will consider a *deterministic* instantiation of Algorithm 1 (specifically, we will set $q = 1$) in the context of monotone submodular b-matching.

To argue about the approximation ratio, we will fit a dual solution to this algorithm. Define the auxiliary submodular functions $g^S : 2^E \to \mathbb{R}^+$ to be $g^S(T) := f(S \cup T)$. We will work with the dual LP (D) for the function g^S, and consider the following dual solution.

$$\mu := f(S) = g^S(\emptyset),$$
$$\phi_v := C \cdot \phi_v^{(|E|)}$$
$$\lambda_e := \begin{cases} f(e : S) & e \notin S \\ 0 & e \in S. \end{cases}$$

We start by showing that the above is indeed dual feasible.

Lemma 4.1. *The dual solution $(\mu, \vec{\phi}, \vec{\lambda})$ is feasible for the LP (D) with function g^S.*

7

Proof. To see that the first set of constraints is satisfied, note that by submodularity of f

$$\sum_{e \in T} \lambda_e = \sum_{e \in T \setminus S} f(e:S) \geq \sum_{e \in T \setminus S} f_S(e) \geq f_S(T \setminus S) = f(S \cup T) - f(S) = g^S(T) - \mu.$$

For the second set of constraints, note that an edge $e = e^{(t)}$ is not added to the stack if and only if the check at Line 6 fails. Therefore, since $\phi_v^{(t)}$ values increase monotonically with t, we have

$$\sum_{v \in e} \phi_v = C \cdot \sum_{v \in e} \phi_v^{(|E|)} \geq C \cdot \sum_{v \in e} \phi_v^{(t-1)} \geq f(e:S) = \lambda_e. \qquad \square$$

It remains to relate the value of the solution M to the cost of this dual. We first prove an auxiliary relationship that will be useful:

Lemma 4.2. *The b-matching M output by Algorithm 1 satisfies*

$$f(M) \geq \frac{1}{2} \cdot \sum_{e \in S} \sum_{v \in e} b_v \cdot w_{ev}.$$

Proof. We first note that for any edge $e = e^{(t)}$ and $v \in e$, since $\phi_v^{(t-1)} = \sum_{e' \ni v, e' < e} w_e$, we have that

$$f(e:S) = b_v \cdot w_{ev} + \sum_{u \in e} \phi_u^{(t-1)} \geq b_v \cdot w_{ev} + \phi_v^{(t-1)} = b_v \cdot w_{ev} + \sum_{\substack{e' \ni v \\ e' < e}} w_{e'v}.$$

Combined with submodularity of f, the above yields the following lower bound on $f(M)$,

$$f(M) = \sum_{e \in M} f(e:M) \geq \sum_{e \in M} f(e:S) \geq \sum_{e \in M} \sum_{v \in e} \left(b_v \cdot w_{ev} + \sum_{\substack{e' \ni v \\ e' < e}} w_{e'v} \right).$$

On the other hand, the greedy manner in which we construct M implies that any edge $e' \in S \setminus M$ must have at least one endpoint v with b_v edges $e > e'$ in M. Consequently, the term $w_{e'v}$ for such e and v is summed b_v times in the above lower bound for $f(M)$. On the other hand, $b_v \cdot w_{ev} = b_u \cdot w_{eu}$ for $e = (u, v)$, by definition. From the above we obtain our desired inequality.

$$f(M) \geq \sum_{e \in M} \sum_{v \in e} b_v \cdot w_{ev} + \frac{1}{2} \cdot \sum_{e \in S \setminus M} \sum_{v \in e} b_v \cdot w_{ev} \geq \frac{1}{2} \cdot \sum_{e \in S} \sum_{v \in e} b_v \cdot w_{ev}. \qquad \square$$

We can now bound the two terms in the dual objective separately with respect to the primal, using the following two corollaries of Lemma 4.2.

Lemma 4.3. *The b-matching M output by Algorithm 1 satisfies $f(M) \geq \frac{1}{2C} \sum_{v \in V} b_v \cdot \phi_v$.*

Proof. Since $\phi_v = C \cdot \phi_v^{(|E|)}$, and $w_{ev} = \phi_v^{(t)} - \phi_v^{(t-1)}$ for all $v \in e = e^{(t)}$, Lemma 4.2 implies that

$$f(M) \geq \frac{1}{2} \cdot \sum_{e \in S} \sum_{v \in e} b_v \cdot w_{ev} = \frac{1}{2} \cdot \sum_{v \in V} \sum_{t=1}^{|E|} b_v \cdot \left(\phi_v^{(t)} - \phi_v^{(t-1)} \right) = \frac{1}{2} \cdot \sum_{v \in V} b_v \cdot \phi_v^{(|E|)} = \frac{1}{2C} \cdot \sum_{v \in V} b_v \cdot \phi_v. \quad \square$$

Lemma 4.4. *The b-matching M output by Algorithm 1 satisfies $f(M) \geq \left(1 - \frac{1}{C}\right) \mu.$*

Proof. We note that $w_e > 0$ for an edge $e = e^{(t)}$ if and only if $f(e : S) \geq C \cdot \sum_{v \in e} \phi_v^{(t-1)}$. Hence,

$$b_v \cdot w_{ev} = f(e : S) - \sum_{v \in e} \phi_v^{(t-1)} \geq \left(1 - \frac{1}{C}\right) \cdot f(e : S).$$

Combining the above with Lemma 4.2, and again recalling that for $e = (u, v)$, we have that $b_v \cdot w_{ev} = b_u \cdot w_{eu}$, by definition, we obtain the desired inequality.

$$f(M) \geq \frac{1}{2} \cdot \sum_{e \in S} \sum_{v \in e} b_v \cdot w_{ev} \geq \left(1 - \frac{1}{C}\right) \sum_{e \in S} f(e : S) = \left(1 - \frac{1}{C}\right) f(S). \qquad \square$$

Combining the above two corollaries and Lemma 4.1 with LP duality, we can now analyze the algorithm's approximation ratio.

Theorem 4.5. *Algorithm 1 run with $q = 1$ and C on a monotone MSBM instance outputs a b-matching M of value*

$$\left(2C + \frac{C}{C-1}\right) \cdot f(M) \geq f(OPT).$$

This is optimized by taking $C = 1 + \frac{1}{\sqrt{2}}$, which yields a $3 + 2\sqrt{2} \approx 5.828$ approximation.

Proof. By weak LP duality and Lemma 4.1, together with monotonicity of f, we have that

$$C \cdot \sum_v b_v \cdot \phi_v + \mu \geq \max_T g^S(T) = \max_T f(S \cup T) \geq f(S \cup OPT) \geq f(OPT).$$

Combining Lemma 4.3 and Lemma 4.4 and rearranging, we get the desired inequality,

$$\left(2C + \frac{C}{C-1}\right) \cdot f(M) \geq C \cdot \sum_v b_v \cdot \phi_v + \mu \geq f(OPT). \qquad \square$$

In Appendix B we show that our analysis of Algorithm 1 is tight.

We note that our analysis of this section required monotonicity, as we lower bounded $f(M)$ by (a multiple of) $f(S \cup OPT) \geq f(OPT)$, where the last step crucially relies on monotonicity. In the next section, we show how the use of randomness (namely, setting $q \neq 1$) allows us to obtain new results for *non-monotone* MSM.

5 Non-Monotone MSM

In this section we consider MSM (so, $b_v = 1$ for all v in this section), for *non-monotone* functions.

To extend our results to non-monotone MSM, we make use of the freedom to choose $q \notin \{0, 1\}$, resulting in a randomized algorithm. This will allow us to lower bound $\mathbb{E}_S[f(S \cup OPT)]$ in terms of $f(OPT)$. But first, we show that for appropriately chosen q, the output matching M has high value compared to $\mathbb{E}_S[f(S \cup OPT)]$. The analysis of this fact will follow the same outline of Section 4, relying on LP duality, but with a twist.

For our dual fitting, we use the same dual solution as in Section 4. However, this time this dual solution will only be feasible *in expectation*, in the following sense. Since we now have $q \notin \{0, 1\}$, Algorithm 1 is now a randomized algorithm, S is a random set, g^S is a random submodular function, and thus (D) is a random LP. Let $\mathbb{E}[(D)]$ denote this LP, which is (D) with the submodular function $g(T) := \mathbb{E}_S[g^S(T)]$. We now show that our dual solution's expectation is feasible for $\mathbb{E}[(D)]$.

Lemma 5.1. *For $q \in [1/(2C+1), 1/2]$, the expected dual solution $(\mathbb{E}[\mu], \mathbb{E}[\vec{\phi}], \mathbb{E}[\vec{\lambda}])$ is feasible for the expected LP $\mathbb{E}[(D)]$.*

Proof. The first set of constraints is satisfied for any realization of the randomness. Indeed, as in the proof of Lemma 4.1, for any realization of S, by submodularity of f, we have

$$\sum_{e \in T} \lambda_e = \sum_{e \in T \setminus S} f(e:S) \geq \sum_{e \in T \setminus S} f_S(e) \geq f_S(T \setminus S) = f(S \cup T) - f(S) = g^S(T) - \mu.$$

Consequently, taking expectation over S, we have that indeed, $\mathbb{E}_S[\mu] + \sum_{e \in T} \mathbb{E}_S[\lambda_e] \geq \mathbb{E}_S[g^S(T)]$. We now turn to proving the second set of constraints, which will only hold in expectation.

