Journal of Machine Learning Research 24 (2023) 1-87 Submitted 7/21; Revised 5/22; Published 1/23

How Do You Want Your Greedy: Simultaneous or Repeated?[*]

Moran Feldman MORANFE@CS.HAIFA.AC.IL
Department of Computer Science
University of Haifa
Haifa, Israel

Christopher Harshaw CHARSHAW@BERKELEY.EDU
Simons Institute
Univeristy of California, Berkeley
Berkeley, CA, USA

Amin Karbasi AMIN.KARBASI@YALE.EDU
Departments of Electrical Engeering, Computer Science, Statsitics & Data Science
Yale University
New Haven, CT, USA

Editor: Francis Bach

Abstract

We present SimultaneousGreedys, a deterministic algorithm for constrained submodular maximization. At a high level, the algorithm maintains ℓ solutions and greedily updates them in a *simultaneous* fashion. SimultaneousGreedys achieves the tightest known approximation guarantees for both k-extendible systems and the more general k-systems, which are $(k+1)^2/k = k + \mathcal{O}(1)$ and $(1+\sqrt{k+2})^2 = k + \mathcal{O}(\sqrt{k})$, respectively. We also improve the analysis of RepeatedGreedy, showing that it achieves an approximation ratio of $k+\mathcal{O}(\sqrt{k})$ for k-systems when allowed to run for $\mathcal{O}(\sqrt{k})$ iterations, an improvement in both the runtime and approximation over previous analyses. We demonstrate that both algorithms may be modified to run in nearly linear time with an arbitrarily small loss in the approximation.

Both SimultaneousGreedys and RepeatedGreedy are flexible enough to incorporate the intersection of m additional knapsack constraints, while retaining similar approximation guarantees: both algorithms yield an approximation guarantee of roughly $k + 2m + \mathcal{O}(\sqrt{k+m})$ for k-systems and SimultaneousGreedys enjoys an improved approximation guarantee of $k + 2m + \mathcal{O}(\sqrt{m})$ for k-extendible systems. To complement our algorithmic contributions, we prove that no algorithm making polynomially many oracle queries can achieve an approximation better than $k + 1/2 - \varepsilon$. We also present `SubmodularGreedy.jl`, a Julia package which implements these algorithms. Finally, we test these algorithms on real datasets.

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

*. Parts of the repeated greedy analysis and the inapproximability results presented in this paper have previously appeared in a preliminary form in a conference paper that appeared in COLT 2017 (Feldman et al., 2017).

1. Introduction

Submodular optimization has become widely adopted into the methodology of many areas of science and engineering. In addition to being a flexible modeling paradigm, submodular functions are defined by a diminishing returns property that naturally appears in a variety of disciplines, from machine learning and information theory to economics and neuroscience. Submodular optimization has been used in sensor placement (Krause and Guestrin, 2005), maximum likelihood inference in determinantal point processes (Gillenwater et al., 2012), influence maximization (Kempe et al., 2003), functional neuroimaging (Salehi et al., 2017), data summarization (Lin and Bilmes, 2011; Mirzasoleiman et al., 2013), crowd teaching (Singla et al., 2014), black-box interpretability (Elenberg et al., 2017), decision making (Alieva et al., 2020; Chen et al., 2015), and experimental design (Bian et al., 2017; Harshaw et al., 2019), to name a few examples. For more information on the applications of submodularity in machine learning and signal processing, we refer the interested reader to the recent survey by Tohidi et al. (2020). The simplest constraint class in these optimization problems is a cardinality constraint, which limits the number of elements any feasible solution may contain. However, as more applications emerge, there is a growing need for the development of fast algorithms that are able to handle more flexible and expressive constraint classes.

In this paper, we study the problem of maximizing a submodular functions subject to two constraint classes: k-systems and its (strict) subclass of k-extendible systems. These constraint classes capture a wide variety of constraints, including cardinality constraints, spanning trees, general matroids, intersection of matroids, graph matchings, scheduling, and even planar subgraphs. Moreover, the classes of k-systems and k-extendible systems enjoy a certain calculus for intersections (described more thoroughly in Section 2.2), which are desirable properties for practitioners who may be looking to optimize over richer sets of constraints. In the literature, there are two main algorithmic approaches for maximizing submodular functions over each of these constraint classes. The repeated greedy approach was initially proposed for submodular optimization over k-systems by Gupta et al. (2010), who showed that $\mathcal{O}(k)$ repeated iterations suffice to achieve a $3k$ approximation guarantee. Mirzasoleiman et al. (2016) refined this analysis, improving the approximation guarantee to $2k$. One contribution of this paper is to further improve the analysis of the repeated greedy technique, showing that $\mathcal{O}(\sqrt{k})$ suffices to achieve a $k + \mathcal{O}(\sqrt{k})$ approximation guarantee. The subsample greedy approach was proposed by Feldman et al. (2017) for submodular optimization over a k-extendible system, and achieves an improved approximation ratio of $(k+1)^2/k = k + \mathcal{O}(1)$.

One of the main downsides to these current approaches is that they are tailor made for the particular constraint class and do not perform as well otherwise. As we show in this paper, our analysis of the repeated greedy technique is tight in the sense that the algorithm attains an approximation guarantee of only $k+\Omega(\sqrt{k})$ for the subclass of k-extendible systems, regardless of the number of repeated iterations; similarly, the subsample greedy approach is not known to provide any approximation guarantee for the more general k-systems. Moreover, the types of approximation guarantees provided by the two algorithmic approaches differ: subsampling approaches are randomized algorithms, and their approximation guarantees hold in expectation—which may be too weak for certain applications where strong deterministic guarantees are preferable. Another downside is that while repeated greedy approaches may be modified to handle additional knapsack constraints (Mirzasoleiman et al., 2016), we are not aware of any known adaptation of subsampling greedy that allows it to handle such additional constraints.

Our main contribution in this work is SimultaneousGreedys, a deterministic algorithm for constrained submodular maximization. The new algorithmic idea is to greedily construct ℓ disjoint solutions in a *simultaneous* fashion. The solutions are all initialized to be empty; and at each iteration, an element is added to a solution in a greedy fashion, maximizing the marginal gain amongst all feasible element-solution pairs. At the end of the algorithm, the best solution is returned amongst the ℓ constructed solutions. One may interpret this SimultaneousGreedys as a derandomization of the subsample greedy technique. Subsample greedy produces a random solution whose objective value is large, in expectation; however, the support of the solution is exponentially sized, and so a naïve derandomization is infeasible. We show that the average objective value of the ℓ deterministically constructed solutions in SimultaneousGreedys is just as large, and in this sense we reduce the support of the distribution from exponential to constant.

Unlike the previous algorithmic techniques which were limited to specific constraint types, we show that SimultaneousGreedys achieves the best known approximation guarantees of $(1+\sqrt{k+2})^2 = k+\mathcal{O}(\sqrt{k})$ and $(k+1)^2/k = k+\mathcal{O}(1)$ for k-systems and k-extendible systems, respectively. In fact, these approximation ratios guaranteed by SimultaneousGreedys further improve to $k+1$ when the submodular objective function is monotone (in the case of k-systems, one needs to modify the value of ℓ to get this improvement).

Another contribution of this work is to show that both SimultaneousGreedys and RepeatedGreedy may be modified to create several different variants. First, we show that by employing an approximate greedy search based on a marginal gain thresholding technique (Badanidiyuru and Vondrák, 2014), both algorithms can be made to require only $\tilde{\mathcal{O}}(n/\varepsilon)$ queries to the value and independence oracles[1] at the cost of a $1+\varepsilon$ factor increase in the approximation guarantees. To our knowledge, this is the first nearly linear time algorithm for submodular maximization over a k-system. Next, we show that additional knapsack constraints may be incorporated into both algorithms by incorporating a density threshold technique (Mirzasoleiman et al., 2016) in the greedy selection procedure. Not only does this work improve upon the approximation guarantees and efficiency of Mirzasoleiman et al. (2016) for submodular maximization subject to a k-system constraint and m additional knapsacks, this work is also the first to provide (further improved) approximations when the subclass of k-extendible systems are considered. Even with these nearly linear time and knapsack modifications, the approximation guarantees of SimultaneousGreedys are still adaptive in the sense that they improve for k-extendible systems and they further improve when the objective function is monotone. For this reason, we consider SimultaneousGreedys to be like a Swiss Army knife for constrained submodular maximization: it is one main tool (the simultaneous greedy procedure) with several variants (nearly linear run time, density ratio technique) that can be used to produce the best known results for several problems of interest including k-systems, k-extendible systems,

1. Throughout the paper, we use the $\tilde{\mathcal{O}}$ notation to suppress poly-logarithmic factors.

Algorithm	**Running Time**	**k-system**	**k-extendible system**
Repeated Greedy (Gupta et al., 2010)	$\mathcal{O}(n^2 \cdot k)$	$3k$	(same as for k-system)
Sample Greedy (Feldman et al., 2017)	$\mathcal{O}(n^2/k)$	-	$k + \mathcal{O}(1)$ (in expectation)
Repeated Greedy **(this work)**	$\mathcal{O}(n^2 \cdot \sqrt{k})$	$k + \mathcal{O}(\sqrt{k})$	(same as for k-system)
SimultaneousGreedys **(this work)**	$\mathcal{O}(n^2 \cdot k^c)$	$k + \mathcal{O}(\sqrt{k})$	$k + \mathcal{O}(1)$
FastSGS **(this work)**	$\tilde{\mathcal{O}}(n/\varepsilon \cdot k^c)$	$(1+\varepsilon)k + \mathcal{O}(\sqrt{k})$	$(1+\varepsilon)k + \mathcal{O}(1)$
FANTOM (Mirzasoleiman et al., 2016)	$\tilde{\mathcal{O}}(n^2/\varepsilon \cdot km)$	$(1+\varepsilon)(2k + (2+2/k)m) + \mathcal{O}(1)$	(same as for k-system)
DensitySearchSGS **(this work)**	$\tilde{\mathcal{O}}(n/\varepsilon \cdot m(k^{c/2} + \sqrt{m}))$	$(1+\varepsilon)(k+2m) + \mathcal{O}(\sqrt{k+m})$	$(1+\varepsilon)(k+2m) + \mathcal{O}(\sqrt{m})$

Table 1: A comparison with previous works. In the running times, c is a constant equal to 2 for k-extendible systems and 1 for k-systems. The last two rows involve m knapsacks constraints in addition to the independence system constraint. Only modifications of the simultaneous greedy approach are shown, while modifications of the repeated greedy approach presented in this paper are suppressed.

intersection of these with additional knapsacks, and a possibly monotone objective. For a succinct summary of the comparison to previous work, see Table 1.

We complement these algorithmic contributions with a hardness result, showing that no algorithm making polynomially many queries to the value and independence oracles can yield an approximation factor smaller than $k + 1/2 - \varepsilon$ over a k-extendible system, a result that holds even for monotone objective functions. This hardness result demonstrates that the approximation produced by SimultaneousGreedys in the setting of k-extendible systems is nearly tight, and we prove it using the symmetric gap technique of Vondrák (2013). Note that because k-extendible systems are a subclass of k-systems, our hardness also holds for the more general class of k-systems; however, whether the additional $\mathcal{O}(\sqrt{k})$ term in the approximation factor is necessary for this class remains an open question. Moreover, an almost as strong hardness of $k - \varepsilon$ was already shown for k-systems by Badanidiyuru and Vondrák (2014).

Before concluding the section, let us make a few remarks on how to interpret the value k in our results. Throughout the paper, we generally view k as a large constant, e.g., $k = 10$ or $k = 25$. This is natural in applications where the complexity of the constraint class is fixed as the number of elements grows. For example, in the movie recommendation system application presented in Section 9, the number of genres is large, but remains fixed even as more movies (elements in the ground set) are added to the database. Indeed, prior work on k-systems and k-extendible systems typically consider (albeit implicitly) the value k to be a large constant—and thus, our discussions regarding k are aligned with those in the existing literature. From this perspective, the difference between an approximation ratio of

$k + \mathcal{O}(\sqrt{k})$ and $k + \mathcal{O}(1)$ is substantial, and hence, the improved approximation guarantees of SimultaneousGreedys compared to previous works are valuable. Nevertheless, we emphasize that all of our results hold for arbitrary values of k, and none of our analyses requires k to be constant with respect to the size of the ground set; rather, it is only the *interpretation* of our results that implicitly assumes that k is a large constant.

Organization The organization of the remainder of the paper is as follows: In the remainder of Section 1, we review the related works. We present the preliminary definitions and problem statement in Section 2. In Section 3, we present SimultaneousGreedys and its analysis. Section 4 contains the nearly linear time modification and Section 5 contains the additional knapsack modification. Our improved analysis of RepeatedGreedy, including linear-time and knapsack modifications, is contained in Section 6. The hardness results are presented in Section 7. Section 8 contains practical considerations when implementing these algorithms as well as a description of the `SubmodularGreedy.jl` package. Section 9 contains experiments on real datasets. Finally, we conclude in Section 10.

1.1 Related Work

The study of sumodular maximization over k-systems goes back to Fisher et al. (1978) who proved that the natural greedy algorithm obtains $(k+1)$-approximation when the objective function is monotone. Algorithms for maximizing *non-monotone* submodular functions under a k-system constraint, however, were not obtained until much more recently. Gupta et al. (2010) proposed a repeated greedy approach for this problem. At a high level, the algorithm repeatedly performs the following procedure: run the greedy algorithm to obtain a solution S, then perform unconstrained maximization on the elements of S to produce a set S', and finally remove the larger set S from the ground set. Among all considered solutions S and S', the set with the largest objective value is returned. Gupta et al. (2010) proved that when the number of iterations of repeated greedy is set to $k+1$, then the approximation ratio is roughly $3k$. This analysis was improved by Mirzasoleiman et al. (2016) who showed that the same repeated greedy algorithm achieves an approximation ratio of roughly $2k$. Note that because the repeated greedy based algorithms first consider the set returned by the greedy algorithm, their approximation ratios automatically improve to $k + 1$ for monotone objectives. We also remark that Mirzasoleiman et al. (2016) demonstrated that the repeated greedy technique may be modified to incorporate additional knapsack constraints through the use of a density thresholding technique.

An important subclass of the k-systems are the k-extendible systems, which were defined by Mestre (2006). The class of k-extendible systems is a strict subclass of the k-systems, and a more through treatment of these set systems, including examples, is provided in Section 2.2. For k-extendible systems, Feldman et al. (2017) introduced a subsampling approach as an alternative to repeated greedy, yielding an algorithm that is faster and also enjoys a somewhat better approximation guarantee. The idea is to independently subsample elements of the ground set, and then run the greedy algorithm once on the subsample. This subsampling approach runs in expected time $\mathcal{O}(n + nr/k)$, where r is the rank of the k-extendible system, and attains an approximation ratio of $(k + 1)^2/k$ in expectation. The main downside to this approach is that the approximation guarantee holds only in expectation, and thus, repetition is necessary to achieve a good approximation ratio with a high probability. The authors also show that even a very small number of repetitions suffices in practice. Nevertheless, the inherent uncertainty in the approximation quality of the returned solution may be undesirable in certain scenarios.

