Proceedings of Machine Learning Research vol 65:1–27, 2017

Greed Is Good: Near-Optimal Submodular Maximization via Greedy Optimization

Moran Feldman MORANFE@OPENU.AC.IL
Department of Mathematics and Computer Science
The Open University of Israel

Christopher Harshaw CHRISTOPHER.HARSHAW@YALE.EDU
Department of Computer Science
Yale Institute for Network Science
Yale University

Amin Karbasi AMIN.KARBASI@YALE.EDU
Department of Electrical Engineering and Computer Science
Yale Institute for Network Science
Yale University

Abstract

It is known that greedy methods perform well for maximizing *monotone* submodular functions. At the same time, such methods perform poorly in the face of non-monotonicity. In this paper, we show—arguably, surprisingly—that invoking the classical greedy algorithm $O(\sqrt{k})$-times leads to the (currently) fastest deterministic algorithm, called REPEATEDGREEDY, for maximizing a general submodular function subject to k-independent system constraints. REPEATEDGREEDY achieves $(1 + O(1/\sqrt{k}))k$ approximation using $O(nr\sqrt{k})$ function evaluations (here, n and r denote the size of the ground set and the maximum size of a feasible solution, respectively). We then show that by a careful sampling procedure, we can run the greedy algorithm only *once* and obtain the (currently) fastest randomized algorithm, called SAMPLEGREEDY, for maximizing a submodular function subject to k-extendible system constraints (a subclass of k-independent system constrains). SAMPLEGREEDY achieves $(k + 3)$-approximation with only $O(nr/k)$ function evaluations. Finally, we derive an almost matching lower bound, and show that no polynomial time algorithm can have an approximation ratio smaller than $k + 1/2 - \varepsilon$. To further support our theoretical results, we compare the performance of REPEATEDGREEDY and SAMPLEGREEDY with prior art in a concrete application (movie recommendation). We consistently observe that while SAMPLEGREEDY achieves practically the same utility as the best baseline, it performs at least two orders of magnitude faster.

Keywords: Submodular maximization, k-systems, k-extendible systems, approximation algorithms

1. Introduction

Submodular functions (Edmonds, 1971; Fujishige, 2005), originated in combinatorial optimization and operations research, exhibit a natural diminishing returns property common in many well known objectives: the marginal benefit of any given element decreases as more and more elements are selected. As a result, submodular optimization has found numerous applications in machine learning, including viral marketing (Kempe et al., 2003), network monitoring (Leskovec et al., 2007;

Gomez Rodriguez et al., 2010), sensor placement and information gathering (Guestrin et al., 2005), news article recommendation (El-Arini et al., 2009; Mirzasoleiman et al., 2016b), nonparametric learning (Reed and Ghahramani, 2013), document and corpus summarization (Lin and Bilmes, 2011; Kirchhoff and Bilmes, 2014; Sipos et al., 2012), data summarization (Mirzasoleiman et al., 2013, 2015b), crowd teaching (Singla et al., 2014), and MAP inference of determinental point process (Gillenwater et al., 2012). The usefulness of submodular optimization in these settings stems from the fact that many such problems can be reduced to the problem of maximizing a submodular function subject to feasibility constraints, such as cardinality, knapsack, matroid or intersection of matroids constraints.

In this paper, we consider the maximization of submodular functions subject to two important classes of constraints known as k-system and k-extendible system constraints. The class of k-system constraints is a very general class of constraints capturing, for example, any constraint which can be represented as the intersection of multiple matroid and matching constraints. The study of the maximization of *monotone* submodular functions subject to a k-system constraint goes back to the work of Fisher et al. (1978), who showed that the natural greedy algorithm achieves an approximation ratio of $1/(k+1)$ for this problem (k is a parameter of the k-system constraint measuring, intuitively, its complexity). In contrast, results for the maximization of *non-monotone* submodular functions subject to a k-system constraint were only obtained much more recently. Specifically, Gupta et al. (2010) showed that, by repeatedly executing the greedy algorithm and an algorithm for unconstrained submodular maximization, one can maximize a non-monotone submodular function subject to a k-system constraint up to an approximation ratio of roughly $3k$ using a time complexity of $O(nrk)$[1]. This was recently improved by Mirzasoleiman et al. (2016a), who showed that the approximation ratio obtained by the above approach is in fact roughly $2k$.

In this paper, we describe algorithms that improve over the above mentioned results both in terms of the approximation ratio and in terms of the time complexity. Our first result is a *deterministic* algorithm, called REPEATEDGREEDY, which obtains an approximation ratio of $(1+O(1/\sqrt{k}))k$ for the maximization of a non-monotone submodular function subject to a k-system constraint using a time complexity of only $O(nr\sqrt{k})$. REPEATEDGREEDY is structurally very similar to the algorithm of Gupta et al. (2010) and Mirzasoleiman et al. (2016a). However, thanks to a tighter analysis, it needs to execute the greedy algorithm and the algorithm for unconstrained submodular maximization much less often, which yields its improvement in the time complexity.

Our second result is a *randomized* algorithm, called SAMPLEGREEDY, which manages to further push the approximation ratio and time complexity to $(k+1)^2/k \leq k+3$ and $O(n+nr/k)$, respectively. However, it does so at a cost. Specifically, SAMPLEGREEDY applies only to a subclass of k-system constraints known as k-extendible system constraints. We note, however, that the class of k-extendible constraints is still general enough to capture, among others, any constraint that can be represented as the intersection of multiple matroid and matching constraints. Interestingly, when the objective function is also monotone the approximation ratio of SAMPLEGREEDY improves to $k+1$, which matches the best known approximation ratio for the maximization of such functions subject to a k-extendible system constraint (Fisher et al., 1978; Jenkyns, 1976; Mestre, 2006). Previously, this ratio was obtained using the greedy algorithm whose time complexity is $O(nr)$. Hence, our algorithm also improves over the state of the art algorithm for maximization

1. Here, and throughout the paper, n represents the size of the ground set of the submodular function and r represents the maximum size of any set satisfying the constraint.

Table 1: Summary of Results for Non-monotone Submodular Objectives

	Approximation Ratio	**Time Complexity**	**Constraint Class**
SAMPLEGREEDY	$(k+1)^2/k \leq k+3$	$O(n+nr/k)$	k-extendible
REPEATEDGREEDY	$k+O(\sqrt{k})$	$O(nr\sqrt{k})$	k-system
Mirzasoleiman et al. (2016a)	$\approx 2k$	$O(nrk)$	k-system
Gupta et al. (2010)	$\approx 3k$	$O(nrk)$	k-system
Inapproximability	$(1-e^{-k})^{-1}-\varepsilon \geq k+1/2-\varepsilon$	–	k-extendible

of monotone submodular functions subject to a k-extendible system constraint in terms of the time complexity.

We complement our algorithmic results with two inapproximability results showing that the approximation ratios obtained by our second algorithm are almost tight. Previously, it was known that no polynomial time algorithm can have an approximation ratio of $k-\varepsilon$ (for any constant $\varepsilon > 0$) for the problem of maximizing a linear function subject to a k-system constraint (Badanidiyuru and Vondrák, 2014). We show that this result extends also to k-extendible systems, i.e., that no polynomial time algorithm can have an approximation ratio of $k-\epsilon$ for the problem of maximizing a linear function subject to a k-extendible system constraint. Moreover, for monotone submodular functions we manage to get a slightly stronger inapproximability result. Namely, we show that no polynomial time algorithm can have an approximation ratio of $1/(1-e^{-k})-\varepsilon \leq k+1/2-\varepsilon$ for the problem of maximizing a monotone submodular function subject to a k-extendible system constraint. Note that the gap between the approximation ratio obtained by SAMPLEGREEDY (namely, $(k+3)$) and the inapproximability result (namely, $(k+1/2-\varepsilon)$) is very small. A short summary of all the results discussed above for non-monotone submodular objectives can be found in Table 1.

Finally, we compare the performance of REPEATEDGREEDY and SAMPLEGREEDY against FATNOM, the current state of the art algorithm introduced by Mirzasoleiman et al. (2016a). We test these algorithms on a movie recommendation problem using the MovieLens dataset, which consists of over 20 million ratings of 27,000 movies by 138,000 users. We find that our algorithms provide solution sets of similar quality as FANTOM while running orders of magnitude faster. In fact, we observe that taking the best solution found by several independent executions of SAMPLEGREEDY clearly yields the best trade-off between solution quality and computational cost. Moreover, our experimental results indicate that our faster algorithms could be applied to large scale problems previously intractable for other methods.

Organization: Section 2 contains a brief summary of additional related work. A formal presentation of our main results and some necessary preliminaries are given in Section 3. Then, in Section 4 we present and analyze our deterministic and randomized approximation algorithms, respectively. The above mentioned experimental results comparing our algorithms with previously known algorithms can be found in Section 6. Finally, our hardness results appear in Appendix A.

2. Related Works

Submodular maximization has been studied with respect to a few special cases of k-extendible system constraints. Lee et al. (2010) described a local search algorithm achieving an approximation ratio of $k + \epsilon$ for maximizing a monotone submodular function subject to the intersection of any k matroid constraints, and explained how to use multiple executions of this algorithm to get an approximation ratio of $k + 1 + \frac{1}{k+1} + \varepsilon$ for this problem even when the objective function is non-monotone. Later, Feldman et al. (2011b) showed, via a different analysis, that the same local search algorithm can also be used to get essentially the same results for the maximization of a submodular function subject to a subclass of k-extendible system constraints known as k-exchange system constraints (this class of constraints captures the intersection of multiple matching and strongly orderable matroid constraints). For $k \geq 4$, the approximation ratio of (Feldman et al., 2011b) for the case of a monotone submodular objective was later improved by Ward (2012) to $(k + 3)/2 + \varepsilon$.

An even more special case of k-extendible systems are the simple matroid constraints. In their classical work, Nemhauser et al. (1978) showed that the natural greedy algorithm gives an approximation ratio of $1 - 1/e$ for maximizing a monotone submodular function subject to a uniform matroid constraint (also known as cardinality constraint), and Călinescu et al. (2011) later obtained the same approximation ratio for general matroid constraints using the Continuous Greedy algorithm. Moreover, both results are known to be optimal (Nemhauser and Wolsey, 1978). However, optimization guarantees for non-montone submodular maximization are much less well understood. After a long series of works (Vondrák, 2013; Gharan and Vondrák, 2011; Feldman et al., 2011a; Ene and Nguyen, 2016), the current best approximation ratio for maximizing a non-monotone submodular function subject to a matroid constraint is 0.385 (Buchbinder and Feldman, 2016a). In contrast, the state of the art inapproximability result for this problem is 0.478 (Gharan and Vondrák, 2011).