Fix an edge $e = e^{(t)}$, and define the event $A_e := [f(e:S) \leq C \cdot \sum_{v \in V} \phi_v^{(t-1)}]$. Then, by definition of A_e and monotonicity of $\phi_v^{(t)}$ in t, we have that

$$\mathbb{E}\left[\sum_{v \in e} \phi_v \,\middle|\, A_e\right] \geq \mathbb{E}\left[C \cdot \sum_{v \in e} \phi_v^{(t-1)} \,\middle|\, A_e\right] \geq \mathbb{E}[f(e:S) \mid A_e] = \mathbb{E}[\lambda_e \mid A_e]. \tag{2}$$

We now prove the same inequality holds when conditioning on the complement, $\overline{A_e}$.

Fix a realization of the randomness R for which $\overline{A_e}$ holds. Then, $e = e^{(t)}$ fails the test in Line 6, and so with probability q, we have $\sum_{v \in e} \phi_v^{(t)} = \sum_{v \in e}(\phi_v^{(t-1)} + w_e) = 2 \cdot f(e:S) - \sum_{v \in e} \phi_v^{(t-1)}$, and with probability $(1-q)$, we have $\sum_{v \in e} \phi_v^{(t)} = \sum_{v \in e} \phi_v^{(t-1)}$. Hence, in this case, as $q \leq \frac{1}{2}$, we have

$$\mathbb{E}\left[\sum_{v \in e} \phi_v^{(t)} \,\middle|\, R\right] = 2q \cdot f(e:S) + (1-2q) \cdot \sum_{v \in e} \phi_v^{(t-1)} \geq 2q \cdot f(e:S).$$

Now, since $\phi_v \geq C \cdot \phi^{(t)}$, and $q \geq 1/(2C+1)$ and since λ_e is set to $f(e:S)$ if e is not added to S (with probability $1-q$) and set to zero otherwise, the above implies that

$$\mathbb{E}\left[\sum_{v \in e} \phi_v \,\middle|\, R\right] \geq 2qC \cdot f(e:S) \geq (1-q) \cdot f(e:S) = \mathbb{E}[\lambda_e \mid R].$$

By the law of total expectation, taken over all $R \subseteq \overline{A_e}$, we have

$$\mathbb{E}\left[\sum_{v \in e} \phi_v \,\middle|\, \overline{A_e}\right] \geq \mathbb{E}\left[\lambda_e \mid \overline{A_e}\right]. \tag{3}$$

Combining inequalities (2) and (3) with the law of total expectation gives the desired inequality,

$$\mathbb{E}\left[\sum_{v \in e} \phi_v\right] \geq \mathbb{E}[\lambda_e]. \qquad \square$$

To bound the performance of this section's randomized variant of Algorithm 1, we can reuse corollaries 4.3 and 4.4, since these follow from Lemma 4.1, which holds for every realization of the random choices of the algorithm. We now use these corollaries, LP duality and Lemma 5.1, together with Lemma 2.2, to analyze this algorithm.

10

Theorem 5.2. *Algorithm 1 run with $q = 1/(2C + 1)$ and C on a non-monotone* MSM *instance outputs a matching M of value*

$$\left(\frac{4C^2 - 1}{2C - 2}\right) \cdot f(M) \geq f(OPT).$$

This is optimized by taking $C = 1 + \frac{\sqrt{3}}{2}$, resulting in an approximation ratio of $4 + 2\sqrt{3} \approx 7.464$. Moreover, the same algorithm is $2C + C/(C - 1)$ approximate for monotone MSM.

Proof. First, by Lemma 4.3 and Lemma 4.4, for every realization of the algorithm, we have

$$\left(2C + \frac{C}{C - 1}\right) \cdot f(M) \geq \sum_v \phi_v + \mu,$$

and thus this relationship holds in expectation as well.

$$\left(2C + \frac{C}{C - 1}\right) \cdot \mathbb{E}[f(M)] \geq \mathbb{E}\left[\sum_v \phi_v + \mu\right]. \tag{4}$$

On the other hand, by Lemma 5.1, the expected dual LP solution is feasible for $\mathbb{E}[(D)]$. Therefore, by weak LP duality, we have

$$\mathbb{E}\left[\sum_v \phi_v + \mu\right] \geq \max_T \mathbb{E}[g(T)] = \max_T \mathbb{E}[f(S \cup T)] \geq \mathbb{E}[f(S \cup OPT)]. \tag{5}$$

The result for monotone MSM follows from equations (4) and (5), together with monotonicity implying $\mathbb{E}[f(S \cup OPT)] \geq f(OPT)$.

For non-monotone MSM, let us define the additional auxiliary function $h : 2^E \to \mathbb{R}^+$, with $h(T) := f(OPT \cup T)$. Now note that by our sampling procedure, S is a random subset of E containing every edge with probability at most q. Hence, by Lemma 2.2, we have

$$\mathbb{E}[f(S \cup OPT)] = \mathbb{E}[h(S)] \geq (1 - q) \cdot h(\emptyset) = (1 - q) \cdot f(OPT). \tag{6}$$

Combining equations (4), (5) and (6), together with our choice of $q = 1/(2C + 1)$, the desired inequality follows by rearranging terms. $\square$

Having explored the use of Algorithm 1 for submodular matchings, we now turn to analyzing this algorithm in the context of streaming *linear* objectives.

6 Linear Objectives

In this section we address the use of Algorithm 1 to matching and b-matching with linear objectives, i.e., MWM and MWBM, using a deterministic variant, with $q = 1$.

For MWM, this algorithm with $C = 1 + \epsilon$ is essentially the algorithm of [63], and so it retrieves the state-of-the-art $(2 + \epsilon)$-approximation for this problem, previously analyzed in [33, 63]. We therefore focus on MWBM, for which a simple modification of Algorithm 1 yields a $3 + \epsilon$ approximation, improving upon the previous best $4 + \epsilon$ approximation due to [18].

The modification to Algorithm 1 which we consider is a natural one: instead of computing M greedily, we simply compute an optimal MWBM M in the subgraph induced by S, using a polytime linear-space offline algorithm (e.g., [5, 32]). Trivially, the b-matching M has weight at least

$$w(M) \geq w(OPT \cap S). \tag{7}$$

Moreover, this b-matching has weight no lower than the greedily-constructed b-matching of lines 15-20. We use LP duality to show that this modified algorithm with $C = 1 + \epsilon$ outputs a b-matching M of weight at least $w(M) \geq \frac{1}{2+\epsilon} \cdot w(OPT \setminus S)$.

Lemma 6.1. *Let M be a* MWBM *in the stack S obtained by running Algorithm 1 with $C = 1 + \epsilon/2$ and $q = 1$ until Line 15. Then, we have $w(M) \geq \frac{1}{2+\epsilon} \cdot w(OPT \setminus S)$.*

Proof. Consider the matching M' obtained by greedily unwinding the stack, as in Algorithm 1. Clearly, $w(M) \geq w(M')$. So, by Lemma 4.3, we have $w(M) \geq \frac{1}{2+\epsilon} \cdot \sum_{v \in V} \phi_v$, for $\phi_v = C \cdot \phi_v^{(|E|)}$. To relate $\sum_{v \in V} \phi_v$ to $w(OPT)$, we show that the dual solution $(0, \vec{\phi}, \vec{w})$ is dual feasible for the LP (D) with function w.

The first set of constraints are trivially satisfied, due to linearity of w, as $0 + \sum_{e \in T} w_e = w(T)$.

For the second set of constraints, note that an edge $e = e^{(t)}$ is not added to the stack if and only if the check at Line 6 fails. Therefore, since $\phi_v^{(t)}$ values increase monotonically with t, we have

$$\sum_{v \in e} \phi_v = C \cdot \sum_{v \in e} \phi_v^{(|E|)} \geq C \cdot \sum_{v \in e} \phi_v^{(t-1)} \geq f(e : S) = w_e.$$

Therefore, by weak LP duality, we have $w(M) \geq 0 + \frac{1}{2+\epsilon} \cdot \sum_{v \in V} \phi_v \geq \frac{1}{2+\epsilon} \cdot w(OPT)$. □

We are now ready to analyze the approximation ratio of this MWBM algorithm.

Theorem 6.2. *For any $\epsilon \geq 0$, Algorithm 1 run with $C = 1 + \epsilon/2$ and $q = 1$ until Line 15, followed by a linear-space offline* MWBM *algorithm run on S to compute a solution M is a $(3+\epsilon)$-approximate streaming* MWBM *algorithm.*

Proof. To see that this is a streaming algorithm, we recall that $|S| = \tilde{O}(\sum_v b_v)$, by Lemma 3.1. Since we compute M by running an offline linear-space algorithm on the subgraph induced by S, therefore using $O(|S|)$ space for this last step, the desired space bound follows.

To analyze the algorithm's approximation ratio, let $\alpha \in [0, 1]$ be the weighted fraction of OPT in S. That is, $w(OPT \cap S) = \alpha \cdot w(OPT)$, and by linearity, $w(OPT \setminus S) = (1 - \alpha) \cdot w(OPT)$. Therefore, by Equation (7) and Lemma 6.1 we have the following.

$$w(M) \geq w(OPT \cap S) = \alpha \cdot w(OPT).$$
$$w(M) \geq \frac{1}{2+\epsilon} \cdot w(OPT \setminus S) = \frac{1-\alpha}{2+\epsilon} \cdot w(OPT).$$

We thus find that the approximation ratio of this algorithm is at most $1/\min\{\alpha, \frac{1-\alpha}{2+\epsilon}\} \leq 3 + \epsilon$. □

Remark. We note that this approach—dual covering constraints for elements outside of the algorithm's memory S, and solving the problem optimally for S—is rather general. In particular, it applies to matching under any sub-additive (not just submodular) set function f, for which $f(OPT) \leq f(OPT \setminus S) + f(OPT \cap S)$. Moreover, this approach extends beyond matchings, to any

12

downward-closed constraints, for which $OPT \setminus S$ and $OPT \cap S$ are both feasible solutions. So, it seems like this approach could find applications to streaming algorithms for other objectives and constraints, provided dual feasibility can be guaranteed using a dual solution of value bounded by that of the output solution.