The class of k-extendible systems includes in its turn other subclasses of interest, including the class of k-exchange systems introduced by Feldman et al. (2011) and the well known class of k-intersection, which includes constraints that can be represented as the intersection of k matroids. Naturally, the above mentioned subsampling technique of Feldman et al. (2017) for k-extendible systems applies also to constraints from these two subclasses, and is arguably the best approximation ratio that can be achieved for these classes using practical techniques. However, local search approaches have been used to achieve improved approximation ratios for both these subclasses whose time complexity is exponential in both k and some error parameter $\varepsilon > 0$—which makes these improved approximation ratios mostly of theoretical interest (except maybe when k is very small). Specifically, for the intersection of $k \geq 2$ matroids, Lee et al. (2010a) proved an approximation ratio of $k + 2 + 1/k + \varepsilon$, which was later improved to $k + 1 + 1/(k + 1) + \varepsilon$ by Lee et al. (2010b). The last approximation ratio was later extended also to k-exchange systems by Feldman et al. (2011). The case of $k = 1$, in which all the above classes reduce to be the class of matroids, was also studied extensively, and the currently best approximation ratios for this case is roughly $(0.385)^{-1} \approx 2.60$ (Buchbinder and Feldman, 2018b). For an in-depth discussion of these algorithmic techniques, we refer readers to the survey of Buchbinder and Feldman (2018a).

The run time of greedy methods is typically quadratic, as each iteration requires examining all the remaining elements in the ground set. A heuristic often used to reduce this time complexity is the so-called "lazy greedy" approach, which uses the submodularity of the objective to avoid examining elements that cannot have the maximal marginal gain in a given iteration (Minoux, 1978). While this method typically yields a substantial improvement in practice, it does not improve the worst-case time complexity. However, inspired by the lazy greedy approach, Badanidiyuru and Vondrák (2014) proposed a technique known as "marginal gain thresholding", which reduces the run time of the greedy algorithm to $\mathcal{O}(n/\varepsilon \cdot \log(n/\varepsilon))$, while incurring only a small additive ε factor in the approximation ratio. Later on, Mirzasoleiman et al. (2015) proposed a stochastic approach which further reduces the run time of the greedy algorithm to $\mathcal{O}(n \log(1/\varepsilon))$, but applies only in the context of the simple cardinality constraint. Additional fast algorithms for submodular maximization were suggested by Badanidiyuru and Vondrák (2014); Buchbinder et al. (2017); Ene and Nguyen (2019a,b). It is also worth mentioning that most of the above algorithms can be further improved, in practice, using a lazy greedy like approach.

Our simultaneous greedy technique is most closely related to a recent work of Kuhnle (2019), where a similar "interlaced greedy" approach is proposed to obtain a $1/4 - \varepsilon$ approximation for maximizing a non-monotone submodular function subject to a cardinality constraint. The proposed idea is similar: simultaneously run the greedy algorithm to construct two disjoint solutions. In addition to extending to more general settings and subsuming these approximation results, the current work also demonstrates a tighter analysis even for the cardinality constraint presented in Kuhnle (2019). Namely, our analysis shows that only one run of the simultaneous greedy technique is required to obtain the $1/4 - \varepsilon$ approximation, whereas the analysis of Kuhnle (2019) requires the algorithm to be run twice in order to obtain this approximation. After an initial preprint of this work appeared online, Han et al. (2021) demonstrated that combining the simultaneous greedy approach with the subsampling approach yields improvements in the running time and the low order terms of the approximation guarantees.

2. Preliminaries

In this section, we introduce several preliminary definitions required for the problem we investigate. In Section 2.1, we define submodular set functions, which are the class of objective functions we consider. In Section 2.2, we discuss the independence systems that act as constraints in our problem. Finally, Section 2.3 formally defines our problem.

2.1 Submodular Functions

Let $\mathcal{N}$ be a finite set of size n which we refer to as the *ground set*. A real valued set function $f\colon 2^{\mathcal{N}} \to \mathbb{R}$ is *submodular* if for all sets $X, Y \subseteq \mathcal{N}$,

$$f(X \cup Y) + f(X \cap Y) \leq f(X) + f(Y) \enspace .$$

Given a set S and element e, we use the shorthand $S + u$ to denote the union $S \cup \{u\}$. Additionally, we define $f(u \mid S) = f(S + u) - f(S)$, i.e., $f(u \mid S)$ is the marginal gain with respect to f of adding e to the set S.[2] An equivalent definition of submodularity is that a function f is submodular if for all subsets $A \subseteq B \subseteq \mathcal{N}$ and element $u \notin B$,

$$f(u \mid A) \geq f(u \mid B) \enspace . \tag{1}$$

Inequality (1) is referred to as the diminishing returns property. Indeed, if f is interpreted as a utility function, then Inequality (1) states that the marginal gain of adding an element e to a subset decreases as the subset grows. Throughout the paper, we restrict our attention to non-negative submodular functions, i.e., functions whose value is non-negative for every set. The non-negativity is a necessary condition for obtaining multiplicative approximation guarantees.

A set function f is *modular* (or linear) if Inequality (1) always holds for it with equality. Any modular function can be represented using the form

$$f(S) = \sum_{u \in S} c_u + b$$

for an appropriate choice of a real number $c_u \in \mathbb{R}$ for every element $u \in \mathcal{N}$ and a fixed bias $b \in \mathbb{R}$. Finally, a set function f is *monotone* if adding more elements only increases its value; that is, $f(A) \leq f(B)$ for all subsets $A \subseteq B \subseteq \mathcal{N}$.

2.2 Independence Systems

The feasible sets in the optimization problems that we consider are described by an independence system. For $\mathcal{I} \subseteq 2^{\mathcal{N}}$, the pair $(\mathcal{I}, \mathcal{N})$ is an *independence system* if $\mathcal{I}$ is non-empty and satisfies the down-closure property, i.e., if $A \subseteq B$ and $B \in \mathcal{I}$ then $A \in \mathcal{I}$. For notational

2. More generally, we define $f(X \mid Y) = f(X \cup Y) - f(Y)$ for all sets $X, Y \subseteq \mathcal{N}$.

simplicity, we occasionally refer to the independence system as $\mathcal{I}$ when the ground set $\mathcal{N}$ is clear from context. A set $A \subseteq \mathcal{N}$ is called *independent* in the independence system $\mathcal{I}$ if $A \in \mathcal{I}$. Furthermore, if A is maximal independent set with respect to inclusion among all the subsets of a given set $B \subseteq \mathcal{N}$, then A is called a *base* of B. A base of the ground set $\mathcal{N}$ is also called a base of the independence system. The cardinality of the largest independent set of a given independence system $\mathcal{I}$ is known as the *rank* of the independence system, and we use r to denote it when the independence system is clear from the context.

There is a wide variety of independence systems which have been studied in the literature, and we review some of them here. An independence system $(\mathcal{I}, \mathcal{N})$ is a *k-system* if for every set $B \subseteq \mathcal{N}$ the ratio between the sizes of any two bases of B is at most k. Any independence system is a k-system for some $k \leq n$; however, we are most interested in settings where k is a constant or otherwise small with respect to the number n of elements in the ground set. A subclass of independence systems are the k-extendible systems. Intuitively, a k-extendible system is an independence system in which adding an element u to any independent set requires removing at most k elements to maintain independence. Formally, this means that an independence system is *k-extendible* if for every pair of independent sets $A \subseteq B \in \mathcal{I}$ and element $u \notin B$ such that $A + u \in \mathcal{I}$, there exists a set $Y \subseteq B \setminus A$ of size at most k such that $(B \setminus Y) \cup \{u\}$ is independent. It is known that every k-extendible system is also a k-system (Călinescu et al., 2011), that the intersection of a k_1-extendible system and a k_2-extendible system is a $(k_1 + k_2)$-extendible system and that the intersection of a k_1-system and a k_2-system is a $(k_1 + k_2)$-system (Haba et al., 2020). These observations provide a way to build more complex independence systems from simpler ones, allowing for a flexible framework for constraints in the optimization problems we consider.

One of the most well-studied examples of a k-extendible systems are the 1-extendible systems, which are also known as matroids.[3] Matroids capture a wide variety of set constraints, including independent sets of vectors, cardinality-constrained partitions, spanning forests, graph matchings, and the simple cardinality constraint. The intersection of k-matroids is a k-extendible system, but the converse is generally not true for $k \geq 2$. Indeed, the class of k-extendible systems includes systems which are not expressible as the intersection of a few matroids, including the class of b-matchings in graphs (which are 2-extendible) and asymmetric TSP (which is 3-extendible), as well as certain scheduling formulations (Mestre, 2006). Although the class of k-systems is strictly larger than the class of k-extendible systems, the majority of interesting examples are k-extendible systems. There are, however, a few exceptions such as the collection of all subsets of edges of a graph which induce a planar subgraph, which is 3-system (Haba et al., 2020). The taxonomy of the independence systems discussed above is depicted below, and all the containments are known to be strict for $k \geq 2$:

$$\text{cardinality constraint} \subset \text{matroid} \subset \text{intersection of } k \text{ matroids} \subset k\text{-extendible system} \subset k\text{-system} \enspace .$$

Knapsack constraints are another popular family of constraints that can be represented as independence systems. Formally, an independence system capturing a knapsack constraint is defined as the collection of sets $S \subseteq \mathcal{N}$ obeying $c(S) \leq 1$ for some non-negative

3. See Mestre (2006) for a proof of the equivalence of a 1-extendible system with the traditional definition of matroids.

modular function $c(S) = \sum_{u \in S} c_u$. We are often interested below in the intersection of m knapsack constraints, and denote the corresponding modular functions by $c_1, c_2, \dots, c_m$. In this work, and more broadly in the literature, knapsack constraints are considered separately from the main independence system constraint. Technically, this is not completely necessary because the intersection of m knapsacks is a k-extendible system for some k. However, this k might be as large as $m \cdot (c_{\max}/c_{\min})$, where $c_{\max}$ and $c_{\min}$ are the largest and smallest knapsack coefficients, respectively (i.e., $c_{\max} = \max_{u \in \mathcal{N}, i \in [m]} c_i(u)$ and $c_{\min} = \min_{u \in \mathcal{N}, i \in [m]} c_i(u)$). In contrast, treating the knapsack constraints as separate from the underlying independence system allows us to aim for approximation ratios that depend only on m, and is thus preferable.

2.3 Problem Statement

In this paper we study the problem of maximizing a non-negative submodular function subject to an independence system and the intersection of m knapsack constraints. More precisely, we aim to solve the following optimization problem

$$\max_{S \subseteq \mathcal{N}} f(S) \quad \text{subject to } S \in \mathcal{I} \text{ and } c_i(S) \leq 1 \quad \forall\, i = 1 \dots m \enspace , \tag{2}$$

where f is non-negative and submodular and $\mathcal{I}$ is an independence system which is either a k-system or a k-extendible system. For simplicity, we assume throughout the work that the singleton $\{u\}$ is a feasible solution for the above problem for every element $u \in \mathcal{N}$. Clearly, any element violating this assumption can be removed from the ground set without affecting the set of feasible solutions. We also denote by OPT an optimal solution to the program.

We evaluate our algorithms by their running times and approximation ratios. As is standard in the literature, our algorithms access the objective function and the independence system constraint only through value and independence oracles, respectively. The *value oracle* takes as input a set $S \subseteq \mathcal{N}$ and returns $f(S)$—the evaluation of f at S. Similarly, the *independence oracle* takes as input a set S and indicates whether or not $S \in \mathcal{I}$. The computational efficiency of algorithms in this model is often judged based on the number of oracle queries they make, and we follow this convention.

3. Simultaneous Greedys

In this section we present an algorithm named SimultaneousGreedys for solving Problem (2) in the special case of $m = 0$, i.e., the case in which there are no knapsack constraints. The main idea behind SimultaneousGreedys is to greedily and *simultaneously* construct ℓ disjoint solutions by iteratively adding elements to the solutions in a way that maximizes the momentary marginal gain. Formally, the algorithm begins by initializing ℓ solutions $S^{(1)}, S^{(2)}, \dots, S^{(\ell)}$ to be empty sets. At each iteration, the algorithm considers all the pairs of element u and solution $S^{(j)}$ such that (1) u does not yet belong to any of the solutions, (2) u can be added to $S^{(j)}$ without violating independence, and (3) the addition of u to $S^{(j)}$ increases the objective value of $S^{(j)}$. The set of such pairs is denoted by $\mathcal{A}$ in the pseudocode of the algorithm. Among all the considered pairs, the algorithm picks the one for which $f(u \mid S^{(j)})$ is maximal (i.e., the pair for which the addition of u to $S^{(j)}$ yields the maximal increase in the value of the solution), and then adds u to $S^{(j)}$. The algorithm terminates when no further pairs with all the above properties can be found. The pseudocode of SimultaneousGreedys appears below as Algorithm 1.

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

```
1  Initialize $\ell$ solutions, $S_0^{(j)} \leftarrow \varnothing$ for $j = 1, \dots \ell$.
2  Initialize available ground set $\mathcal{N}_0 \leftarrow \mathcal{N}$, and iteration counter $i \leftarrow 1$.
3  Initialize feasible element-solution pairs
     $\mathcal{A}_1 = \{(u, j) : \{u\} \in \mathcal{I}, f(u \mid \varnothing) > 0, j \in [\ell]\}$.
4  while $\mathcal{A}_i$ is nonempty do
5      Let $(u_i, j_i) \leftarrow \max_{(u,j) \in \mathcal{A}_i} f(u \mid S_{i-1}^{(j)})$ be a feasible element-solution pair
         maximizing the marginal gain.
6      Update the solutions as $S_i^{(j)} \leftarrow \begin{cases} S_{i-1}^{(j_i)} + u_i & \text{if } j = j_i \\ S_{i-1}^{(j)} & \text{if } j \neq j_i \end{cases}$
7      Update the available ground set $\mathcal{N}_i \leftarrow \mathcal{N}_{i-1} - u_i$.
8      Update the feasible element-solution pairs,
         $\mathcal{A}_{i+1} = \{(u, j) : u \in \mathcal{N}_i, S_i^{(j)} + u \in \mathcal{I}, f(u \mid S_i^{(j)}) > 0\}$.
9      Update iteration counter $i \leftarrow i + 1$.
10 return the set $S$ maximizing $f$ among the sets $\{S_i^{(j)}\}_{j=1}^{\ell}$.
```

We begin our analysis of SimultaneousGreedys by providing a bound on the number of oracle calls used by the algorithm.

Observation 1 SimultaneousGreedys *requires at most $\mathcal{O}(\ell^2 rn)$ calls to the value and independence oracles.*

Proof In every single iteration, the algorithm examines the possibility of adding each of the n elements to each of the ℓ solutions, requiring $\mathcal{O}(\ell n)$ calls to the value and independence oracles. Since exactly one element is added to some solution at every iteration, the number of iterations is the sum of the cardinalities of the produced solutions, which is at most ℓr because all the solutions are feasible. Combining the two above observations, i.e., that there are at most ℓr iterations, each requiring $\mathcal{O}(\ell n)$ oracle calls, we get that the total number of oracle calls required by SimultaneousGreedys is $\mathcal{O}(\ell^2 rn)$. ■

We now present theorems proving approximation guarantees for SimultaneousGreedys when $\mathcal{I}$ is guaranteed to be either a k-system or a k-extendible system. To get the tightest approximation guarantees from these theorems, one has to set the number ℓ of constructed solutions differently for the two classes of constraints.