Recently, there has also been a lot of interest in developing fast algorithms for maximizing submodular functions. Badanidiyuru and Vondrák (2014) described algorithms that achieve an approximation ratio of $1-1/e-\varepsilon$ for maximizing a monotone submodular function subject to uniform and general matroid constraints using time complexities of $O_\varepsilon(n \log k)$ and $O_\varepsilon(nr \log^2 n)$, respectively ($O_\varepsilon$ suppresses a polynomial dependence on ε).[2] For uniform matroid constraints, Mirzasoleiman et al. (2015a) showed that one can completely get rid of the dependence on r, and get an algorithm with the same approximation ratio whose time complexity is only $O_\varepsilon(n)$. Independently, Buchbinder et al. (2015) showed that a technique similar to Mirzasoleiman et al. (2015a) can be used to get also $(e^{-1} - \varepsilon)$-approximation for maximizing a *non-monotone* submodular function subject to a cardinality constraint using a time complexity of $O_\varepsilon(n)$. Buchbinder et al. (2015) also described a different $(1 - 1/e - \varepsilon)$-approximation algorithm for maximizing a monotone submodular function subject to a general matroid constraint. The time complexity of this algorithm is $O_\varepsilon(r^2 + n\sqrt{r} \log^2 n)$ for general matroids, and it can be improved to $O_\varepsilon(r\sqrt{n} \log n + n \log^2 n)$ for generalized partition matroids.

3. Preliminaries and Main Results

In this section we formally describe our main results. However, before we can do that, we first need to present some basic definitions and other preliminaries.

2. Badanidiyuru and Vondrák (2014) also describe a fast algorithm for maximizing a monotone submodular function subject to a knapsack constraint. However, the time complexity of this algorithm is exponential in $1/\varepsilon$, and thus, its contribution is mostly theoretical.

3.1. Preliminaries

For a set A and element e, we often write the union $A \cup \{e\}$ as $A + e$ for simplicity. Additionally, we say that f is a *real-valued set function* with *ground set* $\mathcal{N}$ if f assigns a real number to each subset of $\mathcal{N}$. We are now ready to introduce the class of submodular functions.

Definition 1 *Let $\mathcal{N}$ be a finite set. A real-valued set function $f : 2^{\mathcal{N}} \to \mathbb{R}$ is* submodular *if, for all $X, Y \subseteq \mathcal{N}$,*

$$f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y) \ . \tag{1}$$

Equivalently, for all $A \subseteq B \subseteq \mathcal{N}$ and $e \in \mathcal{N} \setminus B$,

$$f(A + e) - f(A) \geq f(B + e) - f(B) \ . \tag{2}$$

While definitions (1) and (2) are equivalent, the latter one, which is known as the *diminishing returns property*, tends to be more helpful and intuitive in most situations. Indeed, the fact that submodular functions capture the notion of diminishing returns is one of the key reasons for the usefulness of submodular maximization in combinatorial optimization and machine learning. Due to the importance and usefulness of the diminishing returns property, it is convenient to define, for every $e \in \mathcal{N}$ and $S \subseteq \mathcal{N}$, $\Delta f(e|S) := f(S + e) - f(S)$. In this paper we only consider the maximization of submodular functions which are *non-negative* (i.e., $f(A) \geq 0$ for all $A \subseteq \mathcal{N}$).[3]

As explained above, we consider submodular maximization subject to two kinds of constraints: k-system and k-extendible system constraints. Both kinds can be cast as special cases of a more general class of constraints known as *independence system* constraints. Formally, an independence system is a pair $(\mathcal{N}, \mathcal{I})$, where $\mathcal{N}$ is a finite set and $\mathcal{I}$ is a non-empty subset of $2^{\mathcal{N}}$ having the property that $A \subseteq B \subseteq \mathcal{N}$ and $B \in \mathcal{I}$ imply together $A \in \mathcal{I}$.

Let us now define some standard terminology related to independence systems. The sets of $\mathcal{I}$ are called the *independent* sets of the independence system. Additionally, an independent set B contained in a subset X of the ground set $\mathcal{N}$ is a *base* of X if no other independent set $A \subseteq X$ strictly contains B. Using this terminology we can now give the formal definition of k-systems.

Definition 2 *An independence system $(\mathcal{N}, \mathcal{I})$ is called a* k-system *if for every set $X \subseteq \mathcal{N}$ the sizes of the bases of X differ by at most a factor of k. More formally, $|B_1|/|B_2| \leq k$ for every two bases B_1, B_2 of X.*

An important special case of k-systems are the k-extendible systems, which were first introduced by Mestre (2006). We say that an independent set B is an *extension* of an independent set A if B strictly contains A.

Definition 3 *An independence system $(\mathcal{N}, \mathcal{I})$ is* k-extendible *if for every independent set $A \in \mathcal{I}$, an extension B of this set and an element $e \notin A$ obeying $A \cup \{e\} \in \mathcal{I}$ there must exist a subset $Y \subseteq B \setminus A$ with $|Y| \leq k$ such that $B \setminus Y \cup \{e\} \in \mathcal{I}$.*

Intuitively, an independence system is k-extendible if adding an element e to an independent set B requires the removal of at most k other elements in order to keep the resulting set independent. As is shown by Mestre (2006), the intersection of k matroids defined on a common ground set is always a k-extendible system. The converse is not generally true (except in the case of $k = 1$, since every 1-system is a matroid).

3. See (Feige et al., 2011) for an explanation why submodular functions that can take negative values cannot be maximized even approximately.

3.2. Main Contributions

Our main contributions in this paper are two efficient algorithms for submodular maximization: one deterministic and one randomized. The following theorem formally describes the properties of our randomized algorithm. Recall that n is the size of the ground set and r is the size of the maximal feasible set.

Theorem 4 *Let $f\colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function, and let $(\mathcal{N},\mathcal{I})$ be a k-extendible system. Then, there exists a randomized $O(n + nr/k)$ time algorithm for maximizing f over $(\mathcal{N},\mathcal{I})$ whose approximation ratio is at most $\frac{(k+1)^2}{k}$. Moreover, if the function f is also monotone, then the approximation ratio of the algorithm improves to $k+1$.*

Our deterministic algorithm uses an algorithm for unconstrained submodular maximization as a subroutine, and its properties depend on the exact properties of this subroutine. Let us denote by α the approximation ratio of this subroutine and by $T(n)$ its time complexity given a ground set of size n. Then, as long as the subroutine is deterministic, our algorithm has the following properties.

Theorem 5 *Let $f\colon 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function, and let $(\mathcal{N},\mathcal{I})$ be a k-system. Then, there exists a deterministic $O((nr+T(r))\sqrt{k})$ time algorithm for maximizing f over $(\mathcal{N},\mathcal{I})$ whose approximation ratio is at most $k + \left(1+\frac{\alpha}{2}\right)\sqrt{k} + 2 + \frac{\alpha}{2} + O(1/\sqrt{k})$.*

Buchdiner et al. (2015) provide a deterministic linear-time algorithm for unconstrained submodular maximization having an approximation ratio of 3. Using this deterministic algorithm as the subroutine, the approximation ratio of the algorithm guaranteed by Theorem 5 becomes $k + \frac{5}{2}\sqrt{k} + \frac{7}{2} + O(1/\sqrt{k})$ and its time complexity becomes $O(nr\sqrt{k})$. We note that Buchbinder and Feldman (2016b) recently came up with a deterministic algorithm for unconstrained submodular maximization having an optimal approximation ratio of 2. Using this algorithm instead of the deterministic algorithm of Buchdiner et al. (2015) could marginally improve the approximation ratio guaranteed by Theorem 5. However, the algorithm of Buchbinder and Feldman (2016a) has a quadratic time complexity, and thus, it is less practical. It is also worth noting that Buchdiner et al. (2015) also describe a randomized linear-time algorithm for unconstrained submodular maximization which again achieves the optimal approximation ratio of 2. It turns out that one can get a randomized algorithm whose approximation ratio is $k + 2\sqrt{k} + 3 + O(1/\sqrt{k})$ by plugging the randomized algorithm of (Buchdiner et al., 2015) as a subroutine into the algorithm guaranteed by Theorem 5. However, the obtained randomized algorithm is only useful for the rare case of a k-system constraint which is not also a k-extendible system constraint since Theorem 4 already provides a randomized algorithm with a better guarantee for constraints of the later kind.

We end this section with our inapproximability results for maximizing linear and submodular functions over k-extendible systems. Recall that these inapproximability results nearly match the approximation ratio of the algorithm given by Theorem 4.

Theorem 6 *There is no polynomial time algorithm for maximizing a linear function over a k-extendible system that achieves an approximation ratio of $k - \varepsilon$ for any constant $\varepsilon > 0$.*

Theorem 7 *There is no polynomial time algorithm for maximizing a non-negative monotone submodular function over a k-extendible system that achieves an approximation ratio of $(1-e^{-1/k})^{-1} - \varepsilon$ for any constant $\varepsilon > 0$.*

We note that the inapproximability results given by the last two theorems apply also to a fractional relaxation of the corresponding problems known as the multilinear relaxation. This observation has two implications. The first of these implications is that even if we were willing to settle for a fractional solution, still we could not get a better than $(1/(1-e^{-k}))$-approximation for maximizing a monotone submodular function subject to a k-extendible system constraint. Interestingly, this approximation ratio can in fact be reached in the fractional case even for k-system constraints using the well known (computationally heavy) Continuous Greedy algorithm of Călinescu et al. (2011). Thus, we get the second implication of the above observation, which is that further improving our inapproximability results requires the use of new techniques since the current technique cannot lead to different results for the fractional and non-fractional problems.

4. Repeated Greedy: An Efficient Deterministic Algorithm

In this section, we present and analyze the deterministic algorithm for maximizing a submodular function f subject to a k-system constraint whose existence is guaranteed by Theorem 5. Our algorithm works in iterations, and in each iteration it makes three operations: executing the greedy algorithm to produce a feasible set, executing a deterministic unconstrained submodular maximization algorithm on the output set of the greedy algorithm to produce a second feasible set, and removing the elements of the set produced by the greedy algorithm from the ground set. After the algorithm makes ℓ iterations of this kind, where ℓ is a parameter to be determined later, the algorithm terminates and outputs the best set among all the feasible sets encountered during its iterations. A more formal statement of this algorithm is given as Algorithm 1.

Algorithm 1: Repeated Greedy($\mathcal{N}, f, \mathcal{I}, \ell$)

Let $\mathcal{N}_1 \leftarrow \mathcal{N}$.
for $i = 1$ **to** ℓ **do**
  Let S_i be the output of the greedy algorithm given $\mathcal{N}_i$ as the ground set, f as the objective and $\mathcal{I}$ as the constraint.
  Let S'_i be the output of a deterministic algorithm for unconstrained submodular maximization given S_i as the ground set and f as the objective.
  Let $\mathcal{N}_{i+1} \leftarrow \mathcal{N}_i \setminus S_i$.
end
return the set T maximizing f among the sets $\{S_i, S'_i\}_{i=1}^{\ell}$.