7 Lower Bound for MSM

Previous work shows that beating a $\frac{e}{e-1} \approx 1.582$ approximation for MSM in the streaming model is impossible for quasilinear space bounded algorithms [47], or polytime bounded algorithms [21, 25, 56]. In this section, we show that assuming the exponential time hypothesis (ETH), whereby **NP** $\not\subseteq$ **TIME**$(2^{o(n)})$ [46], beating 1.914 is impossible for any algorithm that is both space and time bounded. In particular, we will rely on seminal hardness of approximation results SET COVER from [21]. Recall:

Definition 7.1. A SET COVER *instance consists of a set system* $(\mathcal{U}, \mathcal{S})$, *with* $\mathcal{S} \subseteq 2^{\mathcal{U}}$. *The goal is to pick the smallest number* k *of sets* $S_1, \ldots, S_k \in \mathcal{S}$ *such that* $\left|\bigcup_{i \in [k]} S_i\right| = |\mathcal{U}|$. *We use* K *to denote the size of the minimal cover for the instance* $(\mathcal{U}, \mathcal{S})$, *and* $N = |\mathcal{U}| + |\mathcal{S}|$ *to denote the description size.*

Lemma 7.2 (Extension of Corollary 1.6 of [21]). *Assuming ETH, every algorithm achieving an approximation ratio* $(1 - \alpha) \ln |\mathcal{U}|$ *for* SET COVER *runs in time strictly greater than* $2^{N^{\gamma \cdot \alpha}}$ *for some* $\gamma > 0$. *Furthermore, this holds even under the assumptions that* $|\mathcal{S}| \leq K^{1/(\gamma\alpha)}$ *and* $|\mathcal{U}| \leq |\mathcal{S}|^{1/(\gamma\alpha)}$.

See Appendix D for a proof that the hardness holds even under the extra assumptions. To describe the instance, we will also use some extremal graph theory results from [34].

Definition 7.3. *An* α-*Ruzsa-Szemerédi graph* (α-RS *graph*) *is a bipartite graph* $G = (P, Q, E)$ *with* $|P| = |Q| = n$ *that is a union of induced matchings of size exactly* αn.

Theorem 7.4 (Lemma 53 of [34]). *For any constant* $\epsilon > 0$, *there exists a family of balanced bipartite* $(1/2 - \epsilon)$-RS *graphs with* $n^{1+\Omega(1/\log\log n)}$ *edges.*

In what follows we will show a randomized reduction from SET COVER to streaming MSM. Specifically, we will show that if there is a polytime streaming algorithm for MSM achieving ratio better than 1.914, then there is an algorithm for SET COVER violating Lemma 7.2. We proceed to describing our reduction.

The Reduction. The input is a SET COVER instance $(\mathcal{U}, \mathcal{S})$ for which the minimal cover contains K sets. Fix $n = 2^{k^{1/d}}$ for a degree d to be determined later.

We create an underlying bipartite graph $G = (L, R, E)$ with $n \operatorname{poly} \log n$ vertices as follows. The left/right vertex sets are partitioned into $L = P \sqcup P'$, $R = Q \sqcup Q'$. We let $|P| = |Q| = 2n$, and we let $|P'| = |Q'| = n \cdot |\mathcal{S}|/K$.

The edge set E arrives in two phases. In phase 1, all the edge of a set E_1 arrive, in phase 2 the edges of E_2 arrive. To define E_1, let G_0 be a fixed $(1/2 - \epsilon)$-RS graph with $m = \Omega\left(n^{1+1/\log\log n}\right)$ edges between P and Q, and let this graph be the union of the matchings $M_1, \ldots, M_t$. Let M_i' be a random subset of M_i of size $(1/2 - \delta)n$ for a parameter $\epsilon < \delta < 1/4$ and let $E_1 = M_1' \cup \ldots \cup M_t'$.

Choose one index $r \in [k]$ uniformly at random, and call M'_r the *distinguished matching*. Note that the index r is unknown to the algorithm.

Define E_2 as follows. Let $P_1 \sqcup P_2 \sqcup \ldots \sqcup P_{n/K}$ and $Q_1 \sqcup Q_2 \sqcup \ldots \sqcup Q_{n/K}$ be partitions of the the vertices of P and Q respectively not matched by M_r into subsets of size K. Similarly, let $P'_1 \sqcup P'_2 \sqcup \ldots \sqcup P'_{n/K}$ and $Q'_1 \sqcup Q'_2 \sqcup \ldots \sqcup Q'_{n/K}$ be partitions of P' and Q' into subsets of size $|\mathcal{S}|$. Let F_i be the edges of the complete bipartite graph between P_i and Q'_i, and let G_i be the edges of the complete bipartite graph between Q_i and P'_i. Finally, set $E_2 = \bigcup_i F_i \cup G_i$. See Figure 2.
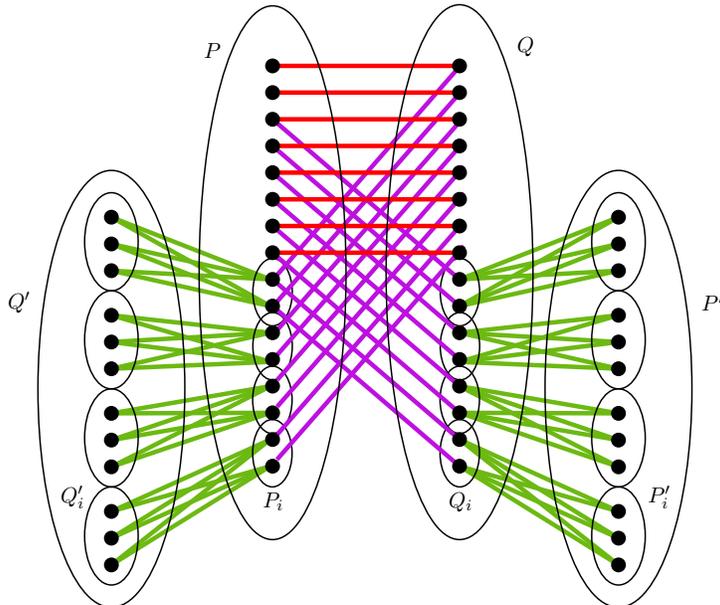

Figure 2: Illustration of lower bound instance.

Red edges represent the edges of the distinguished matching M_r in E_1, purple edges represent other edges in E_1, green edges represent edges of E_2. The red and purple edges together form the $(1/2 - \epsilon)$-RS graph G_0, subsampled.

It remains to describe the submodular function f. First, define the set function $f_1(E) = |E \cap E_1|$. Next, we define the function f_2 which is parametrized by the SET COVER instance. We identify each set of vertices P'_i and Q'_i with a disjoint copy of $\mathcal{S}$. For every edge $e \in E_2$, let $\phi(e)$ denote the set with which the endpoint of e in $P' \cup Q'$ is associated. Now, for some parameter $\eta > 0$ to be determined later, we define

$$f_2(E) := \frac{\eta K}{|\mathcal{U}|} \cdot \left| \bigcup_{e \in E} \phi(e) \right|.$$

Finally, set $f := f_1 + f_2$. Note that f is submodular since it the sum of a scaled coverage function and a linear function. On a technical note, since we assume that $|\mathcal{U}|$ is polynomially bounded in $|\mathcal{S}|$, we can represent the values of this function with poly $\log n$ bits.

Some intuition. Intuitively, we can imagine that all edges of E_1 are worth 1. We imagine that each edge of E_2 is a set in one of the copies of the instance $(\mathcal{S}, \mathcal{U})$, and we let the value of all edges selected in the second phase be the coverage of all the associated sets (scaled by $\eta K / |\mathcal{U}|$). First we we will argue that the algorithm can output almost none of the edges of M'_r, since it after phase 1 it has no information as to which matching is the distinguished one. Hence the majority

14

of the edges it uses from phase 1 must be from $E_1 \setminus M_r'$. However, each edge the algorithm chooses from $E_1 \setminus M_r'$ precludes it from taking between 1 and 2 edges of E_2. Furthermore, maximizing the value of edges of E_2 amounts to solving a hard MAX κ-COVERAGE instance. The coverage is scaled by the parameter η, and as a result, the algorithm is incentivized to take some $k := cK$ edges from each of the bipartite graphs (P_i, Q_i') (and (Q_i, P_i')) of E_2 and the remaining edges from $E_1 \setminus M_r'$. Meanwhile, OPT can take the distinguished matching edges M_r' as well as the edges of E_2 maximizing the coverage instance. Our bound will follow by setting η to maximize the ratio between these.

To start, we show that no streaming algorithm can "remember" more than a $o(1)$ fraction of the edges of the distinguished matching M_r. Since phase 1 of our construction is identical to the one in Appendix H of [34] which shows a 3/2 semi-streaming lower bound for max *weight* matching., we can reuse their result here.

Lemma 7.5 (Appendix H.1 of [34]). *For any constants $\gamma, \delta \in (0, 1/4)$, let $\mathcal{A}$ be an algorithm that at the end of phase 1, with constant probability, outputs at least γn of the the edges of M_r'. Then $\mathcal{A}$ uses $\Omega(E_1) \geq n^{1+\Omega(1/\log\log n)}$ bits of space.*

We reproduce a version of the proof in Appendix D for completeness. With this, we are finally ready to prove the main theorem of the section.