Theorem 2 *Suppose that $(\mathcal{N}, \mathcal{I})$ is a k-extendible system and that the number of solutions is set to $\ell = k + 1$. Then,* SimultaneousGreedys *requires $\mathcal{O}(k^2 rn)$ oracle calls and produces a solution whose approximation ratio is at most $(k+1)^2/k = k + \mathcal{O}(1)$. Moreover, when f is non-negative monotone submodular, then the approximation ratio improves to $k + 1$.*

Theorem 3 *Suppose that $(\mathcal{N}, \mathcal{I})$ is k-system and that the number of solutions is set to $\ell = \lfloor 2+\sqrt{k+2} \rfloor$. Then,* SimultaneousGreedys *requires $\mathcal{O}(krn)$ oracle calls and produces a solution whose approximation ratio is at most $(1+\sqrt{k+2})^2 = k + \mathcal{O}(\sqrt{k})$. Moreover, when f is non-negative monotone submodular and the number of solutions is set to $\ell = 1$, then the approximation ratio improves to $k+1$.*

Note that the improved approximation for k-extendible systems comes at the higher computational cost of an extra $\mathcal{O}(k)$ factor in the running time. Moreover, the gain in approximation is only for the non-monotone setting, as the two approximation guarantees are the same for monotone objectives. In both Theorems 2 and 3, the bound on the required number of oracle calls is a direct application of Observation 1 and the choice of ℓ, the number of constructed solutions. The proof of the approximation ratios is more involved. In Section 3.1, we provide a unified meta-proof for analyzing SimultaneousGreedys given a constraint obeying some kinds of parametrized properties. Then, in Sections 3.2 and 3.3 we show that k-extendible systems and k-systems have these properties for a proper choice of the parameters, respectively, yielding the the different approximation guarantees of Theorems 2 and 3.

The second part of Theorem 3 considers $\ell = 1$, which recovers the greedy algorithm. Although it was previously known that the greedy algorithm achieves $(k+1)$-approximation for monotone submodular objectives under a k-system, we remark that this result for this special setting is cleanly obtained by our unified analysis. We also remark that, for monotone objectives, the result of Theorem 2 holds for any number of solutions $\ell \leq k+1$; which further demonstrates that the analysis of the greedy algorithm is handled by our meta-analysis. The details for the case of $\ell \leq k+1$ are covered in the proof of Theorem 2.

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

```
1 Let $\mathcal{N}' \leftarrow \varnothing$ and $S \leftarrow \varnothing$.
2 for each $u \in \mathcal{N}$ do
3     with probability $(k+1)^{-1}$ do
4         Add $u$ to $\mathcal{N}'$.
5 while there exists $u \in \mathcal{N}'$ such that $S + u \in \mathcal{I}$ and $f(u \mid S) > 0$ do
6     Let $u \in \mathcal{N}'$ be the element of this kind maximizing $f(u \mid S)$.
7     Add $u$ to $S$.
8 return $S$.
```

As mentioned in Section 1, one may interpret SimultaneousGreedys as a de-randomization of SampleGreedy, the subsampling algorithm of Feldman et al. (2017) presented here as Algorithm 2. SampleGreedy creates a subsample of the ground set by sampling each element independently with probability p and then running the vanilla greedy algorithm. Feldman et al. (2017) show that, for k-extendible systems, setting the sampling probability to $p = (k+1)^{-1}$ yields an approximation ratio of $(k+1)^2/k$, which improves to $k+1$ for monotone objectives (i.e., the same approximation guarantees of SimultaneousGreedys for these cases). One of the key step in the analysis of SampleGreedy is an averaging argument over the distribution of solutions it may produce, whose support might be of exponential size. This means that naïvly trying to de-randomize SampleGreedy requires keeping all the states which it might take, and therefore, yields an exponential algorithm.

In the analysis of SimultaneousGreedys we bypass this hurdle by managing to make the above averaging argument work for a much smaller distribution whose support consists only of the ℓ solutions maintained by the algorithm. We note that this idea of de-randomizing a randomized algorithm by coming up with a polynomial size distribution mimicking the behavior of an exponential size distribution was originally used in the context of submodular maximization by Buchbinder and Feldman (2018b), albeit using very different techniques based on linear programming. Finally, unlike SampleGreedy, SimultaneousGreedys has the additional benefit of producing approximation guarantees for the more general class of k-systems.

3.1 Meta-analysis for Approximation Guarantees

In this section, we present a unified analysis for obtaining approximation guarantees for SimultaneousGreedys under general independence system constraints. Specifically, Proposition 4 reduces the conditions for approximation to simple combinatorial statements relating the constructed solutions to OPT. These combinatorial statements are shown to hold for k-extendible systems and k-systems in Sections 3.2 and 3.3, respectively.

The main idea of the unified analysis is to keep track of the elements of OPT which could have been—but were not—added to each of the ℓ solutions by the algorithm. At the beginning of the algorithm, all solutions are initialized to the empty set and so each element of OPT could be added to each solution in the first iteration. However, every time that the algorithm adds an element to one of the solutions, it means that certain elements of OPT are now no longer able to be added to that solution, due to the independence constraint. In this sense, these elements of OPT are "thrown away" from the set of possible elements to be added to the solution. The main technical requirement of the unified approximation analysis is that only a few elements of OPT are thrown away in this sense at each iteration. These conditions are more precisely stated in the hypothesis of Proposition 4.

In the proposition below, we let T be the number of iterations performed by SimultaneousGreedys, and let $U_i^{(j)}$ be the singleton set $\{u_i\}$ if $j = j_i$ and the empty set otherwise.

Proposition 4 *Let us define $O_0^{(j)} = OPT$ for every solution $1 \leq j \leq \ell$. If there exist a value p and sets $O_i^{(j)}$ for every iteration $1 \leq i \leq T$ and solution $1 \leq j \leq \ell$ such that*

- *$S_i^{(j)} + u$ is independent for each iteration $0 \leq i \leq T$, solution $1 \leq j \leq \ell$, and element $u \in O_i^{(j)}$.*
- *$O_i^{(j)} \subseteq O_{i-1}^{(j)} \cap \mathcal{N}_i$ for each iteration $1 \leq i \leq T$ and solution $1 \leq j \leq \ell$.*
- *$(S_T^{(j)} \setminus S_i^{(j)}) \cap OPT \subseteq O_i^{(j)}$ for each iteration $0 \leq i \leq T$ and solution $1 \leq j \leq \ell$.*
- *$\sum_{j=1}^{\ell} |O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})| \leq p$ for each iteration $1 \leq i \leq T$.*

Then, the solution S produced by SimultaneousGreedys is a $\frac{p+1}{1-\ell^{-1}}$-approximation solution. Moreover, this approximation ratio improves to $p + 1$ when f is monotone.

Before proceeding, we would like to provide some intuition for the conditions appearing in Proposition 4. Intuitively, the set $O_i^{(j)}$ contains elements of OPT which have not already been added to a solution and can still be added to the j-th solution at iteration i. Condition

1 formally states this ability to add the elements of $O_i^{(j)}$ to the j-th solution, and Condition 2 formally states that the elements in $O_i^{(j)}$ do not already appear in a solution. Condition 3 requires $O_i^{(j)}$ to include all the elements of OPT which are eventually (but not yet) included in one of the final solutions. Finally, Condition 4 is a bound on the number of elements which are removed from these sets at each iteration. Together, these conditions are strong enough to provide a general approximation guarantee.

The following lemma is the first step towards proving Proposition 4. Intuitively, this lemma shows that as the iteration i increases, the decrease in the value of $f(O_i^{(j)} \mid S_i^{(j)})$ is transferred, at least to some extent, to $S_i^{(j)}$.

Lemma 5 *Given the conditions of Proposition 4, for every iteration* $0 \leq i \leq T$*,*

$$(p+1) \cdot \sum_{j=1}^{\ell} f(S_i^{(j)}) + \sum_{j=1}^{\ell} f(O_i^{(j)} \mid S_i^{(j)}) \geq \sum_{j=1}^{\ell} f(OPT \cup S_i^{(j)}) \enspace .$$

Proof We prove the lemma by induction on the iterations $i = 0, 1, \dots, T$. The base case is the case of $i = 0$, corresponding to the initialization of the algorithm. Recall that the solutions are initialized to be empty, i.e., $S_0^{(j)} = \varnothing$ for every $j \in [\ell]$. This, together with non-negativity of f, implies

$$\begin{aligned}
\sum_{j=1}^{\ell} f(OPT \cup S_0^{(j)}) &= \sum_{j=1}^{\ell} f(OPT \cup \varnothing) && \text{(by the initialization } S_0^{(j)} = \varnothing\text{)}\\
&= \sum_{j=1}^{\ell} f(\varnothing) + \sum_{j=1}^{\ell} f(OPT \mid \varnothing) && \text{(rearranging terms)}\\
&= (p+1) \cdot \sum_{j=1}^{\ell} f(\varnothing) + \sum_{j=1}^{\ell} f(OPT \mid \varnothing) && (f(\varnothing) \geq 0 \text{ by the non-negativity})\\
&\leq (p+1) \cdot \sum_{j=1}^{\ell} f(S_0^{(j)}) + \sum_{j=1}^{\ell} f(O_0^{(j)} \mid S_0^{(j)}) \enspace . && \text{(by the initialization } S_0^{(j)} = \varnothing\text{)}
\end{aligned}$$

Assume now that the lemma holds for all iterations between 0 to $i-1$, and let us prove it for iteration i. Recall that only the solution $S_i^{(j_i)}$ is modified during iteration i. Thus, we have that the change in iteration i in the first sum in the guarantee of the lemma is

$$(p+1) \cdot \sum_{j=1}^{\ell} f(S_i^{(j)}) - (p+1) \cdot \sum_{j=1}^{\ell} f(S_{i-1}^{(j)}) = (p+1) \cdot f(u_i \mid S_{i-1}^{(j_i)}) \enspace . \tag{3}$$

Bounding the change in the second sum in the guarantee is more involved, and is done in two steps. The first step is the following inequality.

$$\begin{aligned}\sum_{j=1}^{\ell} f(O_{i-1}^{(j)} \mid S_{i-1}^{(j)}) - \sum_{j=1}^{\ell} f(O_{i-1}^{(j)} \mid S_{i}^{(j)}) & \qquad\qquad (4)\\ &= f(O_{i-1}^{(j_i)} \mid S_{i-1}^{(j_i)}) - f(O_{i-1}^{(j_i)} \mid S_{i}^{(j_i)}) && \text{(only } S_i^{(j_i)} \text{ is modified)}\\ &= f(u_i \mid S_{i-1}^{(j_i)}) - f(u_i \mid O_{i-1}^{(j_i)} \cup S_{i-1}^{(j_i)}) && \text{(rearranging terms)}\\ &\leq f(u_i \mid S_{i-1}^{(j_i)}) - f(u_i \mid OPT \cup S_{i-1}^{(j_i)}) \enspace ,\end{aligned}$$

where the inequality may be proved by considering two cases. First, suppose that $u_i \in O_{i-1}^{(j_i)} \cup S_{i-1}^{(j_i)}$. In this case, the inequality holds with equality, because $O_{i-1}^{(j_i)} \subseteq OPT$ by assumption. Consider now the case in which $u_i \notin O_{i-1}^{(j_i)} \cup S_{i-1}^{(j_i)}$. In this case, our assumption that $(S_T^{(j_i)} \setminus S_{i-1}^{(j_i)}) \cap OPT \subseteq O_{i-1}^{(j_i)}$ implies $u_i \notin (S_T^{(j_i)} \setminus S_{i-1}^{(j_i)}) \cap OPT$, which implies in its turn $u_i \notin OPT$ since $u_i \in S_i^{(j_i)} \subseteq S_T^{(j_i)}$ and $u_i \in \mathcal{N}_{i-1} \subseteq \mathcal{N} \setminus S_{i-1}^{(j_i)}$. Therefore, we get that in this case that Inequality (4) holds due to the submodularity of f (recall that $O_{i-1}^{(j_i)} \subseteq OPT$ by our assumption).

For the second step in the proof of the above mentioned bound, we need to observe that, by the definition of the pair (u_i, j_i), we have

$$f(u_i \mid S_{i-1}^{(j_i)}) \geq f(u \mid S_{i-1}^{(j)}) \geq f(u \mid S_i^{(j)}),$$

for any element $u \in \mathcal{N}_{i-1}$ and integer $1 \leq j \leq \ell$ for which $S_{i-1}^{(j)} + u$ is independent—the second inequality follows from submodularity when either $u \neq u_i$ or $j \neq j_i$ and from the non-negativity of $f(u_i \mid S_{i-1}^{(j_i)})$ when $u = u_i$ and $j = j_i$. Since $O_{i-1}^{(j)} \subseteq \mathcal{N}_{i-1}$ and $S_{i-1}^{(j)} + u$ is independent for every $u \in O_{i-1}^{(j)}$ by our assumption, the last inequality implies

$$\begin{aligned}&\sum_{j=1}^{\ell} f(O_{i-1}^{(j)} \mid S_i^{(j)})\\ &\quad\leq \sum_{j=1}^{\ell} f(O_i^{(j)} \mid S_i^{(j)}) + \sum_{j=1}^{\ell} \sum_{u \in O_{i-1}^{(j)} \setminus O_i^{(j)}} f(u \mid S_i^{(j)}) && \text{(submodularity, } O_i^{(j)} \subseteq O_{i-1}^{(j)})\\ &\quad= \sum_{j=1}^{\ell} f(O_i^{(j)} \mid S_i^{(j)}) + \sum_{j=1}^{\ell} \sum_{u \in O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})} f(u \mid S_i^{(j)}) && (U_i^{(j)} \subseteq S_i^{(j)})\\ &\quad\leq \sum_{j=1}^{\ell} f(O_i^{(j)} \mid S_i^{(j)}) + \sum_{j=1}^{\ell} \sum_{u \in O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})} f(u_i \mid S_{i-1}^{(j_i)}) && \text{(greedy selection of } u_i)\\ &\quad= \sum_{j=1}^{\ell} f(O_i^{(j)} \mid S_i^{(j)}) + f(u_i \mid S_{i-1}^{(j_i)}) \cdot \sum_{j=1}^{\ell} |O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})| && \text{(rearranging terms)}\end{aligned}$$

$$\leq \sum_{j=1}^{\ell} f(O_i^{(j)} \mid S_i^{(j)}) + p \cdot f(u_i \mid S_{i-1}^{(j_i)}) \ , \tag{5}$$

where the last inequality holds by our assumption that $\sum_{j=1}^{\ell} |O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})| \leq p$ and the non-negativity of $f(u_i \mid S_{i-1}^{(j_i)})$. Combining Inequalities (3), (4) and (5), we get

$$\begin{aligned}
&(p+1) \cdot \sum_{j=1}^{\ell} f(S_i^{(j)}) + \sum_{j=1}^{\ell} f(O_i^{(j)} \mid S_i^{(j)}) \\
&\quad \geq \left[(p+1) \cdot \sum_{j=1}^{\ell} f(S_{i-1}^{(j)}) + (p+1) \cdot f(u_i \mid S_{i-1}^{(j_i)})\right] + \left[\sum_{j=1}^{\ell} f(O_{i-1}^{(j)} \mid S_i^{(j)}) - p \cdot f(u_i \mid S_{i-1}^{(j_i)})\right] \\
&\quad = (p+1) \cdot \sum_{j=1}^{\ell} f(S_{i-1}^{(j)}) + \left[\sum_{j=1}^{\ell} f(O_{i-1}^{(j)} \mid S_i^{(j)}) + f(u_i \mid S_{i-1}^{(j_i)})\right] \\
&\quad \geq (p+1) \cdot \sum_{j=1}^{\ell} f(S_{i-1}^{(j)}) + \sum_{j=1}^{\ell} f(O_{i-1}^{(j)} \mid S_{i-1}^{(j)}) + f(u_i \mid OPT \cup S_{i-1}^{(j_i)}) \\
&\quad \geq \sum_{j=1}^{\ell} f(OPT \cup S_{i-1}^{(j)}) + f(u_i \mid OPT \cup S_{i-1}^{(j_i)}) \\
&\quad = \sum_{j=1}^{\ell} f(OPT \cup S_i^{(j)}) \ ,
\end{aligned}$$

where the second inequality follows from submodularity and the last inequality follows from the induction hypothesis. ■

The following corollary uses the last lemma to prove a lower bound on the sum of the objective values of the ℓ final solutions in terms of the optimal solution.