Observation 8 *The set T returned by Algorithm 1 is independent.*

Proof For every $1 \leq i \leq \ell$, the set S_i is independent because the greedy algorithm returns an independent set. Moreover, the set S'_i is also independent since $(\mathcal{N}, \mathcal{I})$ is an independence system and the algorithm for unconstrained maximization must return a subset of its independent ground set S_i. The observation now follows since T is chosen as one of these sets. ■

We now begin the analysis of the approximation ratio of Algorithm 1. Let OPT be an independent set of $(\mathcal{N}, \mathcal{I})$ maximizing f, and let α be the approximation ratio of the unconstrained submodular maximization algorithm used by Algorithm 1. The analysis is based on three lemmata.

The first of these lemmata states properties of the sets S_i and S'_i which follow immediately from the definition of α and known results about the greedy algorithm. Due to space constraints, the proofs of all three lemmata have been deferred to Appendix B.

Lemma 9 *For every* $1 \leq i \leq \ell$, $f(S_i) \geq \frac{1}{k+1} f(S_i \cup (OPT \cap \mathcal{N}_i))$ *and* $f(S'_i) \geq f(S_i \cap OPT)/\alpha$.

The next lemma shows that the average value of the union between a set from $\{S_i\}_{i=1}^{\ell}$ and OPT must be quite large. Intuitively this follows from the fact that these sets are disjoint, and thus, every "bad" element which decreases the value of OPT can appear only in one of them.

Lemma 10 $\sum_{i=1}^{\ell} f(S_i \cup OPT) \geq (\ell - 1) f(OPT)$.

The final lemma we need is the following basic fact about submodular functions.

Lemma 11 *Suppose* f *is a non-negative submodular function over ground set* $\mathcal{N}$. *For every three sets* $A, B, C \subseteq \mathcal{N}$, $f(A \cup (B \cap C)) + f(B \setminus C) \geq f(A \cup B)$.

Having the above three lemmata, we are now ready to prove Theorem 5.

Proof [Proof of Theorem 5] Observe that, for every $1 \leq i \leq \ell$, we have

$$\text{OPT} \setminus \mathcal{N}_i = \text{OPT} \cap (\mathcal{N} \setminus \mathcal{N}_i) = \text{OPT} \cap \left(\cup_{j=1}^{i-1} S_i\right) = \cup_{j=1}^{i-1} (\text{OPT} \cap S_j) \tag{3}$$

where the first equality holds because $\text{OPT} \subseteq \mathcal{N}$ and the second equality follows from the removal of S_i from the ground set in each iteration of Algorithm 1. Using the previous lemmata and this observation, we get

$$\begin{aligned}
(\ell - 1) f(\text{OPT}) &\leq \sum_{i=1}^{\ell} f(S_i \cup \text{OPT}) && \text{(Lemma 10)} \\
&\leq \sum_{i=1}^{\ell} f(S_i \cup (\text{OPT} \cap \mathcal{N}_i)) + \sum_{i=1}^{\ell} f(\text{OPT} \setminus \mathcal{N}_i) && \text{(Lemma 11)} \\
&= \sum_{i=1}^{\ell} f(S_i \cup (\text{OPT} \cap \mathcal{N}_i)) + \sum_{i=1}^{\ell} f\left(\cup_{j=1}^{i-1} (\text{OPT} \cap S_j)\right) && \text{(Equality (3))} \\
&\leq \sum_{i=1}^{\ell} f(S_i \cup (\text{OPT} \cap \mathcal{N}_i)) + \sum_{i=1}^{\ell} \sum_{j=1}^{i-1} f(\text{OPT} \cap S_j) && \text{(submodularity)} \\
&\leq (k+1) \sum_{i=1}^{\ell} f(S_i) + \alpha \sum_{i=1}^{\ell} \sum_{j=1}^{i-1} f(S'_j) && \text{(Lemma 9)} \\
&\leq (k+1) \sum_{i=1}^{\ell} f(T) + \alpha \sum_{i=1}^{\ell} \sum_{j=1}^{i-1} f(T) && (T\text{'s definition}) \\
&= \left[(k+1)\ell + \alpha \ell(\ell-1)/2\right] f(T) \ .
\end{aligned}$$

Dividing the last inequality by $(k+1)\ell + \alpha\ell(\ell-1)/2$, we get

$$f(T) \geq \frac{\ell - 1}{(k+1)\ell + \frac{\alpha}{2}\ell(\ell-1)} f(\text{OPT}) = \frac{1 - \frac{1}{\ell}}{k + \frac{\alpha}{2}\ell + 1 - \frac{\alpha}{2}} f(\text{OPT}) \ . \tag{4}$$

The last inequality shows that the approximation ratio of Algorithm 1 is at most $\frac{k+\frac{\alpha}{2}\ell+1-\frac{\alpha}{2}}{1-\frac{1}{\ell}}$. To prove the theorem it remains to show that, for an appropriate choice of ℓ, this ratio is at most $k+\left(1+\frac{\alpha}{2}\right)\sqrt{k}+2+\frac{\alpha}{2}+O(1/\sqrt{k})$. It turns out that the right value of ℓ for us is $\lceil\sqrt{k}\rceil$. The theorem follows by plugging this value into Inequality (4). See Appendix B for the exact calculations. ■

5. Sample Greedy: An Efficient Randomized Algorithm

In this section, we present and analyze a randomized algorithm for maximizing a submodular function f subject to a k-extendible system constraint. Our algorithm is very simple: it first samples elements from $\mathcal{N}$, and then runs the greedy algorithm on the sampled set. This algorithm is outlined as Algorithm 2.

Algorithm 2: Sample Greedy($\mathcal{N}, f, \mathcal{I}, k$)

Let $\mathcal{N}' \leftarrow \varnothing$ and $S \leftarrow \varnothing$.
for $u \in \mathcal{N}$ **do**
  with *probability* $(k+1)^{-1}$ **do**
    Add u to $\mathcal{N}'$.
end
while *there exists* $u \in \mathcal{N}'$ *such that* $S + u \in \mathcal{I}$ *and* $\Delta f(u|S) > 0$ **do**
  Let $u \in \mathcal{N}'$ be the element of this kind maximizing $\Delta f(u|S)$.
  Add u to S.
end
return S.

Algorithm 3: Equivalent Algorithm($\mathcal{N}, f, \mathcal{I}, k$)

Let $\mathcal{N}' \leftarrow \mathcal{N}$, $S \leftarrow \varnothing$ and $O \leftarrow OPT$.
while *there exists an element* $u \in \mathcal{N}'$ *such that* $S + u \in \mathcal{I}$ *and* $\Delta f(u|S) > 0$ **do**
  Let $u \in \mathcal{N}'$ be the element of this kind maximizing $\Delta f(u|S)$, and let $S_u \leftarrow S$.
  with *probability* $(k+1)^{-1}$ **do**
    Add u to S and O.
    Let $O_u \subseteq O \setminus S$ be the smallest set such that $O \setminus O_u \in \mathcal{I}$.
  otherwise
    if $u \in O$ **then** Let $O_u \leftarrow \{u\}$.
    else Let $O_u \leftarrow \varnothing$.
  Remove the elements of O_u from O.
  Remove u from $\mathcal{N}'$.
end
return S.

To better analyze Algorithm 2, we introduce an auxiliary algorithm given as Algorithm 3. It is not difficult to see that both algorithms have identical output distributions. The sampling of elements in Algorithm 2 is independent of the greedy maximization, so interchanging these two steps does not affect the output distribution. Moreover, the variables S_u, O and O_u in Algorithm 3 do not affect the output S (and in fact, appear only for analysis purposes). Thus, the two algorithms are equivalent, and any approximation guarantee we can prove for Algorithm 3 immediately carries over to Algorithm 2.

Next, we are going to analyze Algorithm 3. However, before doing it, let us intuitively explain the roles of the sets S, S_u, O and O_u in this algorithm. We say that Algorithm 3 considers an element u in some iteration if u is the element chosen as maximizing $\Delta f(u|S)$ at the beginning of this iteration. Note that an element is considered at most once, and perhaps not at all. As in

Algorithm 2, S is the current solution. Likewise, S_u is the current solution S at the beginning of the iteration in which Algorithm 3 considers u. The set O maintained in Algorithm 3 is an independent set which starts as OPT and changes over time, while preserving three properties:

P1 O is an independent set.

P2 Every element of S is an element of O.

P3 Every element of $O \setminus S$ is an element not yet considered by Algorithm 3.

Because these properties need to be maintained throughout the excecution, some elements may be removed from O in every given iteration. The set O_u is simply the set of elements removed from O in the iteration in which u is considered. Note that O_u and S_u are random sets, and Algorithm 3 defines values for them if it considers u at some point. In the analysis below we assume that both S_u and O_u are empty if u is not considered by Algorithm 3 (which means that Algorithm 3 does not explicitly set values for them).

We now explain how the algorithm manages to maintain the above mentioned properties of O. Let us begin with properties **P1** and **P2**. Clearly the removal of elements from O that do not belong to S cannot violate these properties, thus, we only need to consider the case that the algorithm adds an element u to S in some iteration. To maintain **P2**, u is also added to O. On the one hand, $O + u$ might not be independent, and thus, the addition of u to O might violate **P1**. However, since O is an extension of S by **P2**, $S + u$ is independent by the choice of u and $(\mathcal{N}, \mathcal{I})$ is a k-extendible system, the algorithm is able to choose a set $O_u \subseteq O \setminus S$ of size at most k whose removal restores the independence of $O + u$, and thus, also **P1**. It remains to see why the algorithm preserves also **P3**. Since the algorithm never removes from O an element of S, a violation of **P3** can only occur when the algorithm considers some element of $O \setminus S$. However, following this consideration one of two things must happen. Either u is also added to S, or u is placed in O_u and removed from O. In either case **P3** is restored.

From this point on, every expression involving S or O is assumed to refer to the final values of these sets. The following lemma provides a lower bound on $f(S)$ that holds deterministically. Intuitively, this lemma follows from the observation that, when an element u is considered by Algorithm 3, its marginal contribution is at least as large as the marginal contribution of any element of $\text{OPT} \setminus S$. Due to space constraints, many proofs in this section have been deferred to Appendix C.

Lemma 12 $f(S) \geq f(S \cup OPT) - \sum_{u \in \mathcal{N}} |O_u \setminus S| \Delta f(u|S_u)$.

While the previous lemma was true deterministically, the next two lemmas are statements about expected values. At this point, it is convenient to define a new random variable. For every element $u \in \mathcal{N}$, let X_u be an indicator for the event that u is considered by Algorithm 3 in one of its iterations. The next lemma gives an expression for the expected value of S. It follows quite easily from the linearity of expectation.