Theorem 7.6. *Assuming ETH, there exists a distribution over MSM instances such that any deterministic algorithm achieving an 1.914 approximation must use either $n^{1+\Omega(1/\log\log n)}$ space or $\Omega(2^{(\log n)^{10}})$ time.*

Proof. Our proof is a randomized polytime reduction from SET COVER to streaming MSM. We will show that if there is a randomized streaming algorithm achieving ratio better than 1.914 for MSM, then there is an algorithm for SET COVER achieving approximation ratio $(1-\alpha)\ln(|\mathcal{U}|)$ for constant $\alpha > 0$ that only requires polynomial extra overheard. We then argue that Lemma 7.2 implies that the streaming MSM algorithm must run in super polynomial time, assuming ETH.

Fix a deterministic algorithm A for streaming MSM. Now, given an instance of SET COVER $(\mathcal{U}, \mathcal{S})$ with minimum cover size K and description size $N = |\mathcal{U}| + |\mathcal{S}|$, create a random instance of streaming MSM according to the reduction described in this section. For each bipartite graph (P_i, Q_i') (or (Q_i, P_i')), if the algorithm A chooses cK edges from this graph, it can select at most $2(1-c)K$ edges from E_1 that are adjacent to P_i (or Q_i). Suppose WLOG that it can always achieve the $2(1-c)k$ bound. In this case we can also assume WLOG the algorithm chooses the same number $c \cdot K$ of edges from each such graph, and furthermore that it selects the same sets in the set system $(\mathcal{S}, \mathcal{U})$. Otherwise it can locally improve its solution by copying the solution for the best index i.

Suppose this solution achieves coverage of $(1 - e^{-c} + \gamma) \cdot |\mathcal{U}|$. Since the matchings $M_1, \ldots, M_t$ are induced, and by Lemma 7.5 w.h.p. the algorithm can only output $o(n)$ edges of M_r' after phase 1, the algorithm can only select $o(n)$ edges *not* incident to some P_i or Q_i. Thus the total value achieved by the algorithm is at most:

$$\left[2(1 - e^{-c} + \gamma) \cdot \eta \cdot K + 2(1-c) \cdot K\right] \cdot \frac{n}{K} + o(n)$$
$$\leq 2(1 - e^{-c} + \gamma) \cdot \eta n + 2(1-c) \cdot n + o(n)$$
$$\leq (2\eta - 2\ln(\eta) + 2\gamma) \cdot n + o(n),$$

15

where the last step follows since the expression is maximized at $c = \ln \eta$. On the other hand, the optimal solution can select the distinguished matching edge M'_r, as well as K edges adjacent to each set P_i corresponding to the minimum SET COVER solution. Thus the total value of OPT is at least:

$$(1 - \delta + 2\eta) \cdot n.$$

Thus the ratio between the maximum value achievable by the algorithm and the optimal value is bounded by:

$$\frac{1 + 2\eta - \delta}{2\eta - 2\ln(\eta) + 2\gamma + o(1)}.$$

Finally, we set $\eta = 2.09$ and let $\delta \to 0$. If this ratio converges to a value strictly below 1.914, then we can conclude that $\gamma = \Omega(1)$ and $\gamma > 0$.

We have shown that A can be used to pick cK sets with coverage $(1 - e^{-c} + \gamma) \cdot |\mathcal{U}|$. To finish the proof, we now show that this can be used to recover an approximation algorithm B for SET COVER. For convenience, set constant γ' such that $(1 - e^{-c-\gamma'}) := (1 - e^{-c} + \gamma)$. Then, guess K, and repeat algorithm A recursively $\lceil \ln |\mathcal{U}|/(c + \gamma') \rceil \le \ln |\mathcal{U}|/(c + \gamma') + 1$ times, each time on the residual uncovered set system. Each call to A covers $(1 - e^{-c-\gamma'})$ of the elements remaining, so after this number of iterations, the fraction of uncovered elements is less than $1/|\mathcal{U}|$, i.e. all elements are covered. Since each iteration costs $c \cdot K$, the total number of sets picked here is at most

$$c \left(\frac{\ln |\mathcal{U}|}{c + \gamma'} + 1 \right) \cdot K = \left(\frac{c}{c + \gamma'} + \frac{c}{\ln |\mathcal{U}|} \right) \cdot \ln |\mathcal{U}| \cdot K.$$

Defining the constant $\alpha = \gamma'/(c+\gamma')$, this is a $(1-\alpha-o(1)) \ln |\mathcal{U}|$ approximation to the SET COVER instance. Furthermore, if A runs in time T then B runs in time $\text{poly}(N) \cdot T$ (where $N = |\mathcal{U}| + |\mathcal{S}|$).

To conclude, if $T < 2^{N^\Delta}$ for a constant $\Delta < \gamma \cdot \alpha$, then B runs faster than $2^{N^{\gamma\alpha}}$, contradicting Lemma 7.2. Thus the algorithm A must run in time at least $2^{N^{\gamma\cdot\alpha}} \ge 2^{|\mathcal{S}|^{\gamma\cdot\alpha}} \ge 2^{(\log n)^{d\cdot\gamma\cdot\alpha}}$. Setting $d = 10/(\gamma \cdot \alpha)$, this running time is $2^{(\log n)^{10}}$, which is superpolynomial in n. $\qquad\square$

Theorem 1.3 therefore follows from Theorem 7.6 and Yao's minimax principle [71].

8 Conclusion and Open Questions

In this work we present a number of improved bounds for streaming submodular (b-)matching. Our work suggests a number of natural follow-up directions.

The first is to tighten (and ideally close) the gap between upper and lower bounds for the problems studied in this work. More ideas may be needed in order to resolve the optimal approximation ratio for these problems.

On the techniques side, it would be interesting to see whether the primal-dual method can be used for further applications in streaming submodular algorithms, or to further unify the analysis of prior work. More broadly, it would be interesting to apply our extension of the randomized primal dual method of [19] to other problems.

Finally, we point to a possible connection between our streaming matching algorithms, and *dynamic* matching algorithms. Algorithm 1 when applied to monotone b-matching problems stores in its stack a bounded-degree subgraph with $\tilde{O}(M_{\max}) = \tilde{O}(|OPT|)$ edges. Such size-optimal constant-approximate matching sparsifiers have proven particularly useful in the dynamic matching

literature. See [69] for a discussion, and [6, 10, 11, 12, 39, 64] for more applications of such dynamic matching sparsifiers. Is there a fast dynamic algorithm which maintains a sparisifier similar to that of Algorithm 1? This would result in improved dynamic weighted matching algorithms, and possibly would result in the first dynamic *submodular* matching algorithm. This would add to the recent interest in efficient dynamic algorithms for submodular optimization [36, 50].

Appendix

A Explaining Prior Work using LP Duality

In this section we further demonstrate the generality of our (randomized) primal-dual analysis, showing that it provides fairly simple alternative analyses of the algorithms of [16, 27], giving one unified analysis for these algorithms and ours. To keep things simple, we focus only on MSM, though [16, 27] show that their algorithms also work more broadly for k-matchoid and k-set system constraints.

In [16], Chakrabarti and Kale presented a reduction from MSM to MWM, by showing how to use a subclass of MWM algorithms to solve MSM. We now introduce the algorithm of [16] instantiated with the MWM algorithm of McGregor [57]. The algorithm is a natural and elegant one: when an edge e arrives, we consider its marginal gain with respect to the current matching. If this marginal gain is higher than some slack parameter C times the marginal gains of the currently blocking edges $e' \in N(e) \cap M$, we preempt those edges and add e to the matching.

In anticipation of our analysis of the algorithm of [27] in Appendix A.2, we generalize the algorithm's description and allow the algorithm to preempt with some probability $q \in [0, 1]$. The full pseudo-code is given in Algorithm 2.

Algorithm 2 The MSM Algorithm of [16] and [27]

Initialization
1: $M \leftarrow \emptyset$

Loop
2: **for** $t \in \{1, \ldots, |E|\}$ **do**
3:      $e \leftarrow e^{(t)}$
4:      $B(e) \leftarrow \sum_{e' \in N(e) \cap M} f(e' : M)$
5:      **if** $f(e : M) \leq C \cdot B(e)$ **then**
6:          **continue**   ▷ skip edge e
7:      **else**
8:          **with** probability q **do**
9:              $M \leftarrow (M \setminus N(e)) \cup \{e\}$
10: **return** M

This algorithm only ever adds an edge e to M upon its arrival. After adding an edge to M, this edge can be *preempted*, i.e., removed from M, after which it is never added back to M. Thus we note that this algorithm is not only a streaming algorithm, but also a so-called *preemptive* algorithm: it only stores a single matching in memory and therefore trivially requires $\tilde{O}(n)$ space.

For convenience, we let $M^{(t)}$ denote the matching M at time t, and let $S := \bigcup_t M^{(t)}$ denote the set of edges ever added to M. For an edge e let $B^{(t)}(e) := \sum_{e' \in N(e) \cap M^{(t)}} f(e' : M^{(t)})$. We will also denote by $P := S \setminus M$ the set of *preempted* edges.

A.1 The Framework of [16], Applied to the Algorithm of [57]

In this section we analyze the deterministic algorithm obtained by applying the framework of Chakrabarti and Kale [16] to the MWM algorithm of McGregor [57], corresponding to Algorithm 2 run with $q = 1$.