Corollary 6 *Given the conditions of Proposition 4,*

$$(p+1) \cdot \sum_{j=1}^{\ell} f(S_T^{(j)}) \geq \sum_{j=1}^{\ell} f(OPT \cup S_T^{(j)}) \ .$$

Proof The termination condition of SimultaneousGreedys implies that $f(u \mid S_T^{(j)}) \leq 0$ for every element $u \in \mathcal{N}_T$ and integer $1 \leq j \leq \ell$ such that $S_T^{(j)} + u$ is independent. Since $O_T^{(j)} \subseteq \mathcal{N}_T$ and $S_T^{(j)} + u$ is independent for every $u \in O_T^{(j)}$ by our assumption, this implies

$$f(O_T^{(j)} \mid S_T^{(j)}) \leq \sum_{u \in O_T^{(j)}} f(u \mid S_T^{(j)}) \leq 0 \ ,$$

where the first inequality follows from the submodularity of f. Plugging this observation into the guarantee of Lemma 5 for $i = T$ yields

$$\sum_{j=1}^{\ell} f(OPT \cup S_T^{(j)}) \leq (p+1) \cdot \sum_{j=1}^{\ell} f(S_T^{(j)}) + \sum_{j=1}^{\ell} f(O_T^{(j)} \mid S_T^{(j)}) \leq (p+1) \cdot \sum_{j=1}^{\ell} f(S_T^{(j)}) \ . \ \blacksquare$$

To get an approximation ratio from the guarantee of the last corollary, we need to relate the sum $\sum_{j=1}^{\ell} f(OPT \cup S_T^{(j)})$ to $f(OPT)$. We do this using the following known lemma.

Lemma 7 (Lemma 2.2 of Buchbinder et al. (2014)) *Let $g\colon 2^{\mathcal{N}} \to \mathbb{R}_+$ be non-negative and submodular, and let S a random subset of $\mathcal{N}$ in which each element appears with probability at most p (not necessarily independently). Then, $\mathbb{E}[g(S)] \geq (1-p) \cdot g(\varnothing)$.*

We are now ready to prove Proposition 4.

Proof *(Proof of Proposition 4)*: Recall that the set S returned by SimultaneousGreedys is the one having the largest objective value amongst all of the ℓ solutions. Thus, by a simple averaging argument together with Corollary 6, we obtain the following lower bound on its objective value,

$$f(S) = \max_{j=1\dots\ell} f(S_T^{(j)}) \geq \frac{1}{\ell} \cdot \sum_{j=1}^{\ell} f(S_T^{(\ell)}) \geq \frac{1}{p+1} \left[\frac{1}{\ell} \cdot \sum_{j=1}^{\ell} f(OPT \cup S_T^{(\ell)}) \right] . \tag{6}$$

Consider now a random set $\bar{S}$ chosen uniformly at random from the ℓ constructed solutions $S_T^{(1)}, S_T^{(2)}, \dots, S_T^{(\ell)}$. Since the solutions are disjoint by construction, an element can belong to $\bar{S}$ with probability at most ℓ^{-1}. Hence, by applying Lemma 7 to the submodular function $g(S) = f(OPT \cup S)$, we get

$$\frac{1}{\ell} \cdot \sum_{j=1}^{\ell} f(OPT \cup S_T^{(\ell)}) = \mathbb{E}[f(OPT \cup \bar{S})] = \mathbb{E}[g(\bar{S})] \geq (1-\ell^{-1}) \cdot g(\varnothing) = (1-\ell^{-1}) \cdot f(OPT) \ .$$

Together with Inequality (6), this shows that the returned solution S is a $(p+1)/(1-\ell^{-1})$-approximation, as desired. We remark also that if f is monotone, then for each solution $1 \leq j \leq \ell$ we have that $f(OPT \cup S_T^{(\ell)}) \geq f(OPT)$. Applying this directly to Inequality (6) yields that the returned set S is a $(p+1)$-approximation when f is monotone. $\blacksquare$

3.2 Analysis for k-Extendible Systems

In this section we use Proposition 4 to prove Theorem 2. Throughout this section we assume that $(\mathcal{N}, \mathcal{I})$ is a k-extendible system. We demonstrate that for any number of solutions ℓ, the conditions of Proposition 4 hold with the value $p = \max(k, \ell-1)$. The proof of Theorem 2 follows by setting $\ell = k+1$.

In order to show that the conditions of Proposition 4 hold, we need to construct a set $O_i^{(j)}$ for every iteration $0 \leq i \leq T$ and solution $1 \leq j \leq \ell$. Thus, we begin the section by explaining how to construct these sets. The construction is done in a recursive way, and with the knowledge of the algorithm's execution path. For $i = 0$, we define $O_0^{(j)} = OPT$ for every $1 \leq j \leq \ell$, as is required by Proposition 4. Assume now that the sets $O_{i-1}^{(1)}, O_{i-1}^{(2)}, \ldots, O_{i-1}^{(\ell)}$ have already been constructed for some iteration $i > 0$, then we construct the sets $O_i^{(1)}, O_i^{(2)}, \ldots, O_i^{(\ell)}$ as follows:

- For every solution $1 \leq j \leq \ell$ other than j_i, $O_i^{(j)} = O_{i-1}^{(j)} - u_i$.
- If $u_i \in O_{i-1}^{(j_i)}$, then $O_i^{(j_i)} = O_{i-1}^{(j_i)} - u_i$, else $O_i^{(j_i)}$ is any maximal subset of $O_{i-1}^{(j_i)}$ such that $O_i^{(j_i)} \cup S_i^{(j_i)}$ is independent and $(S_T^{(j_i)} \setminus S_i^{(j_i)}) \cap OPT \subseteq O_i^{(j_i)}$. To see that such a subset must exist, note that the set $(S_T^{(j_i)} \setminus S_i^{(j_i)}) \cap OPT$ has all the necessary properties because $[(S_T^{(j_i)} \setminus S_i^{(j_i)}) \cap OPT] \cup S_i^{(j_i)} \subseteq S_T^{(j_i)}$ is an independent set, and $(S_T^{(j_i)} \setminus S_i^{(j_i)}) \cap OPT \subseteq (S_T^{(j_i)} \setminus S_{i-1}^{(j_i)}) \cap OPT \subseteq O_{i-1}^{(j_i)}$.

Proposition 8 *If $(\mathcal{I}, \mathcal{N})$ is a k-extendible system, then the sets $O_i^{(j)}$ constructed above satisfy the conditions of Proposition 4 with $p = \max(k, \ell - 1)$.*

The next four lemmata together prove Proposition 8 by verifying each of the conditions in Proposition 4. The most technical lemma, which uses the k-extendible assumption, is Lemma 12.

Lemma 9 *For every iteration $0 \leq i \leq T$ and solution $1 \leq j \leq \ell$, $O_i^{(j)} \cup S_i^{(j)}$ is independent, and thus, $S_i^{(j)} + u$ is independent for every $u \in O_i^{(j)}$.*

Proof We prove the lemma by induction on the iteration i. For $i = 0$, the lemma holds since

$$O_i^{(j)} \cup S_i^{(j)} = OPT \cup \varnothing = OPT.$$

Assume now that the lemma holds for all iterations up to and including $i - 1 \geq 0$, and let us prove it for iteration i. For solutions which were not updated at this iteration (that is, $j \neq j_i$), the lemma follows from the induction hypothesis since

$$O_i^{(j)} \cup S_i^{(j)} = [O_{i-1}^{(j)} - u_i] \cup S_{i-1}^{(j)} \subseteq O_{i-1}^{(j)} \cup S_{i-1}^{(j)} \enspace .$$

It remains to prove the lemma for the solution $j = j_i$ which was updated. If $u_i \notin O_{i-1}^{(j_i)}$, then $O_i^{(j)} \cup S_i^{(j)}$ is independent by the construction of $O_i^{(j)}$. Otherwise, $O_i^{(j)} \cup S_i^{(j)}$ is independent by the induction hypothesis since

$$O_i^{(j)} \cup S_i^{(j)} = [O_{i-1}^{(j)} - u_i] \cup [S_{i-1}^{(j)} + u_i] = O_{i-1}^{(j)} \cup S_{i-1}^{(j)} \enspace . \qquad \blacksquare$$

Lemma 10 *For every iteration $1 \leq i \leq T$ and solution $0 \leq j \leq \ell$, $O_i^{(j)} \subseteq O_{i-1}^{(j)} \cap \mathcal{N}_i$. Moreover, for $i = 0$ we have $O_i^{(j)} \subseteq \mathcal{N}_i$ for every solution $1 \leq j \leq \ell$.*

Proof We prove the lemma by induction on iterations i. For $i = 0$, the lemma trivially holds since $\mathcal{N}_0 = \mathcal{N}$. Assume now that the lemma holds for iterations up to and including $i - 1 \geq 0$, and let us prove it for iteration i. By the construction of $O_i^{(j)}$, it is a subset of $O_{i-1}^{(j)}$, an thus, to prove the lemma it suffices to show that $O_i^{(j)} \subseteq \mathcal{N}_i = \mathcal{N}_{i-1} - u_i$.

The last inclusion follows from combining the next two observations: By the induction hypothesis, $O_{i-1}^{(j)}$ is a subset of $\mathcal{N}_{i-1}$, and therefore, so must be $O_i^{(j)}$. If $u_i \notin O_{i-1}^{(j)}$, then u_i cannot belong to $O_i^{(j)}$ because the last set is a subset of $O_{i-1}^{(j)}$. Otherwise, we get by construction $O_i^{(j)} = O_{i-1}^{(j)} - u_i$, which guarantees again that u_i does not belong to $O_i^{(j)}$. ■

Lemma 11 *For every iteration $0 \leq i \leq T$ and solution $1 \leq j \leq \ell$, $(S_T^{(j)} \setminus S_i^{(j)}) \cap OPT \subseteq O_i^{(j)}$.*

Proof We prove the lemma by induction on the iterations i. For $i = 0$, the lemma holds since

$$(S_T^{(j)} \setminus S_i^{(j)}) \cap OPT \subseteq OPT = O_i^{(j)} \enspace .$$

Assume now that the lemma holds for all iterations up to and including $i - 1 \geq 0$, and let us prove it for iteration i. There are two cases to consider. If $O_i^{(j)} = O_{i-1}^{(j)} - u_i$, then by the induction hypothesis, since u_i is the sole element of $S_i^{(j)}$ that does not appear in $S_{i-1}^{(j)}$ (if there is such an element at all),

$$(S_T^{(j)} \setminus S_i^{(j)}) \cap OPT = (S_T^{(j)} \setminus S_{i-1}^{(j)}) \cap OPT - u_i \subseteq O_{i-1}^{(j)} - u_i = O_i^{(j)} \enspace .$$

It remains to consider the case in which $O_i^{(j)} \neq O_{i-1}^{(j)} - u_i$. However, there is only one case in the construction of $O_i^{(j)}$ in which this might happen, and in this case $O_i^{(j)}$ is chosen as a set including $(S_T^{(j)} \setminus S_i^{(j)}) \cap OPT$, so there is nothing to prove. ■

Lemma 12 *For every iteration $1 \leq i \leq T$, $\sum_{j=1}^{\ell} |O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})| \leq \max(k, \ell - 1)$.*

Proof There are two cases to consider. If $u_i \in OPT$, then Lemma 11 guarantees that $O_{i-1}^{(j_i)}$ contains u_i, and thus, by construction, $O_i^{(j)} = O_{i-1}^{(j)} - u_i$ for every solution $1 \leq j \leq \ell$. Thus,

$$\sum_{i=1}^{\ell} |O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})| \leq \sum_{i=1}^{\ell} |\{u_i\} \setminus U_i^{(j)}| = \ell - 1 \enspace ,$$

where the equality holds since, by definition, $U_i^{(j)}$ is equal to $\{u_i\}$ for $j = j_i$ and to $\varnothing$ for every other j.

Consider now the case of $u_i \notin OPT$. In this case u_i does not belong to $O_{i-1}^{(j)}$ for any j because a repeated application of Lemma 10 can show that $O_{i-1}^{(j)}$ is a subset of OPT. Since $O_i^{(j)} = O_{i-1}^{(j)} - u_i$ for every solution $j \neq j_i$, we get for every such solution j,

$$O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)}) = \varnothing \enspace .$$

To understand the set $O^{(j_i)}_{i-1} \setminus (O^{(j_i)}_i \cup U^{(j_i)}_i)$, we need to make a few observations. First, we recall that by Lemma 9, $O^{(j_i)}_{i-1} \cup S^{(j_i)}_{i-1}$ is independent. Second,

$$(S^{(j_i)}_T \setminus S^{(j_i)}_i) \cap OPT \subseteq O^{(j_i)}_i \subseteq O^{(j_i)}_{i-1}$$

by Lemmata 10 and 11, and finally,

$$S^{(j_i)}_{i-1} \cup [(S^{(j_i)}_T \setminus S^{(j_i)}_i) \cap OPT] + u_i \subseteq S^{(j_i)}_T$$

is also independent. Since $(\mathcal{N}, \mathcal{I})$ is k-extendible, these three observations imply together that there must exist a set Y of size at most k such that $(O^{(j_i)}_{i-1} \setminus Y) \cup S^{(j_i)}_i$ is independent, and Y does not include elements of $(S^{(j_i)}_T \setminus S^{(j_i)}_i) \cap OPT$. One can now observe that $O^{(j_i)}_{i-1} \setminus Y$ obeys all the conditions to be $O^{(j_i)}_i$ according to the construction of this set in the case of $u_i \notin O^{(j_i)}_{i-1}$, and thus, since the construction selects a maximal set obeying these conditions as $O^{(j_i)}_i$, we get

$$|O^{(j_i)}_i| \geq |O^{(j_i)}_{i-1} \setminus Y| \geq |O^{(j_i)}_{i-1}| - |Y| \geq |O^{(j_i)}_{i-1}| - k \enspace .$$

Since $O^{(j_i)}_i$ is a subset of $O^{(j_i)}_{i-1}$, this implies

$$|O^{(j_i)}_{i-1} \setminus (O^{(j_i)}_i \cup U^{(j_i)}_i)| \leq |O^{(j_i)}_{i-1} \setminus O^{(j_i)}_i| = |O^{(j_i)}_{i-1}| - |O^{(j_i)}_i| \leq k \enspace .$$

Combining everything that we have proved for the case of $u_i \notin OPT$, we get that in this case

$$\sum_{i=1}^{\ell} |O^{(j)}_{i-1} \setminus (O^{(j)}_i \cup U^{(j)}_i)| = (\ell - 1) \cdot |\varnothing| + |O^{(j_i)}_{i-1} \setminus (O^{(j_i)}_i \cup U^{(j_i)}_i)| \leq k \enspace .$$

The two cases together yields that the sum in question is at most $p = \max(k, \ell - 1)$. ■

We are now ready to prove Theorem 2.