Lemma 13 $\mathbb{E}[f(S)] \geq \frac{1}{k+1} \sum_{u \in \mathcal{N}} \mathbb{E}[X_u \Delta f(u|S_u)]$.

The next lemma relates terms appearing in the last two lemmata. Intuitively, this lemma shows that O_u is on average a small set. For elements of OPT this is true since their O_u set never contains more than one element, and for other elements this is true since their O_u set is empty whenever they are not added to S.

Lemma 14 *For every element $u \in \mathcal{N}$, $\mathbb{E}[|O_u \setminus S|\Delta f(u|S_u)] \leq \frac{k}{k+1}\mathbb{E}[X_u \Delta f(u|S_u)]$.*

We are now ready to prove Theorem 4.

Proof [Proof of Theorem 4] Our objective is to prove approximation ratios for Algorithm 2. As discussed earlier, Algorithms 2 and 3 have identical output distributions, and so it suffices to show that Algorithm 3 achieves the desired approximation approximation ratios. Note that

$$\begin{aligned}
\mathbb{E}[f(S)] &\geq \mathbb{E}[f(S \cup \text{OPT})] - \sum_{u \in \mathcal{N}} \mathbb{E}[|O_u \setminus S|\Delta f(u|S_u)] && \text{(Lemma 12)} \\
&\geq \mathbb{E}[f(S \cup \text{OPT})] - \frac{k}{k+1} \sum_{u \in \mathcal{N}} \mathbb{E}[X_u \Delta f(u|S_u)] && \text{(Lemma 14)} \\
&\geq \mathbb{E}[f(S \cup \text{OPT})] - k\mathbb{E}[f(S)] \ . && \text{(Lemma 13)}
\end{aligned}$$

If f is monotone, then the theorem follows by rearranging the above inequality and observing that $f(S \cup OPT) \geq f(OPT)$. Otherwise, the fact that S contains every element with probability at most $(k+1)^{-1}$ can be used to show via known results (see Appendix C for details) that

$$\mathbb{E}[f(S \cup \text{OPT})] \geq \frac{k}{k+1} f(\text{OPT}) \ .$$

The theorem now follows by combining the two above inequalities, and rearranging. ■

6. Experimental Results

In this section, we present results of an experiment on real dataset comparing the performance of REPEATEDGREEDY and SAMPLEGREEDY with other competitive algorithms. In particular, we compare our algorithms to the greedy algorithm and FANTOM, an $O(krn)$-time algorithm for nonmonotone submodular optimization introduced in (Mirzasoleiman et al., 2016a). We also boost SAMPLEGREEDY by taking the best of four runs, denoted Max Sample Greedy. We test these algorithms on a personalized movie recommendation system, and find that while REPEATEDGREEDY and SAMPLEGREEDY return comparable solutions to FANTOM, they run orders of magnitude faster. These initial results indicate that SAMPLEGREEDY may be applied (without any loss in performance) to massive problem instances that were previously intractable.

In the movie recommendation system application, we observe movie ratings from users, and our objective is to recommend movies to users based on their reported favorite genres. In particular, given a user-specified input of favorite genres, we would like to recommend a short list of movies that are diverse, and yet representative, of those genres. The similarity score between movies that we use is derived from user ratings, as in (Lindgren et al., 2015).

Let us now describe the problem setting in more detail. Let $\mathcal{N}$ be a set of movies, and G be the set of all movie genres. For a movie $i \in \mathcal{N}$, we denote the set of genres of i by $G(i)$ (each movie may have multiple genres). Similarly, for a genre $g \in G$, denote the set of all movies in that genre by $\mathcal{N}(g)$. Let $s_{i,j}$ be a non-negative similarity score between movies $i, j \in \mathcal{N}$, and suppose a user u seeks a representative set of movies from genres $G_u \subseteq G$. Note that the set of movies from

these genres is $\mathcal{N}_u = \cup_{g \in G_u} \mathcal{N}(g)$. Thus, a reasonable utility function for choosing a diverse yet representative set of movies S for u is

$$f_u(S) = \sum_{i \in S} \sum_{j \in \mathcal{N}_u} s_{i,j} - \lambda \sum_{i \in S} \sum_{j \in S} s_{i,j} \tag{5}$$

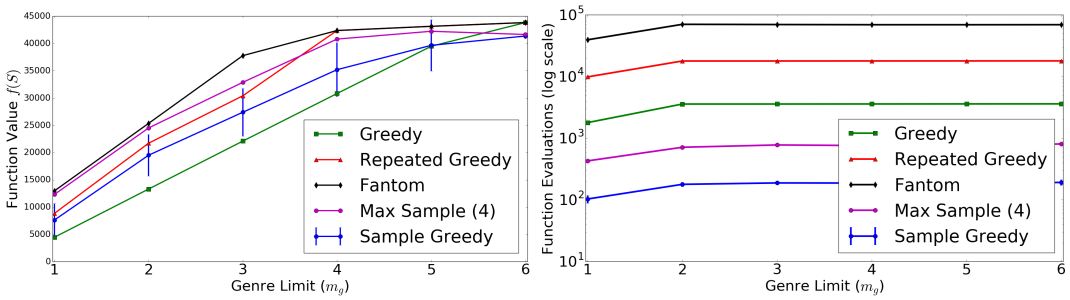
for some parameter $0 \leq \lambda \leq 1$. Observe that the first term is a sum-coverage function that captures the representativeness of S, and the second term is a dispersion function penalizing similarity within S. Moreover, for $\lambda = 1$ this utility function reduces to the simple cut function.

The user may specify an upper limit m on the number of movies in his recommended set. In addition, he is also allowed to specify an upper limit m_g on the number of movies from genre g in the set for each $g \in G_u$ (we call the parameter m_g a *genre limit*). The first constraint corresponds to an m-uniform matroid over $\mathcal{N}_u$, while the second constraint corresponds to the intersection of $|G_u|$ partition matroids (each imposing the genre limit of one genre). Thus, our constraints in this movie recommendation corresponds to the intersection of $1 + |G_u|$ matroids, which is a $(1 + |G_u|)$-extendible system (in fact, a more careful analysis shows that it corresponds to a $|G_u|$-extendible system).

For our experiments, we use the MovieLens 20M dataset, which features 20 million ratings of 27,000 movies by 138,000 users. To obtain a similarity score between movies, we take an approach developed in (Lindgren et al., 2015). First, we fill missing entries of an incomplete movie-user matrix $M \in \mathbb{R}^{n \times m}$ via low-rank matrix completion (Candés and Recht, 2008; Hastie et al., 2015), then we randomly sample to obtain a matrix $\tilde{M} \in \mathbb{R}^{n \times \ell}$ where $\ell \ll m$ and the inner products between rows are preserved. The similarity score between movies i and j is then defined as the inner product of their corresponding rows in $\tilde{M}$. In our experiment, we set the total recommended movies limit to $m = 10$. The genre limits m_g are always equal for all genres, and we vary them from 1 to 9. Finally, we set our favorite genres G_u as Adventure, Animation and Fantasy. As noted above, this means that our constraint set forms a 3-extendible system. For each algorithm and test instance, we record the function value of the returned solution S and the number of calls to f, which is a machine-independent measure of run-time.

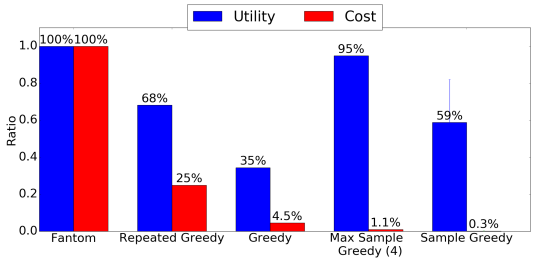
Figure 1(*a*) shows the value of the solution sets for the various algorithms. As we see from Figure 1(*a*), FANTOM consistently returns a solution set with the highest function value; however, REPEATEDGREEDY and SAMPLEGREEDY return solution sets with similarly high function values. We see that Max Sample Greedy, even for four runs, significantly increases the performance for more constrained problems. Note that for such more constrained problems, Greedy returns solution sets with much lower function values. Figure 1(*b*) shows the number of function calls made by each algorithm as the genre limit m_g is varied. For each algorithm, the number of function calls remains roughly constant as m_g is varied—this is due to the lazy greedy implementation that takes advantage of submodularity to reduce the number of function calls. We see that for our problem instances, REPEATEDGREEDY runs about an order of magnitude faster than FANTOM and SAMPLEGREEDY runs roughly three orders of magnitude faster than FANTOM. Moreover, boosting SAMPLEGREEDY by executing it a few times does not incur a significant increase in cost.

To better analyze the tradeoff between the utility of the solution value and the cost of run time, we compare the ratio of these measurements for the various algorithms using FANTOM as a baseline. See Figure 1(*c*) and 1(*d*) for these ratio comparisons for genre limits $m_g = 1$ and $m_g = 4$, respectively. For the case of $m_g = 1$, we see that boosted SAMPLEGREEDY provides nearly the same utility as FANTOM, while only incurring 1.09% of the computational cost. Likewise, for the

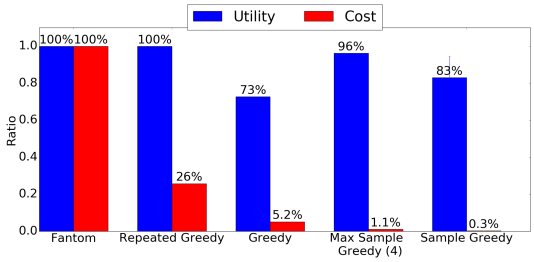


(a) Solution Quality

(b) Run Time



(c) Ratio Comparison $m_g = 1$



(d) Ratio Comparison $m_g = 4$

Figure 1: Performance Comparison. 1(a) shows the function value of the returned solutions for tested algorithms with varying genre limit m_g. 1(b) shows the number of function evaluations on a logarithmic scale with varying genre limit m_g. 1(c) and 1(d) show the ratio of solution quality and cost with FANTOM as a baseline.

case of $m_g = 4$, REPEATEDGREEDY achieves the same utility as FANTOM, while incurring only a quarter of the cost. Thus, we may conclude that our algorithms provide solutions whose quality is on par with current state of the art, and yet they run in a small fraction of the time.

Figure 2: Solution Sets. The movies in the solution sets for $m_g = 1$ returned by FANTOM, Sample Greedy, Repeated Greedy and Greedy are listed here, along with genre information. The favorite genres (G_u) are in red.

While Greedy may get stuck in poor locally optimal solutions, REPEATEDGREEDY and SAMPLEGREEDY avoid this by greedily combing through the solution space many times and selecting random sets, respectively. Fortunately, the movie recommendation system has a very interpretable

solution so we can observe this phenomenon. See Figure 2 for the movies recommended by the different algorithms. Because $m_g = 1$, we are constrained here to have at most one movie from Adventure, Animation and Fantasy. As seen in Figure 2, FANTOM and SAMPLEGREEDY return maximum size solution sets that are both diverse and representative of these genres. On the other hand, Greedy gets stuck choosing a single movie that belongs to all three genres, thus, precluding any other choice of movie from the solution set.