To argue about the approximation ratio, we will again fit a dual solution to this algorithm. Define the auxiliary submodular functions $g^S : 2^E \to \mathbb{R}^+$ to be $g^S(T) := f(S \cup T)$. Similarly to our analysis of Algorithm 1, we define the following dual.

$$\mu := f(S) = g^S(\emptyset),$$
$$\phi_v := C \cdot \max\{f(e : M^{(t)}) \mid t \in [|E|],\ v \in e \in M^{(t)}\},$$
$$\lambda_e := \begin{cases} f(e : S) & e \notin S \\ 0 & e \in S. \end{cases}$$

Note the difference here in the setting of ϕ_v from the algorithms of Sections 4 and 5. We start by showing that this is a dual feasible solution to the LP (D) for the function g^S.

Lemma A.1. *The dual solution $(\vec{\lambda}, \vec{\phi}, \mu)$ is feasible for the LP (D) with function g^S.*

Proof. To see that the first set of constraints are satisfied, note that by submodularity of f,

$$\sum_{e \in T} \lambda_e = \sum_{e \in T \setminus S} f(e : S) \geq \sum_{e \in T \setminus S} f_S(e) \geq f_S(T \setminus S) = f(S \cup T) - f(S) = g^S(T) - \mu.$$

For the second set of constraints, we note that if $e = e^{(t)} \notin S$, then by the test in Line 5 and submodularity, we have that

$$\lambda_e = f(e : S) \leq f(e : M^{(t-1)}) \leq C \cdot B^{(t-1)}(e) \leq \sum_{v \in e} \phi_v. \qquad \square$$

It remains to relate the value of the solution M to the cost of this dual. For this, we introduce the following useful notation. For any edge $e \in S$, we define the weight of e to be

$$w_e := \begin{cases} f(e : M) & e \in M \\ f(e : M^{(t)}) & e \in M^{(t-1)} \setminus M^{(t)} \subseteq P. \end{cases}$$

In words, the weight of an edge in the matching is $f(e : M)$, and the weight of a preempted edge is frozen to its last value before the edge was preempted. One simple consequence of the definition of the weights w_e is the following relationship to $f(M)$.

Observation A.2. $f(M) = \sum_{e \in M} f(e : M) = \sum_{e \in M} w_e = w(M)$.

We now show that the preempted edges' weight is bounded in terms of the weight of M.

Lemma A.3. *The weights of P and M satisfy $w(P) \leq w(M) \cdot \frac{1}{C-1}$.*

Proof. For any edge e, we define the following set of preempted edges which are preempted in favor of e or in favor of an edge (recursively) preempted due to e.[5] First, for an edge $e = e^{(t)}$, we let the set $P^1(e) := N(e) \cap M^{(t-1)}$ denote the edges preempted when e is added to M. For any $i > 1$, we let $P^i(e) := P^1(P^{i-1}(e))$ be the set of edges preempted by an edge with a trail of preemptions of length $i - 1$ from e. By Line 6, we have that any edge $e \in S$ has weight at least $w_e \geq C \cdot P^1(e)$. By induction, this implies that $w_e \geq C \cdot w(P^{(i-1)}(e)) \geq C^i \cdot w(P^i(e))$. Now, since each preempted edge $e' \in P$ belongs to precisely one set $P^i(e)$ for some $i \geq 1$ and $e \in M$, we find that indeed,

$$w(P) = \sum_{e \in M} \sum_{i \geq 1} w(P^i(e)) \leq \sum_{e \in M} \sum_{i \geq 1} \frac{1}{C^i} \cdot w_e = w(M) \cdot \left(\frac{1}{C} + \frac{1}{C^2} + \frac{1}{C^3} + \dots \right) = w(M) \cdot \frac{1}{C - 1}. \quad \square$$

Using Lemma A.3, we can now relate the value of the primal solution M to the cost of the our dual solution, $\mu + \sum_v \phi_v$. We start by bounding μ in terms of $f(M)$.

Lemma A.4. *The matching M output by Algorithm 2 satisfies $f(M) \geq \left(1 - \frac{1}{C}\right) \cdot \mu$.*

Proof. By submodularity of f, Lemma A.3, and Observation A.2 we obtain the desired inequality,

$$\mu = f(S) = f(M \cup P) \leq f(M) + \sum_{e \in P} f(e : M) = w(M) + w(P) \leq \left(1 + \frac{1}{C - 1}\right) \cdot f(M). \quad \square$$

We next bound $\sum_v \phi_v$ in terms of $f(M)$.

Lemma A.5. *The matching M output by Algorithm 2 run with $C > 1$ satisfies*

$$f(M) \geq \frac{1}{2C + C/(C - 1)} \cdot \sum_{v \in V} \phi_v.$$

Proof. Fix a vertex v and edge $e \in M^{(t-1)} \setminus M^{(t)} \subseteq P$ preempted at time t in favor of edge $e' = e^{(t)} \ni v$. For this edge e, by monotonicty in t' of $f(e' : M^{(t')})$, the test of Line 5, non-negativity of $f(e'' : M^{(t-1)})$ for any edge $e'' \in M^{(t-1)}$ and $C > 1$ we have that

$$w_{e'} \geq f(e' : M^{(t-1)}) \geq C \cdot B^{(t-1)}(e') \geq C \cdot f(e : M^{(t-1)}) = C \cdot w_e > w_e.$$

Consequently, again relying on monotonicity in t' of $f(e' : M^{(t')})$, we have that for any edge $e \in P$, there is at most one vertex $v \in e$ such that $w_e = f(e : M^{(t-1)})$ is equal to $\phi_v = \max\{f(e' : M^{(t')}) \mid v \in e' \in M^{(t')}\}$. Edges $e \in M$, on the other hand, clearly have $w_e = f(e : M)$ equal to $\phi_v = \max\{f(e' : M^{(t-1)}) \mid v \in e' \in M^{(t-1)} \setminus M^{(t)}\}$ for at most two vertices $v \in e$. Combined with Lemma A.3 and Observation A.2, this yields the desired inequality,

$$\sum_{v \in V} \phi_v \leq C \cdot \left(2 \sum_{e \in M} w_e + \sum_{e \in P} w_e \right) \leq \left(2C + \frac{C}{C - 1} \right) \cdot w(M) = \left(2C + \frac{C}{C - 1} \right) \cdot f(M). \quad \square$$

Equipped with the above lemmas, we can now analyze Algorithm 2's approximation ratio.

Theorem A.6. *Algorithm 2 run with $C > 1$ and $q = 1$ on a monotone MSM instance outputs a matching M of value*

$$\left(2C + \frac{2C}{C - 1} \right) \cdot f(M) \geq f(OPT).$$

This is optimized by taking $C = 2$, resulting in an approximation ratio of 8.

---

[5]In [26, 57], these sets are referred to by the somewhat morbid term "trail of the dead".

Proof. By weak LP duality and Lemma A.1, together with monotonicity of f, we have that

$$C \cdot \sum_v \phi_v + \mu \geq \max_T g^S(T) = \max_T f(S \cup T) \geq f(S \cup OPT) \geq f(OPT).$$

Combining Lemma A.5 and $f(M) = \mu$ by definition and rearranging, we get the desired inequality,

$$\left(2C + \frac{2C}{C-1}\right) \cdot f(M) \geq C \cdot \sum_v \phi_v + \mu \geq f(OPT). \qquad \square$$

As with our algorithm of Section 4, our analysis of Algorithm 2 relied on monotonicity, crucially using $f(S \cup OPT) \geq f(OPT)$. To extend this algorithm to non-monotone MSM, we again appeal to Lemma 2.2, setting $q = \frac{1}{2C+1}$. This is precisely the algorithm of Feldman et al. [27], which we analyze in the following section.

A.2   The Algorithm of Feldman et al. [27]

In [27], Feldman et al. showed how to generalize the algorithm of [16] to non-monotone function maximization. Here we show an analysis of their algorithm in our primal dual framework. Our proof is an extension of the one in Appendix A.1 in a way that is analogous to how Section 5 extends Section 4.

We reuse the same dual from Appendix A.1, only this time, both our dual object and the function g^S are random variables. The proof of expected dual feasibility for this variant of the algorithm of Appendix A.1 is analogous to that of Lemma 5.1, so we only outline the differences here.

We start with expected feasibility.

Lemma A.7. *The dual solution $(\mathbb{E}[\vec{\lambda}], \mathbb{E}[\vec{\phi}], \mathbb{E}[\mu])$ is feasible for the expected LP $\mathbb{E}[(D)]$.*

Proof (Sketch). The first set of constraints is satisfied for any random realization. Indeed, as in the proof of Lemma A.1, for any realization of S, by submodularity of f, we have

$$\sum_{e \in T} \lambda_e = \sum_{e \in T \setminus S} f(e : S) \geq \sum_{e \in T \setminus S} f_S(e) \geq f_S(T \setminus S) = f(S \cup T) - f(S) = g^S(T) - \mu.$$

Consequently, taking expectation over S, we have that indeed, $\mathbb{E}_S[\mu] + \sum_{e \in T} \mathbb{E}_S[\lambda_e] \geq \mathbb{E}_S[g^S(T)]$.