Proof *(Proof of Theorem 2)*: Lemmata 9, 10, 11 and 12 together prove Proposition 8, which states that the sets we have constructed obey the conditions of Proposition 4 for $p = \max(k, \ell - 1)$. This implies that the approximation ratio of SimultaneousGreedys for k-extendible systems is at most

$$\frac{p+1}{1-\ell^{-1}} = \frac{\max(k, \ell-1)+1}{1-\ell^{-1}} = \frac{\max(k+1, \ell)}{1-\ell^{-1}}$$

Choosing the number of solutions to be $\ell = k + 1$ optimizes this approximation factor and yields

$$\frac{\max(k+1, \ell)}{1-\ell^{-1}} = \frac{k+1}{1-(k+1)^{-1}} = \frac{(k+1)^2}{(k+1)-1} = \frac{(k+1)^2}{k} \enspace .$$

Now further suppose that f is monotone in addition to being submodular and non-negative. In this case, Proposition 4 guarantees an approximation factor of at most

$$p + 1 = \max(k, \ell - 1) + 1 = \max(k + 1, \ell) \enspace ,$$

which demonstrates that the approximation factor improves to $k + 1$ for any number of solutions $\ell \leq k + 1$. ■

3.3 Analysis for k-Systems

In this section we use Proposition 4 to prove Theorem 3. Throughout this section we assume that $(\mathcal{N}, \mathcal{I})$ is a k-system. We demonstrate that for any number of solutions ℓ, the conditions of Proposition 4 hold with the value $p = k + \ell - 1$. Then, to prove Theorem 3, we choose $\ell = \lfloor 2 + \sqrt{k+2} \rfloor$.

To use Proposition 4, we need to construct a set $O_i^{(j)}$ for every iteration $0 \leq i \leq T$ and solution $1 \leq j \leq \ell$. As in Section 3.2, these sets are constructed recursively and with knowledge of the deterministic algorithm's execution path; however, for the case of k-systems, the construction of these sets starts at the final iteration and works backwards to the first iteration. We begin by constructing related sets $\tilde{O}_i^{(j)}$ using the following recursive rule.

- For the final iteration $i = T$, $\tilde{O}_i^{(j)}$ contains all the elements of $OPT \setminus S_i^{(j)}$ that can be added to $S_i^{(j)}$ without violating independence. In other words, $\tilde{O}_i^{(j)} = \{u \in OPT \setminus S_i^{(j)} \mid S_i^{(j)} + u \in \mathcal{I}\}$.
- For earlier iterations $i < T$, if the solution j is unaffected at this iteration (that is, $j \neq j_{i+1}$) then we simply set $\tilde{O}_i^{(j)} = \tilde{O}_{i+1}^{(j)}$. Otherwise, let $B_i^{(j_{i+1})}$ be the set of elements of $OPT \setminus (S_{i+1}^{(j_{i+1})} \cup \tilde{O}_{i+1}^{(j_{i+1})})$ that can be added to $S_i^{(j_{i+1})}$ without violating independence. In other words,

  $$B_i^{(j_{i+1})} = \{u \in OPT \setminus (S_{i+1}^{(j_{i+1})} \cup \tilde{O}_{i+1}^{(j_{i+1})}) \mid S_i^{(j_{i+1})} + u \in \mathcal{I}\}.$$

  We also denote by $\tilde{B}_i^{(j_{i+1})}$ an arbitrary subset of $B_i^{(j_{i+1})}$ of size $\min\{|B_i^{(j_{i+1})}|, k\}$. Using this notation, we can now define

  $$\tilde{O}_i^{(j_{i+1})} = \tilde{O}_{i+1}^{(j_{i+1})} \cup \tilde{B}_i^{(j_{i+1})} \cup (OPT \cap \{u_{i+1}\}).$$

Using the sets $\tilde{O}_i^{(j)}$ defined by the above recursive rule, we can now define the sets $O_i^{(j)}$ using the following formula. For every iteration $0 \leq i \leq r$ and solution $1 \leq j \leq \ell$, let $O_i^{(j)} = \tilde{O}_i^{(j)} \cap \mathcal{N}_i$.

Proposition 13 *If $(\mathcal{I}, \mathcal{N})$ is a k-system, then the sets $O_i^{(j)}$ constructed above satisfy the conditions of Proposition 4 with $p = k + \ell - 1$.*

The following lemmata together prove Proposition 13 by verifying each of the conditions of Proposition 4. Unlike the case in Section 3.2, here it is not clear from the construction that $O_0^{(j)} = OPT$ for each of the solutions $1 \leq j \leq \ell$. The next lemma proves that this is indeed the case.

Lemma 14 *For every solution $1 \leq j \leq \ell$, $\tilde{O}_0^{(j)} = OPT$, and thus, $O_0^{(j)} = \tilde{O}_0^{(j)} \cap \mathcal{N}_0 = OPT$ because $\mathcal{N}_0 = \mathcal{N}$.*

Proof By reverse induction over the iterations, we prove the stronger claim: that for every iteration $0 \leq i \leq T$ and solution $1 \leq j \leq \ell$,

$$|OPT \setminus (\tilde{O}_i^{(j)} \cup S_i^{(j)})| \leq k \cdot |S_i^{(j)}|.$$

Notice that this claim indeed implies the lemma since $\tilde{O}_i^{(j)}$ contains only elements of OPT and $S_0^{(j)} = \varnothing$.

We begin the proof by induction by showing that the claim holds for at the final iteration $i = T$. By the definition of $\tilde{O}_T^{(j)}$, no element of $OPT \setminus (\tilde{O}_T^{(j)} \cup S_T^{(j)})$ can be added to $S_T^{(j)}$ without violating independence, and thus, $S_T^{(j)}$ is a base of $(OPT \setminus \tilde{O}_T^{(j)}) \cup S_T^{(j)}$. In contrast, $OPT \setminus (\tilde{O}_T^{(j)} \cup S_T^{(j)})$ is an independent subset of $(OPT \setminus \tilde{O}_T^{(j)}) \cup S_T^{(j)}$ because it is also a subset of the independent set OPT. Thus, since $(\mathcal{N}, \mathcal{I})$ is a k-system,

$$|OPT \setminus (\tilde{O}_T^{(j)} \cup S_T^{(j)})| \leq k \cdot |S_T^{(j)}| \enspace ,$$

which is the claim that we wanted to prove.

Assume now that the claim holds for all iterations $i+1, i+2, \ldots, T$, and let us prove it for iteration i. There are three cases to consider. If the solution j was not updated during iteration $i+1$ (i.e., $j \neq j_{i+1}$), then $\tilde{O}_i^{(j)} = \tilde{O}_{i+1}^{(j)}$ and $S_i^{(j)} = S_{i+1}^{(j)}$, and therefore, by the induction hypothesis,

$$|OPT \setminus (\tilde{O}_i^{(j)} \cup S_i^{(j)})| = |OPT \setminus (\tilde{O}_{i+1}^{(j)} \cup S_{i+1}^{(j)})| \leq k \cdot |S_{i+1}^{(j)}| = k \cdot |S_i^{(j)}| \enspace .$$

The second case is when $j = j_{i+1}$ and $|\tilde{B}_i^{(j)}| = k$. In this case,

$$\begin{aligned} |OPT \setminus (\tilde{O}_i^{(j)} \cup S_i^{(j)})| &= |OPT \setminus (\tilde{O}_{i+1}^{(j)} \cup S_{i+1}^{(j)})| - |\tilde{B}_i^{(j)}| \\ &= |OPT \setminus (\tilde{O}_{i+1}^{(j)} \cup S_{i+1}^{(j)})| - k \leq k \cdot |S_{i+1}^{(j)}| - k = k \cdot |S_i^{(j)}| \enspace , \end{aligned}$$

where the inequality holds by the induction hypothesis, and the first equality holds since $\tilde{O}_i^{(j)} \setminus \tilde{O}_{i+1}^{(j)} = \tilde{B}_i^{(j)} \cup (OPT \cap \{u_{i+1}\})$, the elements of $\tilde{B}_i^{(j)}$ belong to $OPT \setminus S_{i+1}^{(j)}$ and the element u_{i+1} does not belong to this set.

The last case we need to consider is when $j = j_{i+1}$ and $|\tilde{B}_i^{(j)}| < k$. In this case $\tilde{B}_i^{(j)} = B_i^{(j)}$, which implies that no element of $OPT \setminus (\tilde{O}_i^{(j)} \cup S_i^{(j)})$ can be added to $S_i^{(j)}$ without violating independence, and thus, $S_i^{(j)}$ is a base of $(OPT \setminus \tilde{O}_i^{(j)}) \cup S_i^{(j)}$. This allows us to prove the claim in the same way in which this is done in the base case. Specifically, observe that $OPT \setminus (\tilde{O}_i^{(j)} \cup S_i^{(j)})$ is an independent subset of $(OPT \setminus \tilde{O}_i^{(j)}) \cup S_i^{(j)}$ because it is also a subset of the independent set OPT. Thus, since $(\mathcal{N}, \mathcal{I})$ is a k-system,

$$|OPT \setminus (\tilde{O}_i^{(j)} \cup S_i^{(j)})| \leq k \cdot |S_i^{(j)}| \enspace ,$$

which is the claim that we wanted to prove. ■

We now proceed to proving the explicit conditions of Proposition 4. The most technical condition, which uses the k-system assumption, is Lemma 18.

Lemma 15 *For every iteration $0 \leq i \leq T$ and solution $1 \leq j \leq \ell$, $S_i^{(j)} + u$ is independent for every $u \in O_i^{(j)}$.*

Proof We prove by a reverse induction the stronger claim that for every iteration $0 \leq i \leq T$ and solution $1 \leq j \leq \ell$, the set $S_i^{(j)} + u$ is independent for every $u \in \tilde{O}_i^{(j)}$. Note that this claim implies the lemma because $O_i^{(j)}$ is a subset of $\tilde{O}_i^{(j)}$.

At the last iteration $i = T$, the claim is an immediate consequence of the definition of $\tilde{O}_r^{(j)}$. Assume now that the claim holds for iterations $i+1, i+2, \dots T$, and let us prove it for iteration i. If the solution j was not updated at iteration $i+1$ (i.e., $j \neq j_{i+1}$), then $S_i^{(j)} = S_{i+1}^{(j)}$ and $\tilde{O}_i^{(j)} = \tilde{O}_{i+1}^{(j)}$, and so the claims follows immediately from the induction hypothesis. Thus, it remains to consider only the solution that is updated, i.e., $j = j_{i+1}$. In this case,

$$\tilde{O}_i^{(j)} = \tilde{O}_{i+1}^{(j)} \cup \tilde{B}_i^{(j)} \cup (OPT \cap \{u_{i+1}\}).$$

For every $u \in \tilde{O}_{i+1}^{(j)}$, we have that $S_i^{(j)} + u$ is independent by the induction hypothesis since $S_i^{(j)}$ is a subset of $S_{i+1}^{(j)}$. For every $u \in \tilde{B}_i^{(j)}$, we have that $S_i^{(j)} + u$ is independent by the definition of $B_i^{(j)}$. Finally, for $u = u_{i+1}$, we have $S_i^{(j)} + u = S_{i+1}^{(j)} \in \mathcal{I}$. ■

Lemma 16 *For every iteration $1 \leq i \leq T$ and solution $1 \leq j \leq \ell$, $O_i^{(j)} \subseteq O_{i-1}^{(j)} \cap \mathcal{N}_i$.*

Proof We first observe that $\tilde{O}_i^{(j)} \subseteq \tilde{O}_{i-1}^{(j)}$ by construction, and $\mathcal{N}_i = \mathcal{N}_{i-1} - u_i \subseteq \mathcal{N}_{i-1}$. Thus,

$$O_i^{(j)} = \tilde{O}_i^{(j)} \cap \mathcal{N}_i \subseteq \tilde{O}_{i-1}^{(j)} \cap \mathcal{N}_{i-1} = O_{i-1}^{(j)} \enspace .$$ ■

Lemma 17 *For every iteration $0 \leq i \leq T$ and solution $1 \leq j \leq \ell$, $(S_T^{(j)} \setminus S_i^{(j)}) \cap OPT \subseteq O_i^{(j)}$.*

Proof We prove the lemma by reverse induction on the iterations. At the final iteration $i = T$, the claim that we need to prove is trivial since $S_T^{(j)} \setminus S_i^{(j)} = \varnothing$. Assume now that the lemma holds for iterations $i+1, i+2, \dots T$, and let us prove it for iteration i. If the solution set is not updated at iteration $i+1$ (i.e., $j \neq j_{i+1}$), then $S_i^{(j)} = S_{i+1}^{(j)}$, which implies

$$(S_T^{(j)} \setminus S_i^{(j)}) \cap OPT = (S_T^{(j)} \setminus S_{i+1}^{(j)}) \cap OPT \subseteq O_{i+1}^{(j)} \subseteq O_i^{(j)} \enspace ,$$

where the first inclusion holds by the induction hypothesis, and second inclusion by Lemma 16. Thus, it remains to consider only the solution for which $j = j_{i+1}$.

In this case

$$[(S_T^{(j)} \setminus S_i^{(j)}) \cap OPT] \setminus [(S_T^{(j)} \setminus S_{i+1}^{(j)}) \cap OPT] = OPT \cap \{u_{i+1}\} \subseteq \tilde{O}_i^{(j)} \cap \mathcal{N}_i = O_i^{(j)} \enspace ,$$

where the inclusion follows from the definition of $\tilde{O}_i^{(j)}$ and the fact that u_{i+1} is chosen as an element from $\mathcal{N}_i$. Using the induction hypothesis, we now get

$$(S_T^{(j)} \setminus S_i^{(j)}) \cap OPT = [(S_T^{(j)} \setminus S_{i+1}^{(j)}) \cap OPT] \cup [OPT \cap \{u_{i+1}\}] \subseteq O_{i+1}^{(j)} \cup O_i^{(j)} = O_i^{(j)} \enspace ,$$

where the final equality follows again from Lemma 16. ■

Lemma 18 *For every iteration $1 \leq i \leq T$, $\sum_{j=1}^{\ell} |O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})| \leq k + \ell - 1$.*

Proof For every solution $1 \leq j \leq \ell$ other than j_i, we have by definition $U_i^{(j)} = \varnothing$ and

$$O_{i-1}^{(j)} = \tilde{O}_{i-1}^{(j)} \cap \mathcal{N}_{i-1} = \tilde{O}_i^{(j)} \cap (\mathcal{N}_i + u_i) \subseteq \tilde{O}_i^{(j)} \cap \mathcal{N}_i + u_i = O_i^{(j)} + u_i \enspace .$$

Therefore,

$$|O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})| \leq 1 \enspace .$$

Additionally,

$$\tilde{O}_{i-1}^{(j_i)} = \tilde{O}_i^{(j_i)} \cup \tilde{B}_{i-1}^{(j_i)} \cup (OPT \cap \{u_i\}) = \tilde{O}_i^{(j_i)} \cup \tilde{B}_{i-1}^{(j_i)} \cup (OPT \cap U_i^{(j)}) \subseteq \tilde{O}_i^{(j_i)} \cup \tilde{B}_{i-1}^{(j_i)} \cup U_i^{(j)} \enspace ,$$

and

$$\mathcal{N}_{i-1} = \mathcal{N}_i + u_i = \mathcal{N}_i \cup U_i^{(j)} \enspace .$$

These two observations imply together

$$\begin{aligned} O_{i-1}^{(j_i)} &= \tilde{O}_{i-1}^{(j_i)} \cap \mathcal{N}_{i-1} \\ &\subseteq [\tilde{O}_i^{(j_i)} \cup \tilde{B}_{i-1}^{(j_i)} \cup U_i^{(j)}] \cap [\mathcal{N}_i \cup U_i^{(j)}] \\ &\subseteq [\tilde{O}_i^{(j_i)} \cap \mathcal{N}_i] \cup \tilde{B}_{i-1}^{(j_i)} \cup U_i^{(j)} \\ &= O_i^{(j_i)} \cup \tilde{B}_{i-1}^{(j_i)} \cup U_i^{(j)} \enspace , \end{aligned}$$

and therefore, also

$$|O_{i-1}^{(j_i)} \setminus (O_i^{(j_i)} \cup U_i^{(j_i)})| \leq |\tilde{B}_{i-1}^{(j_i)}| \leq k \enspace ,$$

where the last inequality follows from the definition of $\tilde{B}_{i-1}^{(j_i)}$.