Finally, we would like to make a short remark on the relation between our theoretical analysis and experimental results. In our analysis, we showed that SAMPLEGREEDY achieves a better expected approximation ratio than the worst-case approximation ratio of REPEATEDGREEDY and FANTOM. However, there are many problem instances which are "not hard", and do not make the algorithms exhibit their worst-case behavior. For such instances, REPEATEDGREEDY and FANTOM may out-perform SAMPLEGREEDY. Our experiments demonstrate that even in usual "not hard" problem instances, SAMPLEGREEDY performs competitively at a fraction of the computational cost, indicating its potential for real-world large scale applications.

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. (DGE1122492), DARPA Young Faculty Award (D16AP00046), Simons-Berkeley Fellowship, and Israel Science Foundation (ISF grant number 1357/16). The authors would like to thank Eric Lindgren for kindly sharing processed movieLens data.

References

Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *SODA*, 2014.

Niv Buchbinder and Moran Feldman. Constrained submodular maximization via a non-symmetric technique. *CoRR*, abs/1611.03253, 2016a.

Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. In *SODA*, 2016b.

Niv Buchbinder, Moran Feldman, Joesph (Seffi) Naor, and Roy Schwartz. Submodular maximization with cardinality constrains. In *SODA*, 2014.

Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing apples and oranges: Query tradeoff in submodular maximization. In *SODA*, 2015.

N. Buchdiner, M. Feldman, N.S. Joseph, and R. Schwartz. A tight linear time (1/2)-approximatoin for unconstrained submodular maximization. *SIAM Journal on Computing*, 2015.

Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 2011.

E. Candés and B. Recht. Exact matrix completion via convex optimization. In *Foundations of Computational Mathematics*, 2008.

V. Chvátal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25(3):285–287, 1979.

Jack Edmonds. Matroids and the greedy algorithm. *Mathematical programming*, 1971.

Khalid El-Arini, Gaurav Veda, Dafna Shahaf, and Carlos Guestrin. Turning down the noise in the blogosphere. In *KDD*, 2009.

Alina Ene and Huy L. Nguyen. Constrained submodular maximization: Beyond 1/e. In *FOCS*, 2016.

Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 2011.

Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *FOCS*, 2011a.

Moran Feldman, Joseph Naor, Roy Schwartz, and Justin Ward. Improved approximations for k-exchange systems - (extended abstract). In *ESA*, 2011b.

M. Fisher, G. Nemhauser, and L. Wolsey. An analysis of approximations for maximizing submodular set functions—ii. *Mathematical Programming*, 8:73–87, 1978.

Satoru Fujishige. *Submodular functions and optimization*. Elsevier Science, 2nd edition, 2005.

Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *SODA*, 2011.

Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Near-optimal MAP inference for determinantal point processes. In *NIPS*, 2012.

Manuel Gomez Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. In *KDD*, 2010.

Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in gaussian processes. In *ICML*, 2005.

Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *WINE*, 2010.

Trevor Hastie, Rahul Mazumder, Jason D. Lee, and Rexa Zadeh. Matrix completion and low-rank svd via fast alternating least squares. In *Journal of Machine Learning Research*, 2015.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 1963.

T. A. Jenkyns. The efficacy of the "greedy" algorithm. In *South Eastern Conference on Combinatorics, Graph Theory and Computing*, 1976.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.

Katrin Kirchhoff and Jeff Bilmes. Submodularity for data selection in statistical machine translation. In *EMNLP*, 2014.

Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research*, 2010.

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne Van Briesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *KDD*, 2007.

Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *ACL*, 2011.

Erik M Lindgren, Shanshan Wu, and Alexandros G Dimakis. Sparse and greedy: Sparsifying submodular facility location problems. In *NIPS Workshop on Optimization for Machine Learning*, 2015.

Julián Mestre. Greedy in approximation algorithms. In *ESA*, 2006.

Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *NIPS*, 2013.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrak, and Andreas Krause. Lazier than lazy greedy. In *AAAI*, 2015a.

Baharan Mirzasoleiman, Amin Karbasi, Ashwinkumar Badanidiyuru, and Andreas Krause. Distributed submodular cover: Succinctly summarizing massive data. In *NIPS*, 2015b.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *ICML*, 2016a.

Baharan Mirzasoleiman, Morteza Zadimoghaddam, and Amin Karbasi. Fast distributed submodular cover: Public-private data summarization. In *Advances in Neural Information Processing Systems*, pages 3594–3602, 2016b.

G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research*, 1978.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 1978.

Colorado Reed and Zoubin Ghahramani. Scaling the indian buffet process via submodular maximization. In *ICML*, 2013.

Adish Singla, Ilija Bogunovic, Gábor Bartók, Amin Karbasi, and Andreas Krause. Near-optimally teaching the crowd to classify. In *ICML*, 2014.

Ruben Sipos, Adith Swaminathan, Pannaga Shivaswamy, and Thorsten Joachims. Temporal corpus summarization using submodular word coverage. In *CIKM*, 2012.

Matthew Skala. Hypergeometric tail inequalities: ending the insanity. *CoRR*, abs/1311.5939, 2013.

Jan Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM Journal on Computing*, 2013.

Justin Ward. A (k+3)/2-approximation algorithm for monotone submodular k-set packing and general k-exchange systems. In *STACS*, 2012.

Appendix A. Hardness of Maximization over k-Extendible Systems

In this appendix, we prove Theorems 6 and 7. The proof consists of two steps. In the first step, we will define two k-extendible systems which are indistinguishable in polynomial time. The inapproximability result for linear objectives will follow from the indistinguishability of these systems and the fact that the size of their maximal sets are very different. In the second step we will define monotone submodular objective functions for the two k-extendible systems. Using the symmetry gap technique of (Vondrák, 2013), we will show that these objective functions are also indistinguishable, despite being different. Then, we will use the differences between the objective functions to prove the slightly stronger inapproximability result for monotone submodular objectives.

Given three positive integers k, h and m such that h is an integer multiple of $2k$, let us construct a k-extendible system $\mathcal{M}(k,h,m) = (\mathcal{N}_{k.h,m}, \mathcal{I}_{k,h,m})$ as follows. The ground set of the system is $\mathcal{N}_{k,h,m} = \cup_{i=1}^{h} H_i(k,m)$, where $H_i(k,m) = \{u_{i,j} \mid 1 \leq j \leq km\}$. A set $S \subseteq \mathcal{N}_{k,h,m}$ is independent (i.e., belongs to $\mathcal{I}_{k,h,m}$) if and only if it obeys the following inequality:

$$g_{m,k}(|S \cap H_1(k,m)|) + |S \setminus H_1(k,m)| \leq m \ ,$$

where the function $g_{m,k}$ is defined by

$$g_{m,k}(x) = \min\left\{x, \frac{2km}{h}\right\} + \max\left\{\frac{x - 2km/h}{k}, 0\right\} \ .$$

Intuitively, a set is independent if its elements do not take too many "resources", where most elements requires a unit of resources, but elements of $H_1(k,m)$ take only $1/k$ unit of resources each once there are enough of them. Consequently, the only way to get a large independent set is to pack many $H_1(k,m)$ elements.

Lemma 15 *For every choice of h and m, $\mathcal{M}(k,h,m)$ is a k-extendible system.*

Proof First, observe that $g(x)$ is a monotone function, and therefore, a subset of an independent set of $\mathcal{M}(k,h,m)$ is also independent. Also, $g(0) = 0$, and therefore, $\varnothing \in \mathcal{I}_{k,h,m}$. This proves that $\mathcal{M}(k,h,m)$ is an independence system. In the rest of the proof we show that it is also k-extendible.

Consider an arbitrary independent set $C \in \mathcal{I}_{k,h,m}$, an independent extension D of C and an element $u \notin D$ for which $C + u \in \mathcal{I}_{k,h,m}$. We need to find a subset $Y \subseteq D \setminus C$ of size at most k such that $D \setminus Y + u \in \mathcal{I}_{k,h,m}$. If $|D \setminus C| \leq k$, then we can simply pick $Y = D \setminus C$. Thus, we can assume from now on that $|D \setminus C| > k$.

Let $\Sigma(S) = g(|S \cap H_1(k,m)|) + |S \setminus H_1(k,m)|$. By definition, $\Sigma(D) \leq m$ because $D \in \mathcal{I}_{k,h,m}$. Observe that $g(x)$ has the property that for every $x \geq 0$, $k^{-1} \leq g(x+1) - g(x) \leq 1$. Thus, $\Sigma(S)$ increases by at most 1 every time that we add an element to S, but decreases by at least $1/k$ every

time that we remove an element from S. Hence, if we let Y be an arbitrary subset of $D \setminus C$ of size k, then

$$\Sigma(D \setminus Y + u) \leq \Sigma(D) - \frac{|Y|}{k} + 1 = \Sigma(D) \leq m \ ,$$

which implies that $D \setminus Y + u \in \mathcal{I}_{k,h,m}$. ■

Before presenting the second k-extendible system, let us show that $\mathcal{M}(k,h,m)$ contains a large independent set.

Observation 16 *$\mathcal{M}(k,h,m)$ contains an independent set whose size is $k(m-2km/h)+2km/h \geq mk(1-2k/h)$. Moreover, there is such set in which all elements belong to $H_1(k,m)$.*

Proof Let $s = k(m-2km/h)+2km/h$, and consider the set $S = \{u_{1,j} \mid 1 \leq j \leq s\}$. This is a subset of $H_1(k,m) \subseteq \mathcal{N}_{k,h,m}$ since $s \leq km$. Also,

$$\begin{aligned} g(|S|) = g(s) &= \min\left\{s, \frac{2km}{h}\right\} + \max\left\{\frac{s-2km/h}{k}, 0\right\} \\ &\leq \frac{2km}{h} + \max\left\{\frac{[k(m-2km/h)+2km/h]-2km/h}{k}, 0\right\} \\ &= \frac{2km}{h} + \max\left\{m - \frac{2km}{h}, 0\right\} = m \ . \end{aligned}$$

Since S contains only elements of $H_1(k,m)$, its independence follows from the above inequality. ■

Let us now define our second k-extendible system $\mathcal{M}'(k,h,m) = (\mathcal{N}_{k,h,n}, \mathcal{I}'_{k,n})$. The ground set of this system is the same as the ground set of $\mathcal{M}(k,h,m)$, but a set $S \subseteq \mathcal{N}_{k,h,m}$ is considered independent in this independence system if and only if its size is at most m. Clearly, this is a k-extendible system (in fact, it is a uniform matroid). Moreover, note that the ratio between the sizes of the maximal sets in $\mathcal{M}(k,h,m)$ and $\mathcal{M}'(k,h,m)$ is at least

$$\frac{mk(1-2k/h)}{m} = k(1-2k/h) \ .$$

Our plan is to show that it takes exponential time to distinguish between the systems $\mathcal{M}(k,h,m)$ and $\mathcal{M}'(k,h,m)$, and thus, no polynomial time algorithm can provide an approximation ratio better than this ratio for the problem of maximizing the cardinality function (i.e., the function $f(S) = |S|$) subject to a k-extendible system constraint.