For the second set of constraints, the proof is nearly identical to that of Lemma 5.1, where we show that

$$\mathbb{E}\left[\sum_{v \in e} \phi_v\right] \geq \mathbb{E}[\lambda_e].$$

This is proved by taking total probability over the event $A_e := [f(e : S) \leq C \cdot \sum_{v \in V} \phi_v^{(t-1)}]$ and its complement. The key inequality to prove here is that for any realization of randomness R for which $\overline{A_e}$ holds, we have that

$$\mathbb{E}\left[\sum_{v \in e} \phi_v^{(t)} \,\middle|\, R\right] = 2q \cdot f(e : S) + (1 - 2q) \cdot \sum_{v \in e} \phi_v^{(t-1)} \geq 2q \cdot f(e : S).$$

And indeed, conditioned on R, the edge $e = e^{(t)}$ fails the test in Line 5, and so with probability q, we have $\sum_{v \in e} \phi_v^{(t)} = 2 \cdot f(e : S)$. To see this, note that if e is added to the matching, then for both $v \in e$, by definition $\phi_v^{(t)}$ must be at least $f(e : S)$. Hence, in this case

$$\mathbb{E}\left[\sum_{v \in e} \phi_v^{(t)} \,\middle|\, R\right] \geq 2q \cdot f(e : S).$$

The proof then proceeds as that of Lemma 5.1. $\qquad\square$

To relate the value of the solution M to the cost of the dual, we can define weights as in Appendix A.1 and reuse lemmas A.3, A.4, and A.5, which hold for every realization of the random choices of the algorithm. From here, following our template, we can use these along with LP duality, Lemma A.7 and Lemma 2.2, to analyze this algorithm.

Theorem A.8. *Algorithm 2 run with $q = 1/(2C + 1)$ and C on a non-monotone MSM instance outputs a matching M of value*

$$\left(\frac{2C^2 + C}{C - 1}\right) \cdot f(M) \geq f(OPT).$$

This is optimized by taking $C = 1 + \frac{\sqrt{3}}{2}$, resulting in an approximation ratio of $5 + 2\sqrt{6} \approx 9.899$. Moreover, the same algorithm is $2C + 2C/(C - 1)$ approximate for monotone MSM, yielding an approximation ratio of 8 for $C = 2$.

B   Tight instance for Algorithm 1

In this section we show that there exists a family of instances of MSM instances parametrized by C for which Algorithm 1 with parameter $C > 1$ yields an approximation factor of $2C + C/(C - 1)$.

Lemma B.1. *The approximation ratio of Algorithm 1 with $C > 1$ and $q = 1$ for monotone MSM is at least $2C + \frac{C}{C-1}$.*

Proof. Define the graph G as follows. The vertex set $V(G)$ consists of $\{x_i, y_i\}_{i \in [0,n]}$. For convenience, for every $i \in [1, n]$ we define the edges $d_i = (x_0, x_i)$ and $e_i = (x_i, y_i)$. Then the edge set $E(G)$ consists of the edges $\{d_i\}_{i=1}^{n} \cup \{e_i\}_{i=0}^{n}$.

To define the (monotone) submodular function, we first define an auxiliary weight function $w : E(G) \to \mathbb{R}_{\geq 0}$. The weights are:

$$w(d_i) = C^{i-1} \qquad\qquad\qquad\qquad\qquad (n \geq i \geq 1)$$
$$w(e_1) = 1 + C - \epsilon$$
$$w(e_i) = C^i - \epsilon \qquad\qquad\qquad\qquad\qquad (n \geq i \geq 2)$$
$$w(e_0) = C^n - \epsilon$$

Now the submodular function is:

$$f(T) := w(T \cap \{e_0\}) + \sum_{i=0}^{n} \min(w(T \cap \{d_i, e_i\}), w(e_i))$$
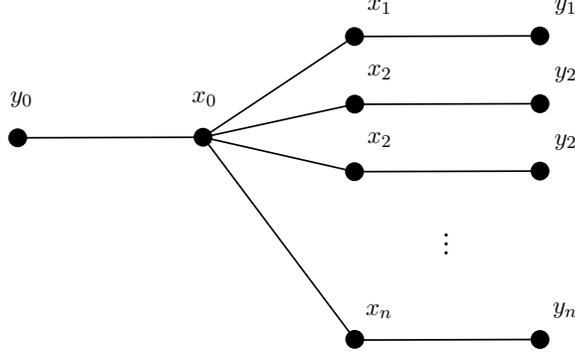
Figure 4: Tight Example for Algorithm 1

Since weights are non-negative, this function is monotone. Submodularity follows from preservation of subdmodularity under linear combinations (and in particular sums), and $\min\{w(S), X\}$ being submodular for any linear function w.

The stream reveals the edges $d_1, \ldots, d_n$ in order, and subsequently reveals $e_0, e_1, \ldots, e_n$ in order. For a run of Algorithm 1 with this choice of C and $q = 1$, several claims hold inductively:

(a) On the arrival of edge d_i, we have $\phi_{x_0} = C^{i-2}$ (except for the arrival of d_1, at which point $\phi_0 = 0$) and $\phi_{x_i} = 0$.

(b) The algorithm takes every edge d_i into the stack.

(c) After d_i is taken into the stack, we have $\phi_{x_0} = C^{i-1}$ and $\phi_{x_i} = C^{i-1} + C^{i-2}$ (except for ϕ_{x_1} which is set to 1).

(d) The algorithm does not take e_i into the stack.

Let Λ_t be the statement that these claims holds for time t. By inspection Λ_1 holds, now consider some time $i > 1$. Claim (a) follows directly from claim (c) of Λ_{i-1}. Claim (b) follows from (a) since $f_S(d_i) = C^{i-1} = C \cdot \phi_0$ when d_i arrives. Claim (c) is a consequence of how the algorithm increases the potentials ϕ when taking edges into the stack. Claim (d) holds since $f_S(e_i) = w(e_i) - w(d_i) = C^i - C^{i-1} - \epsilon < C \cdot \phi_{x_i}$.

From the above, we find that Algorithm 1 with parameter C as above and $q = 1$ will have all edges $d_1, \ldots, d_n$ in its stack by the end, resulting in it outputting the matching consisting of the single edge d_n. The value of this edge (and hence this matching) is C^{n-1}, while on the other hand OPT can take the edges $\{e_i\}_{i=0}^n$, which have value

$$\sum_{i=0}^n w(e_i) = C^n + \sum_{i=0}^n C^i - \epsilon(n+1) \to C^{n-1}\left(2C + \frac{C}{C-1}\right). \qquad \text{(as } n \to \infty \text{ and } \epsilon \to 0)$$

so long as $C > 1$. Hence $c(OPT)/c(\text{ALG}) \to 2C + C/(C-1)$. The lemma follows. $\qquad \square$

C Space Bound of Algorithm 1

In this section we bound the space and time complexities of Algorithm 1. We start by bounding its space usage, as restated in the following lemma.

Lemma 3.1. *For any constant $\epsilon > 0$, Algorithm 1 run with $C = 1 + \epsilon$ uses $\tilde{O}(M_{\max})$ space.*

First, we prove the following simpler lemma.

Lemma C.1. *Each vertex $v \in V$ has at most $\tilde{O}(b_v)$ edges in the stack S.*

Proof. If an edge $e \ni v$ is added to S at time t, then by the test in Line 6, $f(e : S) \geq (1 + \epsilon) \cdot \sum_{u \in e} \phi_u^{(t-1)}$. Consequently, and since ϕ values are easily seen to always be positive, we have

$$\phi_v^{(t)} - \phi_v^{(t-1)} = \frac{f(e : S) - \sum_{u \in e} \phi_u^{(t-1)}}{b_v}$$

$$\geq \frac{\epsilon \cdot \sum_{u \in e} \phi_u^{(t-1)}}{b_v} \geq \frac{\epsilon \cdot \phi_v^{(t-1)}}{b_v}.$$

Thus, adding this edge $e \ni v$ to S results in $\phi_v^{(t)} \geq \phi_v^{(t-1)} \cdot (1 + \epsilon/b_v)$. Moreover, if e is the first edge of v added to S, then we have

$$\phi_v^{(t)} = \frac{f(e : S) - \sum_{u \in e} \phi_u^{(t-1)}}{b_v} \geq \frac{\epsilon}{1 + \epsilon} \cdot \frac{f(e : S)}{b_v}$$

$$\geq \frac{\epsilon}{1 + \epsilon} \cdot \frac{f_{\min}}{b_v}.$$

Hence, if v had k edges added to the stack by time t,

$$\phi_v^{(t)} \geq (\epsilon/(1 + \epsilon)) \cdot (f_{min}/b_v) \cdot (1 + \epsilon/b_v)^{k-1}. \tag{8}$$

On the other hand, since f is polynomially bounded, we have that for some constant d

$$\phi_v^{(t)} \leq \sum_{e \ni v} f_{S_e}(e)/b_v \leq n^d \cdot (f_{min}/b_v). \tag{9}$$

Combining equations (8) and (9) and simplifying, we find that $(1 + \epsilon/b_v)^{k-1} \leq n^d \cdot (1 + \epsilon)/\epsilon$. Taking out logarithms and simplifying further, we find that

$$k \leq 1 + \frac{d \log n + \log(1 + \epsilon) + \log(1/\epsilon)}{\log(1 + \epsilon/b_v)} = O((b_v/\epsilon) \cdot (\log n + \log(1/\epsilon))) = \tilde{O}(b_v).$$

That is, the number of edges of v in the stack is at most $\tilde{O}(b_v)$. $\qquad\square$

Proof of Lemma 3.1. Consider a maximum cardinality b-matching M (i.e., $|M| = M_{\max}$). Denote by d_v the degree of a vertex v in G. Then, the set U of saturated vertices $v \in V$ in M, i.e., those with $\min\{b_v, d_v\}$ edges in M, is a vertex cover. To see this, we note that no edge $e = (u, v)$ can have both endpoints not saturated by M, as the converse would imply that $M \cup \{e\}$ is a feasible b-matching, contradicting maximality of M. Now, since each edge of M has at most two saturated endpoints, we have that $\sum_{u \in U} \min\{b_u, d_u\} \leq 2|M|$. On the other hand, since U is a vertex cover, the number of edges in S is upper bounded by the number of edges of vertices in U in S. But each vertex $u \in U$ trivially has at most d_u edges in S. Combined with Lemma C.1, we find that the number of edges in S is at most

$$|S| = \tilde{O}\left(\sum_{u \in U} \min\{b_u, d_u\}\right) = \tilde{O}(|M|). \qquad\square$$

Finally, since each edge added to S causes at most two nodes to have non-zero ϕ value, and since each such value can be specified using $\tilde{O}(1)$ bits, by the assumption of f being poly-bounded, the space to store all non-zero ϕ values is at most $\tilde{O}(|S|) = \tilde{O}(|M|) = \tilde{O}(M_{\max})$.