Combining all the above results, we get

$$\sum_{j=1}^{\ell} |O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})| \leq (\ell - 1) \cdot 1 + k = k + \ell - 1 \enspace . \qquad \blacksquare$$

We are now ready to prove Theorem 3.

Proof *(Proof of Theorem 3)*: Lemmata 15, 16, 17 and 18 prove together Proposition 13, which states that the sets we have constructed obey the conditions of Proposition 4 with $p = k + \ell - 1$. Thus, the last proposition implies that the approximation ratio of SimultaneousGreedys for k-systems and $\ell = \lfloor 2 + \sqrt{k+2} \rfloor$ is at most

$$\frac{p+1}{1 - \ell^{-1}} = \frac{k + \ell}{1 - \ell^{-1}} = \frac{k + \lfloor 2 + \sqrt{k+2} \rfloor}{1 - 1/\lfloor 2 + \sqrt{k+2} \rfloor} \leq \frac{k + 2 + \sqrt{k+2}}{1 - 1/(1 + \sqrt{k+2})} \enspace .$$

To simplify some calculations, let $\alpha = k + 2$. By substituting α, rearranging terms, and re-substituting α we obtain that the right hand side of the last inequality may be expressed as

$$\frac{\alpha + \sqrt{\alpha}}{1 - 1/(1 + \sqrt{\alpha})} = \frac{(1 + \sqrt{\alpha}) \cdot (\alpha + \sqrt{\alpha})}{\sqrt{\alpha}} = (1 + \sqrt{\alpha})(1 + \sqrt{\alpha}) = (1 + \sqrt{\alpha})^2 = (1 + \sqrt{k+2})^2 \enspace .$$

Thus, the approximation ratio is at most $(1 + \sqrt{k+2})^2$. Suppose that f is monotone so that Proposition 4 guarantees the returned solution is a $(p + 1)$-approximation. Setting the number of solutions to $\ell = 1$ yields $p = k + \ell - 1 = k$, so that the returned set is a $(k+1)$-approximation. This demonstrates that our unified analysis recovers the guarantees of the greedy algorithm for monotone submodular objectives under a k-system constraint.

4. A Nearly Linear Time Implementation

In this section, we present FastSGS, a nearly linear-time variant of SimultaneousGreedys. Recall that SimultaneousGreedys greedily constructs ℓ candidate solutions in a simultaneous fashion. Because the algorithm uses an exact greedy search for the feasible element-solution pair with the largest marginal gain, the overall runtime is $\mathcal{O}(\ell^2 rn)$. Although we consider ℓ to be a constant (as it scales with k), the size of the largest base r could be as large as $\mathcal{O}(n)$. This means that, like other exact greedy approaches, SimultaneousGreedys has a quadratic runtime. In this section, we show that SimultaneousGreedys may be modified to run in nearly linear time by using the thresholding technique of Badanidiyuru and Vondrák (2014) for faster approximate greedy search.

The key idea of FastSGS is to replace the exact greedy search with an approximate greedy search via the use of a marginal gain acceptance threshold: if an element-solution pair is feasible and has a marginal gain which exceeds the threshold, then the update is made without considering other possible pairs. By appropriately initializing and iteratively lowering this marginal gain threshold, we can ensure that the algorithm runs much quicker at the cost of only a small loss in the approximation. The allowed loss in approximation is given as an input parameter $\varepsilon \in (0, 1/2)$ to the algorithm. A formal description of FastSGS appears as Algorithm 3. It begins by initializing the ℓ solutions $S_0^{(1)}, S_0^{(2)}, \dots, S_0^{(\ell)}$ to be empty sets; and the acceptance threshold, denoted by τ, is initially set to be the largest objective value of any element. During each iteration of the while loop, the algorithm iterates once through the set of feasible element-solution pairs. If a feasible element-solution pair is found whose gain exceeds the threshold, then the element is added to that solution. After the completion of each iteration through all the feasible element-solution pairs, the acceptance threshold is reduced by a multiplicative factor of $1 - \varepsilon$, and the algorithm terminates when this threshold becomes sufficiently low.

Algorithm 3: FastSGS $(\mathcal{N}, f, \mathcal{I}, \ell, \varepsilon)$

```
1  Initialize $\ell$ solutions, $S_0^{(j)} \leftarrow \varnothing$ for every $j = 1, \ldots, \ell$.
2  Initialize ground set $\mathcal{N}_0 \leftarrow \mathcal{N}$, and iteration counter $i \leftarrow 1$.
3  Let $\Delta_f = \max_{u \in \mathcal{N}} f(u)$, and initialize threshold $\tau = \Delta_f$.
4  while $\tau > (\varepsilon/n) \cdot \Delta_f$ do
5      for every element-solution pairs $(u, j)$ with $u \in \mathcal{N}_{i-1}$ and $1 \leq j \leq \ell$ do
6          if $S_{i-1}^{(j)} + u \in \mathcal{I}$ and $f(u \mid S_{i-1}^{(j)}) \geq \tau$ then
7              Let $u_i \leftarrow u$ and $j_i \leftarrow j$.
8              Update the solutions as $S_i^{(j)} \leftarrow \begin{cases} S_{i-1}^{(j_i)} + u_i & \text{if } j = j_i \ , \\ S_{i-1}^{(j)} & \text{if } j \neq j_i \ . \end{cases}$
9              Update the available ground set $\mathcal{N}_i \leftarrow \mathcal{N}_{i-1} - u_i$.
10             Update the iteration counter $i \leftarrow i + 1$.
11     Update the marginal gain $\tau \leftarrow (1 - \varepsilon) \cdot \tau$.
12 return the set $S$ maximizing $f$ among the sets $\{S_i^{(j)}\}_{j=1}^{\ell}$.
```

We note that the iteration counter i of FastSGS is used to index the state of the solutions, and does not necessarily correspond to the iterations of any specific loop. We begin our analysis of FastSGS by proving that the number of oracle queries it uses is nearly linear in the number of elements in the ground set.

Observation 19 FastSGS *requires at most $\tilde{\mathcal{O}}(\ell n/\varepsilon)$ calls to the value and independence oracles.*

Proof In every iteration of the while loop, each element-solution pair is considered once, requiring one value query and one independence query. Thus, $\mathcal{O}(\ell n)$ oracle queries are made at each iteration of the while loop. Next, we seek to bound the number of iterations of the while loop. Note that the threshold is initially set to $\tau = \Delta_f$, and is decreased by a multiplicative factor of $1 - \varepsilon$ at each iteration of the while loop. Since the while loop ends once the threshold is below $(\varepsilon/n) \cdot \Delta_f$, the number of iteration of the while loop is the smallest integer a such that $(1 - \varepsilon)^a \cdot \Delta_f \leq (\varepsilon/n) \cdot \Delta_f$. Dividing by Δ_f and taking the $\log_{1-\varepsilon}$ of both sides, we get that a is the smallest integer such that

$$a \geq \log_{1-\varepsilon}(\varepsilon/n) = \frac{\log(\varepsilon/n)}{\log(1-\varepsilon)} = \frac{\log(n/\varepsilon)}{-\log(1-\varepsilon)} \geq \frac{1-\varepsilon}{\varepsilon} \log(n/\varepsilon) \geq \frac{1}{2\varepsilon} \log(n/\varepsilon) \ ,$$

where the penultimate inequality uses $-\varepsilon/(1-\varepsilon) \leq \log(1-\varepsilon) < 0$, which holds for $\varepsilon \in (0, 1)$, and the last inequality follows from our assumption that $\varepsilon < 1/2$. Thus, the number of iterations of the while loops is $\mathcal{O}(1/\varepsilon \cdot \log(n/\varepsilon))$ so that the total number of oracle queries is $\mathcal{O}(\ell n/\varepsilon \cdot \log(n/\varepsilon))$. Using the $\tilde{\mathcal{O}}$ notation that suppresses log factors, the total number of oracle queries becomes $\tilde{\mathcal{O}}(\ell n/\varepsilon)$. ■

Now, we turn to the approximation guarantees of FastSGS. The two theorems below show that FastSGS achieves the same approximation guarantees as Simultaneous-Greedys, but with a multiplicative increase that depends on the error term ε. In this sense, the parameter ε controls the trade-off between the computational cost of the oracle queries and the approximation guarantee.

Theorem 20 *Suppose that $(\mathcal{N}, \mathcal{I})$ is a k-extendible system and that the number of solutions is set to $\ell = k + 1$. Then,* FastSGS *requires $\tilde{\mathcal{O}}(k^2 n/\varepsilon)$ oracle calls and produces a solution whose approximation ratio is at most $(1 - 2\varepsilon)^{-2} \cdot (k+1)^2/k$. Moreover, when f is non-negative monotone submodular and the number of solutions is chosen so $\ell \leq k+1$, then the approximation ratio improves to $(1-\varepsilon)^{-2} \cdot (k+1)$.*

Theorem 21 *Suppose that $(\mathcal{N}, \mathcal{I})$ is k-system and that the number of solutions is set to $\ell = \lfloor 2 + \sqrt{k+2} \rfloor$. Then,* FastSGS *requires $\tilde{\mathcal{O}}(kn/\varepsilon)$ oracle calls and produces a solution whose approximation ratio is at most $(1 - 2\varepsilon)^{-2} \cdot (1 + \sqrt{k+2})^2$. Moreover, when f is non-negative monotone submodular and the number of solutions is set to $\ell = 1$, then the approximation ratio improves to $(1-\varepsilon)^{-2} \cdot (k+1)$.*

The proofs of Theorems 20 and 21 are very similar to their counterparts in Section 3. In particular, the same style of unified meta-proof may be used for analyzing FastSGS. There are, however, two key differences in the analysis when we use the thresholding technique rather than an exact greedy search. The first difference is that rather than a feasible element-solution pair whose marginal gain is maximal, we choose in each iteration a feasible element-solution pair whose marginal gain is within a $(1-\varepsilon)$ multiplicative factor of the largest marginal gain. This $(1-\varepsilon)$ factor carries throughout the analysis. The second difference is that, at the end of the algorithm, there may be elements which are feasible to add to solutions and have positive marginal gain; however, by the termination conditions, the marginal gain of each of these elements is at most $(\varepsilon/n) \cdot \Delta_f$. Using submodularity, we can ensure that leaving these elements behind does not incur a significant loss in the objective value. Formally, one can prove Theorems 20 and 21 by observing that they follow from Proposition 23 (that appears in the next section) by plugging in $m = \rho = 0$ in the same way that Theorems 25 and 26 follow from Proposition 24.

5. Incorporating Knapsack Constraints

In this section, we consider the general form of Problem (2), where the constraint is the intersection of an independence system $\mathcal{I}$ with m knapsack constraints. We present KnapsackSGS, an algorithm which extends the simultaneous greedy technique (Section 3) and the faster thresholding variant (Section 4) to handle knapsack constraints by incorporating a density threshold technique. The density threshold technique we consider was first introduced by Mirzasoleiman et al. (2016) in the context of a repeated-greedy style algorithm for maximizing a submodular function over the intersection of a k-system and m knapsack constraints. By incorporating this density threshold technique into the SimultaneousGreedys framework, we obtain a nearly linear time algorithm which improves both the approximation guarantees and the runtimes of previous methods.

The main idea behind the density threshold technique is to consider adding an element u to a solution S only if the marginal gain is larger than a fixed multiple ρ of the sum of its knapsack weights, i.e., $f(u \mid S) \geq \rho \cdot \sum_{r=1}^{m} c_r(u)$. Here, the quantity $f(u \mid S)/\sum_{r=1}^{m} c_r(u)$ is referred to as the *density* of an element u with respect to a set S. The density threshold technique received its name because it only adds an element to a solution if the density of the element is larger than the threshold ρ. For convenience, given a set S, an element u is said to have high density if its density is larger than (or equal to) ρ and low density if its density is less than ρ.

The algorithm KnapsackSGS is presented below as Algorithm 4. As before, the algorithm begins by initializing ℓ solutions $S_0^{(1)}, S_0^{(2)}, \dots, S_0^{(\ell)}$ to be empty sets. Furthermore, a fast approximate greedy search is again achieved by using a marginal threshold τ which is initially set to Δ_f and then iteratively decreased by a multiplicative factor of $(1-\epsilon)$. The key difference here, compared to FastSGS, is that in order to add an element u to a set $S_{i-1}^{(j)}$, we additionally require that the density ratio $f(u \mid S)/\sum_{r=1}^{m} c_r(u)$ is larger than a fixed threshold ρ and also that the updated set that we are considering, $S_{i-1}^{(j)} + u$, satisfies all the knapsack constraints. For the purposes of analysis, we break these two conditions into separate lines, where the knapsack feasibility condition is checked on its own in Line 7.