Consider a polynomial time deterministic algorithm that gets either $\mathcal{M}_{k,h,m}$ or $\mathcal{M}'_{k,h,m}$ after a random permutation was applied to the ground set. We will prove that with high probability the algorithm fails to distinguish between the two possible inputs. Notice that by Yao's lemma, this implies that for every random algorithm there exists a permutation for which the algorithms fails with high probability to distinguish between the inputs.

Assuming our deterministic algorithm gets $\mathcal{M}'_{k,h,m}$, it checks the independence of a polynomial collection of sets. Observe that the sets in this collection do not depend on the permutation because the independence of a set in $\mathcal{M}'_{k,h,m}$ depends only on its size, and thus, the algorithm will take the same execution path given every permutation. If the same algorithm now gets $\mathcal{M}_{k,h,m}$ instead, it

will start checking the independence of the same sets until it will either get a different answer for one of the checks (different than what is expected for $\mathcal{M}'_{k,h,m}$) or it will finish all the checks. Note that in the later case the algorithm must return the same answer that it would have returned had it been given $\mathcal{M}'_{k,h,m}$. Thus, it is enough to upper bound the probability that any given check made by the algorithm will result in a different answer given the inputs $\mathcal{M}_{k,h,m}$ and $\mathcal{M}'_{k,h,m}$.

Lemma 17 *Following the application of the random ground set permutation, the probability that a set S is independent in $\mathcal{M}_{k,h,m}$ but not in $\mathcal{M}'_{k,h,m}$, or vice versa, is at most $e^{-\frac{2km}{h^2}}$.*

Proof Observe that as long as we consider a single set, applying the permutation to the ground set is equivalent to replacing S with a random set of the same size. So, we are interested in the independence in $\mathcal{M}_{k,h,m}$ and $\mathcal{M}'_{k,h,m}$ of a random set of size $|S|$. If $|S| > km$, then the set is never independent in either $\mathcal{M}_{k,h,m}$ or $\mathcal{M}'_{k,h,m}$, and if $|S| \leq m$, then the set is always independent in both $\mathcal{M}_{k,h,m}$ and $\mathcal{M}'_{k,h,m}$. Thus, the interesting case is when $m < |S| \leq km$.

Let $X = |S \cap H_1(k,m)|$. Notice that X has a hypergeometric distribution, and $\mathbb{E}[X] = |S|/h$. Thus, using bounds given in (Skala, 2013) (these bounds are based on results of (Chvátal, 1979; Hoeffding, 1963)), we get

$$\Pr\left[X \geq \frac{2km}{h}\right] = \Pr\left[X \geq \mathbb{E}[|X|] + \frac{km}{h}\right] \leq e^{-2\left(\frac{km/h}{|S|}\right)^2 \cdot |S|} = e^{-\frac{2k^2m^2}{h^2 \cdot |S|}} \leq e^{-\frac{2km}{h^2}} \quad .$$

The lemma now follows by observing that $X \leq 2km/h$ implies that S is a dependent set under both $\mathcal{M}_{k,h,m}$ and $\mathcal{M}'_{k,h,m}$. ■

We now think of m as going to infinity and of h and k as constants. Notice that given this point of view the size of the ground set $\mathcal{N}_{k,h,m}$ is $nkh = O(m)$. Thus, the last lemma implies, via the union bound, that with high probability an algorithm making a polynomial number (in the size of the ground set) of independence checks will not be able to distinguishes between the cases in which it gets as input $\mathcal{M}_{k,h,m}$ or $\mathcal{M}'_{k,h,m}$.

We are now ready to prove Theorem 6.

Proof [Proof of Theorem 6] Consider an algorithm that needs to maximize the cardinality function over the k-extendible system $\mathcal{M}_{k,h,m}$ after the random permutation was applied, and let T be its output set. Notice that T must be independent in $\mathcal{M}_{k,h,m}$, and thus, its size is always upper bounded by mk. Moreover, since the algorithm fails, with high probability, to distinguish between $\mathcal{M}_{k,h,m}$ and $\mathcal{M}'_{k,h,m}$, T is with high probability also independent in $\mathcal{M}'_{k,h,m}$, and thus, has a size of at most m. Therefore, the expected size of T cannot be larger than $m + o(1)$.

On the other hand, Lemma 16 shows that $\mathcal{M}_{k,h,m}$ contains an independent set of size at least $mk(1 - 2k/h)$. Thus, the approximation ratio of the algorithm is no better than

$$\frac{mk(1-2k/h)}{m+o(1)} \geq \frac{mk(1-2k/h)}{m} - \frac{k}{m}o(1) = k - 2k^2/h - o(1) \quad .$$

Choosing a large enough h (compared to k), we can make this approximation ratio larger than $k - \varepsilon$ for any constant $\varepsilon > 0$. ■

To prove a stronger inapproximability result for monotone submodular objectives, we need to associate a monotone submodular function with each one of our k-extendible systems. Towards this goal, consider the monotone submodular function $f_h : 2^{\mathcal{N}_h} \rightarrow \mathbb{R}^+$ defined over the ground set $\mathcal{N}_h = [h]$ by

$$f_h(S) = \min\{|S|, 1\} \ .$$

Let $F_h \colon [0,1]^{\mathcal{N}_h} \rightarrow \mathbb{R}_{\geq 0}$ be the mutlilinear extension of f_h, i.e., $F_h(x) = \mathbb{E}[f_h(R(x))]$ for every vector $x \in [0,1]^{\mathcal{N}_h}$ (where $R(x)$ is a random set containing every element $u \in \mathcal{N}_u$ with probability x_u, independently). Additionally, given a vector $x \in [0,1]^{\mathcal{N}_h}$, let us define $\bar{x} = (\|x\|_1/h) \cdot \mathbf{1}_{\mathcal{N}_h}$. Notice that f_h is invariant under any permutation of the elements of $\mathcal{N}_h$. Thus, by Lemma 3.2 in (Vondrák, 2013), for every $\varepsilon' > 0$ there exists $\delta_h > 0$ and two functions $\hat{F}_h, \hat{G}_h : [0,1]^{\mathcal{N}_h} \rightarrow \mathbb{R}_{\geq 0}$ with the following properties.

- For all $x \in [0,1]^{\mathcal{N}_h}$: $\hat{G}_h(x) = \hat{F}_h(\bar{x})$.
- For all $x \in [0,1]^{\mathcal{N}_h}$, $|\hat{F}_h(x) - F_h(x)| \leq \varepsilon'$.
- Whenever $|x - \bar{x}|_2^2 \leq \delta_h$, $\hat{F}_h(x) = \hat{G}_h(x)$.
- The first partial derivatives of $\hat{F}_h$ and $\hat{G}_h$ are absolutely continuous.
- $\frac{\partial \hat{F}_h}{\partial x_u}, \frac{\partial \hat{G}_h}{\partial x_u} \geq 0$ everywhere for every $u \in \mathcal{N}_h$.
- $\frac{\partial^2 \hat{F}_h}{\partial x_u \partial x_v}, \frac{\partial^2 \hat{G}_h}{\partial x_u \partial x_v} \leq 0$ almost everywhere for every pair $u, v \in \mathcal{N}_h$.

The objective function we associate with $\mathcal{M}(k,h,m)$ is $\hat{F}_h(y(S))$, where $y(S)$ is a vector in $[0,1]^{\mathcal{N}_h}$ whose i^{th} coordinate is $|S \cap H_i(k,m)|/(km)$. Similarly, the objective function we associate with $\mathcal{M}'(k,h,m)$ is $\hat{G}_h(y(S))$. Notice that both objective functions are monotone and submodular by Lemma 3.1 of (Vondrák, 2013). We now bound the maximum value of a set in $\mathcal{M}(k,h,m)$ and $\mathcal{M}'(k,h,m)$ with respect to their corresponding objective functions.

Lemma 18 *The maximum value of a set in $\mathcal{M}(k,h,m)$ with respect to the objective $\hat{F}_h(y(S))$ is at least $1 - 2k/h - \varepsilon'$.*

Proof Observation 16 guarantees the existence of an independent set $S \subseteq H_1(k,m)$ in $\mathcal{M}(k,h,m)$ of size $s \geq k(m - 2km/h)$. The objective value associated with this set is

$$\hat{F}_h(y(S)) \geq F_h(y(S)) - \varepsilon' = \frac{s}{km} - \varepsilon' \geq \frac{k(m - 2km/h)}{km} - \varepsilon' = 1 - 2k/h - \varepsilon' \ .$$

■

Lemma 19 *The maximum value of a set in $\mathcal{M}'(k,h,m)$ with respect to the objective $\hat{G}_h(y(S))$ is at most $1 - e^{-1/k} + h^{-1} + \varepsilon'$.*

Proof The objective $\hat{G}_h(y(S))$ is monotone. Thus, the maximum value set in $\mathcal{M}'(k,h,m)$ must be of size m. Notice that for every set S of this size, we get

$$\begin{aligned}\hat{G}_h(y(S)) &= \hat{F}_h(\overline{y(S)}) = \hat{F}_h((kh)^{-1}\cdot\mathbf{1}_{\mathcal{N}_h}) \leq F_h((kh)^{-1}\cdot\mathbf{1}_{\mathcal{N}_h}) + \varepsilon' \\ &= 1 - \left(1-\frac{1}{kh}\right)^h + \varepsilon' \leq 1 - e^{-1/k}\left(1-\frac{1}{k^2h}\right) + \varepsilon' \leq 1 - e^{-1/k} + h^{-1} + \varepsilon' \enspace .\end{aligned}$$

■

As before, our plan is to show that after a random permutation is applied to the ground set it is difficult to distinguish between $\mathcal{M}(k,h,m)$ and $\mathcal{M}'(k,h,m)$ even when each one of them is accompanied with its associated objective. This will give us an inapproximability result which is roughly equal to the ratio between the bounds given by the last two lemmata.

Observe that Lemma 17 holds regardless of the objective function. Thus, $\mathcal{M}(k,h,m)$ and $\mathcal{M}'(k,h,m)$ are still polynomially indistinguishable. Additionally, the next lemma shows that their associated objective functions are also polynomially indistinguishable.