Finally, we briefly analyze the algorithm's running time.

Lemma 3.2. *A randomized (deterministic) implementation of Algorithm 1 requires $O(1)$ ($O(\log n)$) operations and $O(1)$ function evaluations per arrival, followed by $\tilde{O}(M_{\max})$ time post-processing. Using $\tilde{O}(n)$ space, the deterministic time per arrival can be decreased to $O(1)$, by storing all ϕ_v.*

Proof. The algorithm clearly requires a constant number of evaluations of ϕ_v and f for each edge arrival. The difference between deterministic and randomized implementations is due to the data structures used to maintain the mapping from v to ϕ_v (if non zero). Finally, the preprocessing time follows directly from our upper bound on $|S|$ and the post-processing stage clearly taking time linear in $|S|$. □

D   Deferred Proofs of Section 7

Lemma 7.2 (Extension of Corollary 1.6 of [21])**.** *Assuming ETH, every algorithm achieving an approximation ratio $(1-\alpha)\ln|\mathcal{U}|$ for SET COVER runs in time strictly greater than $2^{N^{\gamma\cdot\alpha}}$ for some $\gamma > 0$. Furthermore, this holds even under the assumptions that $|\mathcal{S}| \leq K^{1/(\gamma\alpha)}$ and $|\mathcal{U}| \leq |\mathcal{S}|^{1/(\gamma\alpha)}$.*

Proof. The first statement is precisely Corollary 1.6 of [21].

For the extra assumptions, if $K < |\mathcal{S}|^{\gamma\alpha}$ then the brute force algorithm that checks all subsets of size K runs in time $|\mathcal{S}|^K < 2^{|\mathcal{S}|^{\gamma\alpha}\log|\mathcal{S}|} \leq 2^{N^{\gamma\alpha}}$. If $|\mathcal{S}| < |\mathcal{U}|^{\gamma\alpha}$, then one can brute force over all sub collections of S in time $2^{|\mathcal{S}|} \leq 2^{|\mathcal{U}|^{\gamma\alpha}} \leq 2^{N^{\gamma\alpha}}$. Both running times contradict Lemma 7.2. □

Lemma 7.5 (Appendix H.1 of [34])**.** *For any constants $\gamma, \delta \in (0, 1/4)$, let $\mathcal{A}$ be an algorithm that at the end of phase 1, with constant probability, outputs at least γn of the the edges of M'_r. Then $\mathcal{A}$ uses $\Omega(E_1) \geq n^{1+\Omega(1/\log\log n)}$ bits of space.*

Proof. Let $\mathcal{A}$ be an algorithm that outputs γn of the edges of M'_r at the end of phase 1 that uses fewer than $s = n\,\text{poly}\log n$ bits. We will show that $\gamma = o(1)$.

Let $\mathcal{G}$ be the set of possible first phase graphs. Then

$$|\mathcal{G}| = \binom{n/2}{\delta n}^t = 2^{\gamma m}$$

for some $\gamma > 0$. Let $\phi : \mathcal{G} \to \{0,1\}^s$ be the function that takes an input graph G to the state of the algorithm $\mathcal{A}$ after running $\mathcal{A}$ on G. Let $\Gamma(G) = \{H \mid \phi(G) = \phi(H)\}$, that is the set of graphs inducing the same internal state for $\mathcal{A}$ at the end of phase 1.

Define $\Psi(G) = \bigcap_{H\in\Gamma(G)} E(H)$. Note that for any input graph G, the algorithm $\mathcal{A}$ can output an edge e if and only if $e \in \Psi(G)$. Also, for any G let t' be the number of matchings in the RS graph G_0 for which $\Psi(G)$ contains at least γn edges. Since algorithm $\mathcal{A}$ outputs γn edges of M'_r, the number of graphs in $\Gamma(G)$ is bounded by

$$\binom{(1/2-\gamma)n}{\delta n}^{t'}\binom{n/2}{\delta n}^{t-t'} = \left(2^{-\Omega(\gamma n)}\binom{n/2}{\delta n}\right)^{t'}\binom{n/2}{\delta n}^{t-t'} = 2^{-\Omega(t'\gamma n)}2^{\gamma m} \qquad (*)$$

On the other hand, since the first phase graph G is chosen uniformly at random, by a counting argument, with probability at least $1 - o(1)$ we have that $|\Gamma(G)| \geq 2^{(\gamma - o(1))m}$. Conditioning on this happening, we also know that $t' \geq \Omega(t)$ since the input graph is uniformly chosen within $\Gamma(G)$, and the algorithm succeeds with constant probability. These two facts together with $(*)$ imply that $\gamma = o(1)$. $\qquad\square$

References

[1] Bertinoro workshop 2014, problem 63. https://sublinear.info/index.php?title=Open_Problems:63. Accessed: 2020-06-20.

[2] AGRAWAL, R., GOLLAPUDI, S., HALVERSON, A., AND IEONG, S. 2009. Diversifying search results. In *Proceedings of 2nd International Conference on Web Search and Data Mining (WSDM)*. 5–14.

[3] AHMED, F., DICKERSON, J. P., AND FUGE, M. 2017. Diverse weighted bipartite b-matching. In *Proceedings of 26th International Joint Conference on Artificial Intelligence (IJCAI)*. 35–41.

[4] ALALUF, N., ENE, A., FELDMAN, M., NGUYEN, H. L., AND SUH, A. 2020. Optimal streaming algorithms for submodular maximization with cardinality constraints. In *Proceedings of 47th International Colloquium on Automata, Languages and Programming (ICALP)*. 6:1–6:19.

[5] ANSTEE, R. P. 1987. A polynomial algorithm for b-matchings: an alternative approach. *Inf. Process. Lett. 24,* 3, 153–157.

[6] ARAR, M., CHECHIK, S., COHEN, S., STEIN, C., AND WAJC, D. 2018. Dynamic matching: Reducing integral algorithms to approximately-maximal fractional algorithms. In *Proceedings of 45th International Colloquium on Automata, Languages and Programming (ICALP)*. 7:1–7:16.

[7] BADANIDIYURU, A., MIRZASOLEIMAN, B., KARBASI, A., AND KRAUSE, A. 2014. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of 20th International Conference on Knowledge Discovery and Data Mining (KDD)*. 671–680.

[8] BAR-NOY, A., BAR-YEHUDA, R., FREUND, A., NAOR, J., AND SCHIEBER, B. 2001. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM (JACM) 48,* 5, 1069–1090.

[9] BAR-YEHUDA, R. AND RAWITZ, D. 2005. On the equivalence between the primal-dual schema and the local ratio technique. *SIAM Journal on Discrete Mathematics 19,* 3, 762–797.

[10] BERNSTEIN, A. AND STEIN, C. 2015. Fully dynamic matching in bipartite graphs. In *Proceedings of 42nd International Colloquium on Automata, Languages and Programming (ICALP)*. 167–179.

[11] BERNSTEIN, A. AND STEIN, C. 2016. Faster fully dynamic matchings with small approximation ratios. In *Proceedings of 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 692–711.

[12] Bhattacharya, S., Henzinger, M., and Italiano, G. F. 2015. Deterministic fully dynamic data structures for vertex cover and matching. In *Proceedings of 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 785–804.

[13] Buchbinder, N. and Feldman, M. 2019. Constrained submodular maximization via a nonsymmetric technique. *Mathematics of Operations Research 44*, 3, 988–1005.

[14] Buchbinder, N., Feldman, M., Naor, J., and Schwartz, R. 2014. Submodular maximization with cardinality constraints. In *Proceedings of 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1433–1452.

[15] Calinescu, G., Chekuri, C., Pál, M., and Vondrák, J. 2011. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing (SICOMP) 40*, 6, 1740–1766.

[16] Chakrabarti, A. and Kale, S. 2015. Submodular maximization meets streaming: Matchings, matroids, and more. *Mathematical Programming 154*, 1-2, 225–247.

[17] Chekuri, C., Gupta, S., and Quanrud, K. 2015. Streaming algorithms for submodular function maximization. In *Proceedings of 42nd International Colloquium on Automata, Languages and Programming (ICALP)*. 318–330.

[18] Crouch, M. and Stubbs, D. M. 2014. Improved streaming algorithms for weighted matching, via unweighted matching. In *Proceedings of 17th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*. 96.

[19] Devanur, N. R., Jain, K., and Kleinberg, R. D. 2013. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 101–107.

[20] Dickerson, J. P., Sankararaman, K. A., Srinivasan, A., and Xu, P. 2019. Balancing relevance and diversity in online bipartite matching via submodularity. In *Proceedings of 53rd AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 33. 1877–1884.

[21] Dinur, I. and Steurer, D. 2014. Analytical approach to parallel repetition. In *Proceedings of 46th Annual ACM Symposium on Theory of Computing (STOC)*. 624–633.

[22] Ene, A. and Nguyen, H. L. 2016. Constrained submodular maximization: Beyond 1/e. In *Proceedings of 57th Symposium on Foundations of Computer Science (FOCS)*. 248–257.