Algorithm 4: KnapsackSGS $(\mathcal{N}, f, \mathcal{I}, \ell, \rho, \varepsilon)$

```
1  Initialize $\ell$ solutions, $S_0^{(j)} \leftarrow \varnothing$ for every $j = 1, \dots, \ell$.
2  Initialize ground set $\mathcal{N}_0 \leftarrow \mathcal{N}$, and iteration counter $i \leftarrow 1$.
3  Let $\Delta_f = \max_{u \in \mathcal{N}} f(u)$, and initialize threshold $\tau = \Delta_f$.
4  while $\tau > (\varepsilon/n) \cdot \Delta_f$ do
5      for every element-solution pairs $(u, j)$ with $u \in \mathcal{N}_{i-1}$ and $1 \le j \le \ell$ do
6          if $S_{i-1}^{(j)} + u \in \mathcal{I}$ and $f(u \mid S_{i-1}^{(j)}) \ge \max(\tau, \rho \cdot \sum_{r=1}^{m} c_r(u))$ then
7              if $c_r(S_{i-1}^{(j)} + u) \le 1$ for all $1 \le r \le m$ then
8                  Let $u_i \leftarrow u$ and $j_i \leftarrow j$.
9                  Update the solutions as $S_i^{(j)} \leftarrow \begin{cases} S_{i-1}^{(j_i)} + u_i & \text{if } j = j_i \ , \\ S_{i-1}^{(j)} & \text{if } j \neq j_i \ . \end{cases}$
10                 Update the available ground set $\mathcal{N}_i \leftarrow \mathcal{N}_{i-1} - u_i$.
11                 Update the iteration counter $i \leftarrow i + 1$.
12     Update marginal gain $\tau \leftarrow (1 - \varepsilon) \cdot \tau$.
13 return the set $S$ maximizing $f$ among the sets $\{S_i^{(j)}\}_{j=1}^{\ell}$ and the singletons
   $\{u\}_{u \in \mathcal{N}}$.
```

We begin the study of KnapsackSGS by analyzing its running time. In addition to analyzing the number of calls made to the value and independence oracles, we also analyze the number of arithmetic operations required by KnapsackSGS that arise when working with the knapsack constraints. In many practical scenarios, however, the computational burden of even a few calls to the value oracle is much greater than the total cost of all arithmetic operations required by the knapsack constraints; and thus, the bound on the number of such operations is of less significance.

Observation 22 KnapsackSGS *requires at most $\tilde{\mathcal{O}}(\ell n/\varepsilon)$ calls to the value and independence oracles and $\tilde{\mathcal{O}}(m\ell n/\varepsilon)$ arithmetic operations.*

Proof As shown in Observation 19, there are $\tilde{\mathcal{O}}(1/\varepsilon)$ iterations of the while loop. At each iteration of the while loop, each of the $\mathcal{O}(\ell n)$ element-solution pairs are considered and checking feasibility of each pair requires a single call to the value and independence oracles. Thus, the number of oracle calls is $\tilde{\mathcal{O}}(\ell n/\varepsilon)$.

The arithmetic operations are required for handling the knapsack constraints. Note that for each element $u \in \mathcal{N}$, the term $\rho \cdot \sum_{r=1}^{m} c_r(u)$ can be computed at the beginning of the algorithm using m additions and 1 multiplication. Thus, each of the n terms may be computed using $\mathcal{O}(mn)$ arithmetic operations. If each of the knapsack values $c_r(S^{(i)})$ are maintained for each of the m knapsacks and ℓ solutions, then checking the condition in Line 7 requires $\mathcal{O}(m)$ arithmetic operations. Since this condition is checked for possibly every element-solution pair, this means $\mathcal{O}(m\ell n)$ arithmetic operations per iteration of the while loop. Moreover, since the number of iterations of the while loop is $\tilde{\mathcal{O}}(1/\varepsilon)$, we have that a total number of $\tilde{\mathcal{O}}(m\ell n/\varepsilon)$ arithmetic operations is required. ■

Next, we present a unified analysis of KnapsackSGS which yields approximation guarantees for k-systems and k-extendible systems. At a high level, the analysis is similar to that of Proposition 4. That is, we analyze the elements of the optimal solution OPT which must be thrown away as each new element is added. The key difference here is that we must factor into our analysis the knapsack constraints—which arise via the density threshold criteria in Line 6 and the new feasibility condition in Line 7. It will be beneficial to break up our analysis into two cases based on whether KnapsackSGS returns false on any instance of Line 7 in its execution. Towards this goal, let us introduce some new notation. Let E be an indicator variable for the event that the knapsack check in Line 7 evaluates to false at any point in the algorithm. That is, if $E = 0$ then the "if statement" in Line 7 always evaluates to true; otherwise, $E = 1$ means that it returned false at some point.

Proposition 23 *Suppose that there exists sets $O_i^{(j)}$ for every iteration $0 \leq i \leq T$ and solution $1 \leq j \leq \ell$ and a value p which satisfy the following properties:*

- *$O_0^{(j)} = OPT$ for every solution $1 \leq j \leq \ell$.*
- *$S_i^{(j)} + u \in \mathcal{I}$ for every iteration $0 \leq i \leq T$, solution $1 \leq j \leq \ell$, and element $u \in O_i^{(j)}$.*
- *$O_i^{(j)} \subseteq O_{i-1}^{(j)} \cap \mathcal{N}_i$ for every iteration $1 \leq i \leq T$ and solution $1 \leq j \leq \ell$.*
- *$(S_T^{(j)} \setminus S_i^{(j)}) \cap OPT \subseteq O_i^{(j)}$ for every iteration $0 \leq i \leq T$ and solution $1 \leq j \leq \ell$.*
- *$\sum_{j=1}^{\ell} |O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})| \leq p$ for every iteration $1 \leq i \leq T$.*

Then, the solution S produced by KnapsackSGS *satisfies the following approximation guarantees:*

$$f(S) \geq \begin{cases} \frac{1}{2}\rho & \text{if } E = 1 \ , \\ \frac{1-\epsilon}{p+1} \cdot \Big(\left(1 - \ell^{-1} - \varepsilon\right) f(OPT) - m\rho \Big) & \text{if } E = 0 \ . \end{cases} \tag{7}$$

Moreover, when f is monotone, these approximation guarantees improve to

$$f(S) \geq \begin{cases} \frac{1}{2}\rho & \text{if } E = 1 \ , \\ \frac{1-\epsilon}{p+1} \cdot \Big(\left(1 - \varepsilon\right) f(OPT) - m\rho \Big) & \text{if } E = 0 \ . \end{cases} \tag{8}$$

Proposition 23 provides a guarantee on the solution produced by KnapsackSGS, which depends on the input density threshold ρ and also on the question whether Line 7 ever evaluates to false during the algorithm's execution. The conditions of Proposition 23 are identical to those in Proposition 4 in Section 3, and hence, the previous constructions of these sets $O_i^{(j)}$ for k-systems and k-extendible systems can be used here as well. Note that when $\rho = 0$, then every element has high density and we are back in the setting of SimultaneousGreedys.

The analysis is very similar in spirit to Proposition 4, except that it features the marginal gain threshold technique for faster approximate greedy search and the density ratio threshold technique for knapsack constraints. Because the main proof ideas involving the simultaneous greedy technique are presented in Section 3 and the marginal gain threshold and density ratio threshold techniques already appear in existing works, we defer the proof of Proposition 23 to Appendix A.

Now we address the remaining question, which is how to choose a density threshold ρ which yields a good approximation. Note that we always have either $E = 0$ or $E = 1$, and hence, by taking the minimum of the two lower bounds for the two cases, we obtain the approximation guarantee

$$f(S) \geq \min\left\{\frac{1}{2}\rho, (1-\varepsilon)\left(\frac{1-\ell^{-1}-\varepsilon}{p+1}\right)f(OPT) - \left(\frac{m}{p+1}\right)\rho\right\} . \tag{9}$$

To maximize this lower bound, we would like to set the density ratio threshold to

$$\rho^* = 2(1-\varepsilon)\left(\frac{1-\ell^{-1}-\varepsilon}{p+1+2m}\right)f(OPT) \ ,$$

which would yield an approximation guarantee of

$$f(S) \geq (1-\varepsilon)\left(\frac{1-\ell^{-1}-\varepsilon}{p+1+2m}\right)f(OPT) \ .$$

Unfortunately, we cannot efficiently calculate this optimal choice for density ratio threshold ρ^* because it involves $f(OPT)$. Nevertheless, by the submodularity and non-negativity of the objective, we know that the optimal value lies within the range

$$\Delta_f \leq f(OPT) \leq r \cdot \Delta_f \ ,$$

where we recall that r is the size of the largest independent set. This interval, which is guaranteed to contain $f(OPT)$, can be transformed into an interval containing the optimal density threshold ρ^*. In particular, we get

$$\rho^* = 2(1-\varepsilon)\left(\frac{1-\ell^{-1}-\varepsilon}{p+1+2m}\right)\Delta_f \cdot \alpha \ ,$$

for some $\alpha \in [1, r]$. When r is not known exactly, an upper bound may be used here. One upper bound we can use is that for any base $B \in \mathcal{I}$, r is at most $k \cdot |B|$, which follows by the definition of a k-system. Such a base B may be known beforehand or constructed in a greedy fashion using $\mathcal{O}(n)$ calls to the independence oracle. However, for simplicity, we use the somewhat weaker upper bound of $r \leq n$. Using the above mentioned stronger upper bound, or an instance specific upper bound, will reduce the interval in question, and thus, also the runtime. However, the improvement will only be in the logarithmic component of the runtime.

The high level idea is to design an algorithm which calls KnapsackSGS several times as a subroutine using various density ratio thresholds in this range and to return the best solution. Mirzasoleiman et al. (2016) propose using a multiplicative grid search over this interval, running the algorithm on each point in the interval. The multiplicative grid search guarantees that the subroutine algorithm is run with an input density threshold ρ which is close to the optimal ρ^* in the sense that $(1-\delta)\rho^* \leq \rho \leq \rho^*$ (for some error parameter $\delta \in (0, 1/2)$). One may verify that by using this "approximately-optimal" density threshold, the approximation ratio obtained by the lower bound (9) is at most a factor $(1-\delta)^{-1}$ larger than if the optimal threshold ρ^* were used. This multiplicative grid search approach requires running the subroutine on every point in the multiplicative grid, which translates to $\mathcal{O}(1/\delta \cdot \log(n))$ calls to the subroutine. Thus, this "brute force" multiplicative grid search adds an additional $\tilde{\mathcal{O}}(1/\delta)$ factor to the running time, which is undesirable, especially for higher accuracy applications where a smaller δ is preferred.

We propose a binary search method which achieves the same approximation guarantee using exponentially fewer calls to KnapsackSGS as a subroutine. The key to our binary search method is a careful use of the case analysis in Proposition 23. The algorithm DensitySearchSGS is stated formally below as Algorithm 5. We consider points on a multiplicatively spaced grid of the interval $[1, n]$, which is given by $\alpha_k = (1+\delta)^k$ for $k = 0, 1, \ldots, \left\lceil \frac{1}{\delta} \log n \right\rceil$, where δ is an input parameter that specifies the granularity of the grid. Another input to the algorithm is β, which specifies the relation between the points in the grid $[1, n]$ and the density thresholds which are used. Let us consider the non-monotone case for now, in which case we should set

$$\beta = 2(1-\varepsilon)\left(\frac{1-\ell^{-1}-\varepsilon}{p+1+2m}\right).$$

In this case, note that each point α_k in the $[1, n]$ grid corresponds to the choice of density threshold

$$\rho_k = \beta \cdot \Delta_f \cdot \alpha_k = 2(1-\varepsilon)\left(\frac{1-\ell^{-1}-\varepsilon}{p+1+2m}\right)\Delta_f \cdot \alpha_k.$$

The algorithm tries to zoom in on the optimal density threshold using binary search, while using the value of the indicator E for each call to KnapsackSGS to make the decision in each iteration of the search (we denote by E_i the value of this indicator for call number i). While this indicator does not necessarily indicate the relationship between the the current density threshold and ρ^*, it does give enough of a signal around which we may construct a binary search. In particular, if $E_i = 0$, then we get a good approximation as long as our current density threshold is an overestimate of ρ^*, and thus, in the future we only need to consider higher density thresholds. Likewise, if $E_i = 1$, then we get a good approximation as long as our current density threshold is an underestimate of ρ^*, and thus, in the future we only need to consider lower density thresholds.

Algorithm 5: DensitySearchSGS $(\mathcal{N}, f, \mathcal{I}, \ell, \delta, \varepsilon, \beta)$

```
1  Initialize upper and lower bounds $k_\ell = 1$, $k_u = \lceil \frac{1}{\delta} \log n \rceil$.
2  Let $\Delta_f = \max_{u \in \mathcal{N}} f(u)$, and initialize iteration counter $i \leftarrow 1$.
3  while $|k_u - k_\ell| > 1$ do
4      Set middle bound $k_i = \left\lceil \frac{k_\ell + k_u}{2} \right\rceil$.
5      Set density ratio $\rho_i \leftarrow \beta \cdot \Delta_f (1+\delta)^{k_i}$.
6      Obtain set $S_i \leftarrow$ KnapsackSGS$(\mathcal{N}, f, \mathcal{I}, \ell, \rho_i, \varepsilon)$.
7      if $E_i = 0$ then
8          Increase lower bound $k_\ell \leftarrow k_i$.
9      else
10         Decrease upper bound $k_u \leftarrow k_i$.
11     Update iteration counter $i \leftarrow i + 1$.
12 Set density ratio $\rho_i \leftarrow \beta \cdot \Delta_f (1+\delta)^{k_\ell}$.
13 Obtain set $S_i \leftarrow$ KnapsackSGS$(\mathcal{N}, f, \mathcal{I}, \ell, \rho_i, \varepsilon)$.
14 return the set $S$ maximizing $f$ among the sets $\{S_j\}_{j=1}^{i}$.
```

The following proposition bounds the number of calls made to KnapsackSGS and provides an approximation guarantee.

Proposition 24 DensitySearchSGS *makes* $\tilde{\mathcal{O}}(1)$ *calls to* KnapsackSGS. *Additionally assume that the independence system satisfies the conditions in Proposition 23 for every execution of* KnapsackSGS. *If* $\beta = 2(1-\varepsilon)\left(\frac{1-\ell^{-1}-\varepsilon}{p+1+2m}\right)$, *then the solution* S *returned by* DensitySearchSGS *satisfies*

$$f(S) \geq (1-\delta)(1-\varepsilon)\left(\frac{1-\ell^{-1}-\varepsilon}{p+1+2m}\right) f(OPT) \geq (1-\delta)(1-2\varepsilon)^2\left(\frac{1-\ell^{-1}}{p+1+2m}\right) f(OPT)$$

when the number of solutions ℓ *is at least 2. Moreover, if* f *is monotone and* $\beta = 2(1-\varepsilon)\left(\frac{1-\varepsilon}{p+1+2m}\right)$ *then this lower bound further improves to*

$$f(S) \geq (1-\delta)(1-\varepsilon)^2\left(\frac{1}{p+1+2m}\right) f(OPT)$$

for any number of solutions ℓ.

Remark: Observe that Proposition 24 requires a different value for β in the cases of monotone and non-monotone functions. This is necessary because the ρ^* corresponding to these two cases are different, and the value of β is used to adjust the range in which Algorithm 5 searches for a density approximating ρ^* so that this range is guarantee to include ρ^*. One can avoid this by slightly increasing the range in which Algorithm 5 searches so that it is guaranteed to include both possible values for ρ^*. Since the ratio between the values of ρ^* corresponding to the two cases is only a constant as long as the sum $\ell^{-1} + \varepsilon$ is bounded away from 1, such an expansion of the search range will have an insignificant effect on the time complexity of the algorithm in most regimes of interest.

Proof *(Proof of Proposition 24)*: We begin by bounding the number of calls that DensitySearchSGS makes to KnapsackSGS. Recall that the number of points in the δ-multiplicative discretization is $\mathcal{O}(1/\delta \cdot \log n)$. At each iteration of the binary search, KnapsackSGS is called once. It is well known that binary search requires only logarithmically many iterations to terminate. Thus, the number of calls to KnapsackSGS is $\mathcal{O}(\log(1/\delta \cdot \log n)) = \mathcal{O}(\log(1/\delta) + \log\log n) = \tilde{\mathcal{O}}(1)$ calls to KnapsackSGS.