Lemma 20 *Following the application of the random ground set permutation, the probability that any given set S gets two different values under the two possible objective functions is at most $2h\cdot e^{-2mk\delta_h/h^2}$.*

Proof Recall that, as long as we consider a single set S, applying the permutation to the ground set is equivalent to replacing S with a random set of the same size. Hence, we are interested in the value under the two objective functions of a random set of size $|S|$. Define $X_i = |S\cap H_i(k,m)|$. Since X_i has the a hypergeometric distribution, the bound of (Skala, 2013) gives us

$$\begin{aligned}\Pr\left[X_i \geq \frac{|S|}{h} + mk\cdot\sqrt{\frac{\delta_h}{h}}\right] &= \Pr\left[X_i \geq \mathbb{E}[X_i] + mk\cdot\sqrt{\frac{\delta_h}{h}}\right] \\ &\leq e^{-2\cdot\left(\frac{mk\cdot\sqrt{\delta_h/h}}{|S|}\right)^2\cdot|S|} = e^{-\frac{2\delta_h}{h}\cdot\frac{m^2k^2}{|S|}} \leq e^{-2mk\delta_h/h^2} \enspace .\end{aligned}$$

Similarly, we also get

$$\Pr\left[X_i \leq \frac{|S|}{h} - mk\cdot\sqrt{\frac{\delta_h}{h}}\right] \leq e^{-2mk\delta_h/h^2} \enspace .$$

Combining both inequalities using the union bound now yields

$$\Pr\left[\left|X_i - \frac{|S|}{h}\right| \geq mk\cdot\sqrt{\frac{\delta_h}{h}}\right] \leq 2e^{-2mk\delta_h/h^2} \enspace .$$

Using the union bound again, the probability that $\left|X_i - \frac{|S|}{h}\right| \geq mk\cdot\sqrt{\frac{\delta_h}{h}}$ for any $1\leq i\leq h$ is at most $2h\cdot e^{-2mk\delta_h/h^2}$. Thus, to prove the lemma it only remains to show that the value of the two objective functions for S are equal when $\left|X_i - \frac{|S|}{h}\right| < mk\cdot\sqrt{\frac{\delta_h}{h}}$ for every $1\leq i\leq h$.

Notice that $\overline{y(S)}$ is a vector in which all the coordinates are equal to $|S|/(mkh)$. Thus, the inequality $\left|X_i - \frac{|S|}{h}\right| < mk \cdot \sqrt{\frac{\delta_h}{h}}$ is equivalent to $[y_i(S) - \overline{y_i(S)}] < \sqrt{\frac{\delta_h}{h}}$. Hence,

$$|y(S) - \overline{y(S)}|_2^2 = \sum_{i=1}^{h} (y_i(S) - \overline{y_i(S)})^2 < \sum_{i=1}^{h} \left(\sqrt{\frac{\delta_h}{h}}\right)^2 = \sum_{i=1}^{h} \frac{\delta_h}{h} = \delta_h \ ,$$

which implies the lemma by the properties of $\hat{F}_h$ and $\hat{G}_h$. ■

Consider a polynomial time deterministic algorithm that gets either $\mathcal{M}_{k,h,m}$ with its corresponding objective or $\mathcal{M}'_{k,h,m}$ with its corresponding objective after a random permutation was applied to the ground set. Consider first the case that the algorithm gets $\mathcal{M}'_{k,h,m}$ (and its corresponding objective). In this case, the algorithm checks the independence and value of a polynomial collection of sets (we may assume, without loss of generality, that the algorithm checks both things for every set that it checks). As before, one can observe that the sets in this collection do not depend on the permutation because the independence of a set in $\mathcal{M}'_{k,h,m}$ and its value with respect to $\hat{G}_h(y(S)) = \hat{F}_h(\overline{y(S)})$ depend only on the set's size, which guarantees that the algorithm takes the same execution path given every permutation. If the same algorithm now gets $\mathcal{M}_{k,h,m}$ instead, it will start checking the independence and values of the same sets until it will either get a different answer for one of the checks (different than what is expected for $\mathcal{M}'_{k,h,m}$) or it will finish all the checks. Note that in the later case the algorithm must return the same answer that it would have returned had it been given $\mathcal{M}'_{k,h,m}$.

By the union bound, Lemmata 17 and 20 imply that the probability that any of the sets whose value or independence is checked by the algorithm will result in a different answer for the two inputs decreases exponentially in m, and thus, with high probability the algorithm fails to distinguish between the inputs, and returns the same output for both. Moreover, note that by Yao's principal this observation extends also to polynomial time randomized algorithms.

We are now ready to prove Theorem 7.

Proof [Proof of Theorem 7] Consider an algorithm that needs to maximize $\hat{F}(y(S))$ over the k-extendible system $\mathcal{M}_{k,h,m}$ after the random permutation was applied, and let T be its output set. Notice that $\hat{F}(y(T)) \leq \hat{F}(\mathbf{1}_{\mathcal{N}_h}) \leq F(\mathbf{1}_{\mathcal{N}_h}) + \varepsilon' = 1 + \varepsilon'$. Moreover, the algorithm fails, with high probability, to distinguish between $\mathcal{M}_{k,h,m}$ and $\mathcal{M}'_{k,h,m}$. Thus, with high probability T is independent in $\mathcal{M}'(k,h,m)$ and has the same value under both objective functions $\hat{F}(y(S))$ and $\hat{G}(y(S))$, which implies, by Lemma 19, $\hat{F}(y(S)) = \hat{G}(y(S)) \leq 1 - e^{-1/k} + h^{-1} + \varepsilon'$. Hence, in conclusion we proved

$$\mathbb{E}[\hat{F}(y(T))] \leq 1 - e^{-1/k} + h^{-1} + \varepsilon' + o(1) \ .$$

On the other hand, Lemma 18 shows that $\mathcal{M}_{k,h,m}$ contains an independent set of value of at least $1 - 2k/h - \varepsilon'$ (with respect to $\hat{F}(y(S))$). Thus, the approximation ratio of the algorithm is no

better than

$$\begin{aligned}
&\frac{1-2k/h-\varepsilon'}{1-e^{-1/k}+h^{-1}+\varepsilon'+o(1)} \\
&\geq (1-e^{-1/k}+h^{-1}+\varepsilon'+o(1))^{-1}-(1-e^{-1/k})^{-1}(2k/h+\varepsilon') \\
&\geq (1-e^{-1/k})^{-1}-(1-e^{-1/k})^{-2}(h^{-1}+\varepsilon'+o(1))-(1-e^{-1/k})^{-1}(2k/h+\varepsilon') \\
&\geq (1-e^{-1/k})^{-1}-(k+1)^2(h^{-1}+\varepsilon'+o(1))-(k+1)(2k/h+\varepsilon') \ ,
\end{aligned}$$

where the last inequality holds since $1-e^{-1/k} \geq (k+1)^{-1}$. Choosing a large enough h (compared to k) and a small enough ε' (again, compared to k), we can make this approximation ratio larger than $(1-e^{-1/k})^{-1}-\varepsilon$ for any constant $\varepsilon > 0$. ■

Appendix B. Missing Proofs of Section 4

Lemma 9 *For every* $1 \leq i \leq \ell$, $f(S_i) \geq \frac{1}{k+1} f(S_i \cup (OPT \cap \mathcal{N}_i))$ *and* $f(S'_i) \geq f(S_i \cap OPT)/\alpha$.

Proof The set S_i is the output of the greedy algorithm when executed on the k-system obtained by restricting $(\mathcal{N}, \mathcal{I})$ to the ground set $\mathcal{N}_i$. Note that $\text{OPT} \cap \mathcal{N}_i$ is an independent set of this k-system. Thus, the inequality $f(S_i) \geq \frac{1}{k+1} f(S_i \cup (\text{OPT} \cap \mathcal{N}_i))$ is a direct application of Lemma 3.2 of [Gupta et al. (2010)] which states that the sets S obtained by running greedy with a k-system constraint must obey $f(S) \geq \frac{1}{k+1} f(S \cup C)$ for all independent sets C of the k-system.

Let us now explain why the second inequality of the lemma holds. Suppose that OPT_i is the subset of S_i maximizing f. Then,

$$f(S'_i) \geq f(\text{OPT}_i)/\alpha \geq f(S_i \cap \text{OPT})/\alpha \ ,$$

where the first inequality follows since α is the approximation ratio of the algorithm used for unconstrained submodular maximization, and the second inequality follows from the definition of OPT_i. ■

Lemma 10 $\sum_{i=1}^{\ell} f(S_i \cup OPT) \geq (\ell - 1) f(OPT)$.

Proof The proof is based on the following known result.

Claim 21 (Lemma 2.2 of Buchbinder et al. (2014)) *Let* $g : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ *be non-negative and submodular, and let* S *a random subset of* $\mathcal{N}$ *where each element appears with probability at most* p *(not necessarily independently). Then,* $\mathbb{E}[g(S)] \geq (1-p)g(\varnothing)$.

Using this claim we can now prove the lemma as follows. Let S be a random set which is equal to every one of the sets $\{S_i\}_{i=1}^{\ell}$ with probability $\frac{1}{\ell}$. Since these sets are disjoint, every element of $\mathcal{N}$ belongs to S with probability at most $p = \frac{1}{\ell}$. Additionally, let us define $g : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ as

$g(T) = f(T \cup \text{OPT})$ for every $T \subseteq \mathcal{N}$. One can observe that g is non-negative and submodular, and thus, by Claim 21,

$$\frac{1}{\ell}\sum_{i=1}^{\ell} f(S_i \cup \text{OPT}) = \mathbb{E}[f(S \cup \text{OPT})] = \mathbb{E}[g(S)] \geq (1-p)g(\varnothing) = \left(1 - \frac{1}{\ell}\right) f(\text{OPT}) \ .$$

The lemma now follows by multiplying both sides of the last inequality by ℓ. ■

Lemma 11 *Suppose f is a non-negative submodular function over ground set $\mathcal{N}$. For every three sets $A, B, C \subseteq \mathcal{N}$, $f(A \cup (B \cap C)) + f(B \setminus C) \geq f(A \cup B)$.*

Proof Observe that

$$\begin{aligned} f(A \cup (B \cap C)) + f(B \setminus C) &\geq f(A \cup (B \cap C) \cup (B \setminus C)) + f((A \cup (B \cap C)) \cap (B \setminus C)) \\ &\geq f(A \cup (B \cap C) \cup (B \setminus C)) = f(A \cup B) \ , \end{aligned}$$

where the first inequality follows from the submodularity of f, and the second inequality follows from its non-negativity. ■

In the proof of Theorem 5 we omitted the calculation showing that for $\ell = \lceil \sqrt{k} \rceil$, we get

$$\begin{aligned} \frac{k + \frac{\alpha}{2}\ell + 1 - \frac{\alpha}{2}}{1 - \frac{1}{\ell}} &\leq \frac{k + \frac{\alpha}{2}(\sqrt{k}+1) + 1 - \frac{\alpha}{2}}{1 - \frac{1}{\sqrt{k}}} = \frac{k + \frac{\alpha}{2}\sqrt{k} + 1}{1 - \frac{1}{\sqrt{k}}} \\ &\leq k + \left(1 + \frac{\alpha}{2}\right)\sqrt{k} + 2 + \frac{\alpha}{2} + \frac{4+\alpha}{\sqrt{k}} \ , \end{aligned}$$

where the last inequality holds because for $k \geq 4$ it holds that

$$\begin{aligned} \left[k + \left(1 + \frac{\alpha}{2}\right)\sqrt{k} + 2 + \frac{\alpha}{2} + \frac{4+\alpha}{\sqrt{k}}\right]\left(1 - \frac{1}{\sqrt{k}}\right) &= k + \frac{\alpha}{2}\sqrt{k} + 1 + \frac{4+\alpha}{2\sqrt{k}} - \frac{4+\alpha}{k} \\ &\geq k + \frac{\alpha}{2}\sqrt{k} + 1 \ . \end{aligned}$$

Appendix C. Missing Proofs of Section 5

Lemma 12 $f(S) \geq f(S \cup OPT) - \sum_{u \in \mathcal{N}} |O_u \setminus S| \Delta f(u|S_u)$.