[23] Epstein, L., Levin, A., Segev, D., and Weimann, O. 2013. Improved bounds for online preemptive matching. In *Proceedings of 30th International Symposium on Theoretical Aspects of Computer Science (STACS)*. 389.

[24] Fahrbach, M., Huang, Z., Tao, R., and Zadimoghaddam, M. 2020. Edge-weighted online bipartite matching. In *Proceedings of 61st Symposium on Foundations of Computer Science (FOCS)*. To appear.

[25] Feige, U. 1998. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM) 45*, 4, 634–652.

[26] FEIGENBAUM, J., KANNAN, S., MCGREGOR, A., SURI, S., AND ZHANG, J. 2005. On graph problems in a semi-streaming model. *Theor. Comput. Sci. 348,* 2-3, 207–216.

[27] FELDMAN, M., KARBASI, A., AND KAZEMI, E. 2018. Do less, get more: streaming submodular maximization with subsampling. In *Proceedings of 31st Annual Conference on Neural Information Processing Systems (NeurIPS).* 732–742.

[28] FELDMAN, M., NAOR, J., AND SCHWARTZ, R. 2011a. A unified continuous greedy algorithm for submodular maximization. In *Proceedings of 52nd Symposium on Foundations of Computer Science (FOCS).* 570–579.

[29] FELDMAN, M., NAOR, J. S., SCHWARTZ, R., AND WARD, J. 2011b. Improved approximations for k-exchange systems. In *Proceedings of 19th Annual European Symposium on Algorithms (ESA).* 784–798.

[30] FELDMAN, M., NOROUZI-FARD, A., SVENSSON, O., AND ZENKLUSEN, R. 2020. The one-way communication complexity of submodular maximization with applications to streaming and robustness. In *Proceedings of 52nd Annual ACM Symposium on Theory of Computing (STOC).* 1363–1374.

[31] FISHER, M. L., NEMHAUSER, G. L., AND WOLSEY, L. A. 1978. An analysis of approximations for maximizing submodular set functions—ii. In *Polyhedral combinatorics.* Springer, 73–87.

[32] GABOW, H. N. 2018. Data structures for weighted matching and extensions to b-matching and f-factors. *Transactions on Algorithms (TALG) 14,* 3, 1–80.

[33] GHAFFARI, M. AND WAJC, D. 2019. Simplified and space-optimal semi-streaming $(2 + \epsilon)$-approximate matching. In *Proceedings of 2nd Symposium of Simplicity in Algorithms (SOSA).*

[34] GOEL, A., KAPRALOV, M., AND KHANNA, S. 2012. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* 468–485.

[35] GUPTA, A., KRISHNASWAMY, R., KUMAR, A., AND PANIGRAHI, D. 2017. Online and dynamic algorithms for set cover. In *Proceedings of 49th Annual ACM Symposium on Theory of Computing (STOC).* 537–550.

[36] GUPTA, A. AND LEVIN, R. 2020a. Fully-dynamic submodular cover with bounded recourse. In *Proceedings of 61st Symposium on Foundations of Computer Science (FOCS).* To appear.

[37] GUPTA, A. AND LEVIN, R. 2020b. The online submodular cover problem. In *Proceedings of 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* 1525–1537.

[38] GUPTA, A., ROTH, A., SCHOENEBECK, G., AND TALWAR, K. 2010. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *Proceedings of 6th Conference on Web and Internet Economics (WINE).* 246–257.

[39] GUPTA, M. AND PENG, R. 2013. Fully dynamic $(1+\epsilon)$-approximate matchings. In *Proceedings of 54th Symposium on Foundations of Computer Science (FOCS).* 548–557.

[40] HUANG, Z., KANG, N., TANG, Z. G., WU, X., ZHANG, Y., AND ZHU, X. 2018a. How to match when all vertices arrive online. In *Proceedings of 50th Annual ACM Symposium on Theory of Computing (STOC)*. 17–29.

[41] HUANG, Z., PENG, B., TANG, Z. G., TAO, R., WU, X., AND ZHANG, Y. 2019. Tight competitive ratios of classic matching algorithms in the fully online model. In *Proceedings of 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2875–2886.

[42] HUANG, Z., TANG, Z. G., WU, X., AND ZHANG, Y. 2018b. Online vertex-weighted bipartite matching: Beating 1-1/e with random arrivals. In *Proceedings of 45th International Colloquium on Automata, Languages and Programming (ICALP)*. 1070–1081.

[43] HUANG, Z., TANG, Z. G., WU, X., AND ZHANG, Y. 2020a. Fully online matching ii: Beating ranking and water-filling. In *Proceedings of 61st Symposium on Foundations of Computer Science (FOCS)*. To appear.

[44] HUANG, Z. AND ZHANG, Q. 2020. Online primal dual meets online matching with stochastic rewards: configuration lp to the rescue. In *Proceedings of 52nd Annual ACM Symposium on Theory of Computing (STOC)*. 1153–1164.

[45] HUANG, Z., ZHANG, Q., AND ZHANG, Y. 2020b. Adwords in a panorama. In *Proceedings of 61st Symposium on Foundations of Computer Science (FOCS)*. To appear.

[46] IMPAGLIAZZO, R. AND PATURI, R. 2001. On the complexity of k-sat. *Journal of Computer and System Sciences 62,* 2, 367–375.

[47] KAPRALOV, M. 2013. Better bounds for matchings in the streaming model. In *Proceedings of 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1679–1697.

[48] KAZEMI, E., MITROVIC, M., ZADIMOGHADDAM, M., LATTANZI, S., AND KARBASI, A. 2019. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In *Proceedings of 36th International Conference on Machine Learning (ICML)*. 3311–3320.

[49] KORULA, N., MIRROKNI, V., AND ZADIMOGHADDAM, M. 2018. Online submodular welfare maximization: Greedy beats 1/2 in random order. *SIAM Journal on Computing (SICOMP) 47,* 3, 1056–1086.

[50] LATTANZI, S., MITROVIĆ, S., NOROUZI-FARD, A., TARNAWSKI, J., AND ZADIMOGHADDAM, M. 2020. Fully dynamic algorithm for constrained submodular optimization. In *Proceedings of 34th Annual Conference on Neural Information Processing Systems (NeurIPS)*. To appear.

[51] LEE, J., MIRROKNI, V. S., NAGARAJAN, V., AND SVIRIDENKO, M. 2009. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of 41st Annual ACM Symposium on Theory of Computing (STOC)*. 323–332.

[52] LEE, J., MIRROKNI, V. S., NAGARAJAN, V., AND SVIRIDENKO, M. 2010a. Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM Journal on Discrete Mathematics 23,* 4, 2053–2078.

[53] Lee, J., Sviridenko, M., and Vondrák, J. 2010b. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research 35,* 4, 795–806.

[54] Lehmann, B., Lehmann, D., and Nisan, N. 2006. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior 55,* 2, 270–296.

[55] Lin, H. and Bilmes, J. 2011. Word alignment via submodular maximization over matroids. In *Proceedings of 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL).* 170–175.

[56] Manurangsi, P. 2020. Tight running time lower bounds for strong inapproximability of maximum k-coverage, unique set cover and related problems (via t-wise agreement testing theorem). In *Proceedings of 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* 62–81.

[57] McGregor, A. 2005. Finding graph matchings in data streams. In *Proceedings of 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX).* 170–181.

[58] Mirzasoleiman, B., Jegelka, S., and Krause, A. Streaming non-monotone submodular maximization: Personalized video summarization on the fly. In *Proceedings of 32nd AAAI Conference on Artificial Intelligence (AAAI).* 1379–1386.

[59] Nemhauser, G. L. and Wolsey, L. A. 1978. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research 3,* 3, 177–188.

[60] Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming 14,* 1, 265–294.

[61] Norouzi-Fard, A., Tarnawski, J., Mitrovic, S., Zandieh, A., Mousavifar, A., and Svensson, O. 2018. Beyond 1/2-approximation for submodular maximization on massive data streams. In *Proceedings of 35th International Conference on Machine Learning (ICML).* 3829–3838.

[62] Oveis Gharan, S. and Vondrák, J. 2011. Submodular maximization by simulated annealing. In *Proceedings of 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* 1098–1116.

[63] Paz, A. and Schwartzman, G. 2018. A $(2+\epsilon)$-approximation for maximum weight matching in the semi-streaming model. *Transactions on Algorithms (TALG) 15,* 2, 18.

[64] Peleg, D. and Solomon, S. 2016. Dynamic $(1 + \epsilon)$-approximate matchings: a density-sensitive approach. In *Proceedings of 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* 712–729.

[65] Tang, Z. G., Wu, X., and Zhang, Y. 2020. Towards a better understanding of randomized greedy matching. In *Proceedings of 52nd Annual ACM Symposium on Theory of Computing (STOC).* 1097–1110.

[66] Vondrák, J. 2007. Submodularity in combinatorial optimization.

[67] VONDRÁK, J. 2008. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of 40th Annual ACM Symposium on Theory of Computing (STOC)*. 67–74.

[68] VONDRÁK, J. 2013. Symmetry and approximability of submodular maximization problems. *SIAM Journal on Computing (SICOMP) 42,* 1, 265–304.

[69] WAJC, D. 2020. Rounding dynamic matchings against an adaptive adversary. In *Proceedings of 52nd Annual ACM Symposium on Theory of Computing (STOC)*. 194–207.

[70] WOLSEY, L. A. 1982. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica 2,* 4, 385–393.

[71] YAO, A. C.-C. 1977. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of 18th Symposium on Foundations of Computer Science (FOCS)*. 222–227.