Proof We first show that $f(S) \geq f(O)$, then we lower bound $f(O)$ to complete the proof. By **P1** and **P2**, we have $O \in \mathcal{I}$ and $S \subseteq O$, and thus, $S + v \in \mathcal{I}$ for all $v \in O \setminus S$ because $(\mathcal{N}, \mathcal{I})$ is an independence system. Consequently, the termination condition of Algorithm 3 guarantees that $\Delta f(v|S) \leq 0$ for all $v \in O \setminus S$. To use these observations, let us denote the elements of $O \setminus S$ by $v_1, v_2, \ldots, v_{|O \setminus S|}$ in an arbitrary order. Then

$$f(O) = f(S) + \sum_{i=1}^{|O \setminus S|} \Delta f\left(v_i | S \cup \{v_1, \ldots, v_{|O \setminus S|}\}\right) \leq f(S) + \sum_{i=1}^{|O \setminus S|} \Delta f\left(v_i | S\right) \leq f(S) \ ,$$

where the first inequality follows by the submodularity of f.

It remains to prove the lower bound on $f(O)$. By definition, O is the set obtained from OPT after the elements of $\cup_{u\in\mathcal{N}} O_u$ are removed and the elements of S are added. Additionally, an element that is removed from O is never added to O again, unless it becomes a part of S. This implies that the sets $\{O_u \setminus S\}_{u\in\mathcal{N}}$ are disjoint and that O can also be written as

$$O = (S \cup \text{OPT}) \setminus \cup_{u\in\mathcal{N}} (O_u \setminus S) \quad . \tag{6}$$

Denoting the the elements of $\mathcal{N}$ by $u_1, u_2, \ldots, u_n$ in an arbitrary order, and using the above, we get

$$\begin{aligned}
f(O) &= f(S \cup \text{OPT}) - \sum_{i=1}^{n} \Delta f\left(O_{u_i} \setminus S | (S \cup \text{OPT}) \setminus \cup_{1\leq j\leq i}(O_{u_j} \setminus S)\right) \qquad \text{(Equality (6))} \\
&\geq f(S \cup \text{OPT}) - \sum_{i=1}^{n} \Delta f(O_{u_i} \setminus S | S_{u_i}) \\
&\geq f(S \cup \text{OPT}) - \sum_{i=1}^{n} \sum_{v\in O_{u_i}\setminus S} \Delta f(v|S_{u_i}) \\
&= f(S \cup \text{OPT}) - \sum_{u\in\mathcal{N}} \sum_{v\in O_u\setminus S} \Delta f(v|S_u) \quad ,
\end{aligned}$$

where the first inequality follows from the submodularity of f because $S_{u_i} \subseteq S \subseteq (S \cup \text{OPT}) \setminus \cup_{u\in\mathcal{N}} (O_u \setminus S)$ and the second inequality follows from the submodularity of f as well.

To complete the proof of the lemma we need one more observation. Consider an element u for which O_u is not empty. Since O_u is not empty, we know that u was considered by the algorithm at some iteration. Moreover, every element of O_u was also a possible candidate for consideration at this iteration, and thus, it must be the case that u was selected for consideration because its marginal contribution with respect to S_u is at least as large as the marginal contribution of every element of O_u. Plugging this observation into the last inequality, we get the following desired lower bound on $f(O)$.

$$\begin{aligned}
f(O) &\geq f(S \cup \text{OPT}) - \sum_{u\in\mathcal{N}} \sum_{v\in O_u\setminus S} \Delta f(v|S_u) \\
&\geq f(S \cup \text{OPT}) - \sum_{u\in\mathcal{N}} \sum_{v\in O_u\setminus S} \Delta f(u|S_u) = f(S \cup \text{OPT}) - \sum_{u\in\mathcal{N}} |O_u \setminus S| \Delta f(u|S_u) \quad .
\end{aligned}$$

■

Lemma 13 $\mathbb{E}[f(S)] \geq \frac{1}{k+1} \sum_{u\in\mathcal{N}} \mathbb{E}[X_u \Delta f(u|S_u)]$.

Proof For each $u \in \mathcal{N}$, let G_u be a random variable whose value is equal to in the increase in the value of S when u is added to S by Algorithm 3. If u is never added to S by Algorithm 3, then the value of G_u is simply 0. Clearly,

$$f(S) = f(\varnothing) + \sum_{u\in\mathcal{N}} G_u \geq \sum_{u\in\mathcal{N}} G_u \quad .$$

By the linearity of expectation, it only remains to show that

$$\mathbb{E}[G_u] = \frac{1}{k+1}\mathbb{E}[X_u \Delta f(u|S_u)] \ . \tag{7}$$

Let $\mathcal{E}_u$ be an arbitrary event specifying all random decisions made by Algorithm 3 up until the iteration in which it considers u if u is considered, or all random decisions made by Algorithm 3 throughout its execution if u is never considered. By the law of total probability, since these events are disjoint, it is enough to prove that Equality (7) holds when conditioned on every such event $\mathcal{E}_u$. If $\mathcal{E}_u$ is an event that implies that Algorithm 3 does not consider u, then, by conditioning on $\mathcal{E}_u$, we obtain

$$\mathbb{E}[G_u|\mathcal{E}_u] = 0 = \frac{1}{k+1}\mathbb{E}[0 \cdot \Delta f(u|S_u)|\mathcal{E}_u] = \frac{1}{k+1}\mathbb{E}[X_u \Delta f(u|S_u)|\mathcal{E}_u] \ .$$

On the other hand, if $\mathcal{E}_u$ implies that Algorithm 3 does consider u, then we observe that S_u is a deterministic set given $\mathcal{E}_u$. Denoting this set by S'_u, we obtain

$$\mathbb{E}[G_u|\mathcal{E}_u] = \Pr\left[u \in S \mid \mathcal{E}_u\right] \Delta f(u|S'_u) = \frac{1}{k+1}\Delta f(u|S'_u) = \frac{1}{k+1}\mathbb{E}[X_u \Delta f(u|S_u)|\mathcal{E}_u] \ .$$

where the second equality hold since an element considered by Algorithm 3 is added to S with probability $(k+1)^{-1}$. ■

Lemma 14 *For every element $u \in \mathcal{N}$,*

$$\mathbb{E}[|O_u \setminus S|\Delta f(u|S_u)] \leq \frac{k}{k+1}\mathbb{E}[X_u \Delta f(u|S_u)] \ . \tag{8}$$

Proof As in the proof of Lemma 13, let $\mathcal{E}_u$ be an arbitrary event specifying all random decisions made by Algorithm 3 up until the iteration in which it considers u if u is considered, or all random decisions made by Algorithm 3 throughout its exectution if it never considers u. By the law of total probability, since these events are disjoint, it is enough to prove Inequality (8) conditioned on every such event $\mathcal{E}_u$. If $\mathcal{E}_u$ implies that u is not considered, then both $|O_u|$ and X_u are 0 conditioned on $\mathcal{E}_u$, and thus, the inequality holds as an equality. Thus, we may assume in the rest of the proof that $\mathcal{E}_u$ implies that u is considered by Algorithm 3. Notice that conditioned on $\mathcal{E}_u$ the set S_u is deterministic and X_u takes the value 1. Denoting the deterministic value of S_u conditioned on $\mathcal{E}_u$ by S'_u, Inequality (8) reduces to

$$\mathbb{E}[|O_u \setminus S| \mid \mathcal{E}_u]\Delta f(u|S'_u) \leq \frac{k}{k+1}\Delta f(u|S'_u) \ .$$

Since u is being considered, it must hold that $\Delta f(u|S'_u) > 0$, and thus, the above inequality is equivalent to $\mathbb{E}[|O_u \setminus S| \mid \mathcal{E}_u] \leq \frac{k}{k+1}$. There are now two cases to consider. If $\mathcal{E}_u$ implies that $u \in O$ at the beginning of the iteration in which Algorithm 3 considers u, then O_u is empty if u is added to S and is $\{u\}$ if u is not added to S. As u is added to S with probability $\frac{k}{k+1}$, this gives

$$\mathbb{E}[|O_u \setminus S| \mid \mathcal{E}_u] \leq \frac{1}{k+1} \cdot |\varnothing| + \left(1 - \frac{1}{k+1}\right)|\{u\}| = \frac{k}{k+1} \ ,$$

and we are done. Consider now the case that $\mathcal{E}_u$ implies that $u \notin O$ at the beginning of the iteration in which Algorithm 3 considers u. In this case, O_u is always of size at most k by the discussion following Algorithm 3, and it is empty when u is not added to S. As u is, again, added to S with probability $\frac{k}{k+1}$, we get in this case

$$\mathbb{E}[|O_u \setminus S| \mid \mathcal{E}_u] \leq \frac{1}{k+1} \cdot k + \left(1 - \frac{1}{k+1}\right) |\varnothing| = \frac{k}{k+1} \ .$$

■

In the proof of Theorem 4 we omitted the explanation why the inequality $\mathbb{E}[f(S \cup \text{OPT})] \geq \frac{k}{k+1} f(\text{OPT})$ holds. To see why this is the case, let us define $g : 2^{\mathcal{N}} \to \mathbb{R}_{\geq 0}$ as $g(T) = f(T \cup \text{OPT})$ for every $T \subseteq \mathcal{N}$. One can observe that g is non-negative and submodular. Since S contains every element with probability at most $(k+1)^{-1}$, we get by Claim 21

$$\mathbb{E}[f(S \cup \text{OPT})] = \mathbb{E}[g(S)] \geq \left(1 - \frac{1}{k+1}\right) g(\varnothing) = \frac{k}{k+1} f(\text{OPT}) \ .$$