Now we prove the approximation guarantee of DensitySearchSGS using the approximation guarantees of KnapsackSGS. Let us first consider the general non-monotone case when $\beta = 2(1-\varepsilon)\left(\frac{1-\ell^{-1}-\varepsilon}{p+1+2m}\right)$. We proceed by a case analysis. For the first case, suppose that at some iteration i of DensitySearchSGS, it called KnapsackSGS with a density threshold ρ_i such that $\rho_i \le \rho^*$ and the indicator E_i ended up with the value 0. By Proposition 23, we get in this case

$$\begin{aligned} f(S) &\ge f(S_i) \\ &\ge \frac{1-\varepsilon}{p+1}\cdot\Big(\left(1-\ell^{-1}-\varepsilon\right)f(OPT) - m\rho_i\Big) \\ &\ge \frac{1-\varepsilon}{p+1}\cdot\Big(\left(1-\ell^{-1}-\varepsilon\right)f(OPT) - m\rho^*\Big) \\ &= (1-\varepsilon)\left(\frac{1-\ell^{-1}-\varepsilon}{p+1+2m}\right)f(OPT) \ . \end{aligned}$$

The second case is that at some iteration i of DensitySearchSGS, it called KnapsackSGS with a density threshold ρ_i such that $\rho_i \ge \rho^*$ and the indicator E_i ended up with the value 1. By Proposition 23, we get in this case

$$f(S) \ge f(S_i) \ge \frac{1}{2}\rho_i \ge \frac{1}{2}\rho^* = (1-\varepsilon)\left(\frac{1-\ell^{-1}-\varepsilon}{p+1+2m}\right)f(OPT) \ .$$

The last case we need to consider is the case that neither of the above cases happens in any iteration. One can observe that in this case the binary search of DensitySearchSGS chooses in each iteration the half of its current range that includes ρ^*. Thus, we have

$$2(1-\varepsilon)\left(\frac{1-\ell^{-1}-\varepsilon}{p+1+2m}\right)\Delta_f(1+\delta)^{k_\ell} \le \rho^* \le 2(1-\varepsilon)\left(\frac{1-\ell^{-1}-\varepsilon}{p+1+2m}\right)\Delta_f(1+\delta)^{k_u} \ .$$

Let us now denote the final value of i by $\hat{\imath}$. Since the leftmost side of the last inequality is equal to $\rho_{\hat{\imath}}$ and the rightmost side is larger than the leftmost side by at most a factor $1+\delta$ (since $k_u - k_\ell \le 1$ when DensitySearchSGS terminates), we get $\rho_{\hat{\imath}} \le \rho^* \le (1+\delta)\rho_{\hat{\imath}}$, and by Proposition 23,

$$\begin{aligned}
f(S) &\geq f(S_{\hat{\imath}}) \\
&\geq \min\left\{\frac{1}{2}\rho_{\hat{\imath}}, \frac{1-\epsilon}{p+1}\cdot\Big(\left(1-\ell^{-1}-\varepsilon\right)f(OPT) - m\rho_{\hat{\imath}}\Big)\right\} \\
&\geq \min\left\{\frac{1}{2}\frac{\rho^*}{1+\delta}, \frac{1-\epsilon}{p+1}\cdot\Big(\left(1-\ell^{-1}-\varepsilon\right)f(OPT) - m\rho^*\Big)\right\} \\
&\geq (1-\delta)\min\left\{\frac{1}{2}\rho^*, \frac{1-\epsilon}{p+1}\cdot\Big(\left(1-\ell^{-1}-\varepsilon\right)f(OPT) - m\rho^*\Big)\right\} \\
&= (1-\delta)(1-\varepsilon)\left(\frac{1-\ell^{-1}-\varepsilon}{p+1+2m}\right)f(OPT) \enspace ,
\end{aligned}$$

which establishes the first inequality in the statement of the proposition. We establish the second inequality by observing that for $\ell \geq 2$, the quantity $-\varepsilon + 2\varepsilon\ell^{-1}$ is non-positive; and thus,

$$1-\ell^{-1}-\varepsilon \geq 1-\ell^{-1}-2\varepsilon+2\varepsilon\ell^{-1} = (1-\ell^{-1})(1-2\varepsilon) \enspace ,$$

which establishes the proposition for non-monotone objectives. The analysis for monotone objectives follows in an analogous manner. ■

We are now ready to present the main approximation results for DensitySearchSGS when the independence system is either a k-system or a k-extendible system.

Theorem 25 *Suppose that $(\mathcal{N}, \mathcal{I})$ is a k-extendible system, the number of solutions is set to $\ell = M+1$—where $M = \max\left(\lceil\sqrt{1+2m}\rceil, k\right)$, and the two error terms are set to be equal (i.e., $\varepsilon = \delta \in (0, 1/2)$). Then,* DensitySearchSGS *requires $\tilde{O}(Mn/\varepsilon)$ oracle calls as well as $\tilde{O}(Mmn/\varepsilon)$ arithmetic operations and produces a solution whose approximation ratio is at most*

$$(1-2\varepsilon)^{-3}\left[\max\left\{k+\frac{2m+1}{k}, 1+2\sqrt{2m+1}\right\}+2m+2\right] \enspace .$$

Moreover, when f is non-negative monotone submodular and the number of solutions is chosen so that $\ell \leq k+1$, then the approximation ratio improves to $(1-\varepsilon)^{-3}\cdot(k+2m+1)$.

Theorem 26 *Suppose that $(\mathcal{N}, \mathcal{I})$ is k-system, the number of solutions is $\ell = \lfloor 2+\sqrt{k+2m+2}\rfloor$, and the two error terms are set equal as $\varepsilon = \delta \in (0, 1/2)$. Then,* DensitySearchSGS *requires $\tilde{O}(n\sqrt{k+m}/\varepsilon)$ oracle calls as well as $\tilde{O}(mn\sqrt{k+m}/\varepsilon)$ arithmetic operations and produces a solution whose approximation ratio is at most $(1-2\varepsilon)^{-3}\cdot(1+\sqrt{k+2m+2})^2$. Moreover, when f is non-negative monotone submodular and the number of solutions is set to $\ell = 1$, then the approximation ratio improves to $(1-\varepsilon)^{-3}\cdot(k+2m+1)$.*

As for the previous algorithms, the approximation ratio guaranteed by DensitySearchSGS is improved for the subclass of k-extendible systems, at the cost of a slightly larger running time. The algorithm guarantees the same approximation factor for monotone objectives for both k-extendible and k-systems. In both cases, the best choice of the number of solutions ℓ depends on the number of knapsack constraints m. Moreover, we remark that in the absence of any additional knapsack constraints (that is, $m = 0$), the approximation guarantees of Theorems 25 and 26 recover the guarantees of the slower SimultaneousGreedys, up to the $(1-\varepsilon)^{-3}$ error terms. However, unlike SimultaneousGreedys, the running time of DensitySearchSGS is nearly-linear in the size of the ground set.

The $(1-2\varepsilon)^{-3}$ multiplicative error terms may seem somewhat non-intuitive at first glance, but it turns out that they can be replaced with $1+\mathcal{O}(\varepsilon)$. In particular, for $\varepsilon \in (0, 1/4)$ the multiplicative error term $(1-2\varepsilon)^{-3}$ is at most $1+28\varepsilon$. This follows by the convexity of the function $y(t) = (1-2t)^{-3}$ within the range $[0, 1/2)$. More specifically, by setting $\lambda = 4\varepsilon \in (0,1)$, we get

$$\begin{aligned}(1-2\varepsilon)^{-3} &= y(\varepsilon) = y\left((1-\lambda)\cdot 0 + \lambda \cdot 1/4\right) \leq (1-\lambda)\cdot y(0) + \lambda \cdot y(1/4) \\ &= (1-4\varepsilon) + 4\varepsilon\left(1 - 2\cdot\frac{1}{4}\right)^{-3} = (1-4\varepsilon) + 4\varepsilon\cdot 8 = 1+28\varepsilon \ .\end{aligned}$$

Similarly, one can also show that for all $\varepsilon \in (0, 1/4)$, the error term $(1-\varepsilon)^{-3}$, which appears in the approximation for monotone submodular objectives, is at most $(1+6\varepsilon)$. Furthermore, by scaling ε one may transfer the constant in front of ε to the running time, which remains $\tilde{\mathcal{O}}(n/\varepsilon)$. This way, one may consider the multiplicative error term in the approximation factor of the algorithm to be a clean $1+\varepsilon$.

The proofs of Theorems 25 and 26 follow from the unified meta-analysis of Proposition 24 in the same way that the Theorems 2 and 3 follow from the meta-analysis of Proposition 4. Namely, the constructions of the sets $O_i^{(j)}$ in Sections 3.2 and 3.3 demonstrate that the conditions of Proposition 24 hold with $p = \max(k, \ell-1)$ for k-extendible systems and $p = k+\ell-1$ for general k-systems. The final step is then to choose the number of solutions ℓ to optimize the resulting approximation ratios. Although these steps are conceptually similar to the choice of ℓ in the analysis of SimultaneousGreedys, they are somewhat involve, and so we reproduce them here.

Proof *(Proof of Theorem 25)*: The construction of sets $O_i^{(j)}$ in Proposition 8 demonstrates that the conditions of Proposition 23 are satisfied with with $p = \max(k, \ell-1)$. Thus, Proposition 24 implies that the approximation ratio of DensitySearchSGS with $\ell = M+1$ is at most $(1-2\varepsilon)^{-3}$ times the quantity,

$$\frac{p+1+2m}{1-\ell^{-1}} = \frac{\max(k,\ell-1)+1+2m}{1-\ell^{-1}} \ .$$

Trying to optimize this quantity, we may set $\ell = M+1$, where $M = \max\left(\lceil\sqrt{1+2m}\rceil, k\right)$. Note that M is at least k and so $\max(k, \ell-1) = \max(k, M) = M$. Thus, by the above, we have that the approximation ratio of DensitySearchSGS with $\ell = M+1$ is at most

$(1-2\varepsilon)^{-3}$ times the quantity

$$\begin{aligned}\frac{M+1+2m}{1-\frac{1}{M+1}} &= \frac{(M+1)\,(M+1+2m)}{M} \\ &= \frac{(M+1)^2}{M} + \left(\frac{M+1}{M}\right)2m \\ &\le \max\left\{k+2m+2+\frac{2m+1}{k}, 2m+3+2\sqrt{2m+1}\right\} \\ &= 2m+2+\max\left\{k+\frac{2m+1}{k}, 1+2\sqrt{2m+1}\right\} \ .\end{aligned}$$

For monotone submodular objectives, Proposition 24 implies that for all number of solutions $\ell \le k+1$, the approximation ratio is at most $(1-\varepsilon)^{-3}$ times the quantity

$$p+1+2m \le \max(k, \ell-1)+1+2m \le k+1+2m \ . \qquad \blacksquare$$

Proof *(Proof of Theorem 26)*: The construction of sets $O_i^{(j)}$ in Proposition 13 demonstrates that the conditions of Proposition 23 are satisfied with with $p = k+\ell-1$. Thus, Proposition 24 implies that the approximation ratio of DensitySearchSGS with $\ell = \lfloor 2+\sqrt{k+2m+2}\rfloor$ is at most $(1-2\varepsilon)^{-3}$ times the quantity,

$$\frac{p+1+2m}{1-\ell^{-1}} = \frac{k+\ell+2m}{1-\ell^{-1}} = \frac{k+2m+\lfloor 2+\sqrt{k+2m+2}\rfloor}{1-1/\lfloor 2+\sqrt{k+2m+2}\rfloor} \le \frac{k+2m+2+\sqrt{k+2m+2}}{1-1/(1+\sqrt{k+2m+2})} .$$

To simplify some calculations, let $\alpha = k+2m+2$. By substituting α and rearranging terms, we obtain that the right hand side may be expressed as

$$\frac{\alpha+\sqrt{\alpha}}{1-1/(1+\sqrt{\alpha})} = \frac{(1+\sqrt{\alpha})\cdot(\alpha+\sqrt{\alpha})}{\sqrt{\alpha}} = (1+\sqrt{\alpha})(1+\sqrt{\alpha}) = (1+\sqrt{\alpha})^2 \ .$$

Substituting back the value of $\alpha = k+2m+2$, we have that the approximation ratio is at most $(1-2\varepsilon)^{-3}\cdot(1+\sqrt{k+2m+2})^2$. Finally, suppose that f is monotone and $\ell = 1$. Then, by Proposition 24, the approximation factor is at most

$$(1-\varepsilon)^{-3}\cdot(p+2m+1) = (1-\varepsilon)^{-3}\cdot(k+\ell-1+2m+1) = (1-\varepsilon)^{-3}\cdot(k+2m+1) \ . \ \blacksquare$$

6. Repeated Greedy

In this section, we present and analyze the RepeatedGreedy algorithm for maximizing a submodular function f subject to a k-system constraint $(\mathcal{N}, \mathcal{I})$. As discussed in Section 1, RepeatedGreedy was first proposed by Gupta et al. (2010), who showed that $\mathcal{O}(k)$ repeated iterations of the greedy procedure suffice to achieve a $3k$ approximation guarantee.

This approximation guarantee was improved to $2k$ by Mirzasoleiman et al. (2016). In this section, we demonstrate that $\mathcal{O}(\sqrt{k})$ iterations of the RepeatedGreedy algorithm suffice to achieve a $k + \mathcal{O}(\sqrt{k})$ approximation guarantee, which improves both the runtime and approximation guarantees.

RepeatedGreedy iteratively executes the following three operations: first, the greedy algorithm is called as a subroutine to produce a feasible set S_i, then a subroutine for unconstrained submodular maximization produces a set $S'_i \subset S_i$ with large objective value, and elements of S_i are removed from the remaining ground set. After the algorithm makes ℓ iterations of this kind, the algorithm terminates and outputs the best set among all the sets constructed during its iterations. The choice of ℓ will be determined later to yield the best approximation ratio. A more formal description of RepeatedGreedy is given as Algorithm 7.

Algorithm 6: Greedy $(\mathcal{N}, f, \mathcal{I})$

```
1 $S \leftarrow \varnothing$
2 while there exists $u \in \mathcal{N}'$ such that $S + u \in \mathcal{I}$ and $f(u \mid S) > 0$ do
3     Let $u \in \mathcal{N}'$ be the element of this kind maximizing $f(u \mid S)$.
4     Add $u$ to $S$.
5 return $S$.
```

Algorithm 7: Repeated Greedy$(\mathcal{N}, f, \mathcal{I}, \ell)$

```
1 Let $\mathcal{N}_1 \leftarrow \mathcal{N}$.
2 for $i = 1$ to $\ell$ do
3     Run greedy procedure $S_i \leftarrow$ Greedy$(\mathcal{N}_i, f, \mathcal{I})$
4     Filter the greedy solution $S'_i \leftarrow \text{USM}(S_i, f)$
5     Update ground set $\mathcal{N}_{i+1} \leftarrow \mathcal{N}_i \setminus S_i$.
6 return the set $S$ maximizing $f$ among the sets $\{S_i, S'_i\}_{i=1}^{\ell}$.
```

RepeatedGreedy calls a subroutine USM for unconstrained submodular maximization. Formally, the subroutine $\text{USM}(A, f)$ takes as input a set A and a non-negative submodular function f defined on subsets of A and returns a set $X \subset A$ such that $f(X) \geq \frac{1}{\alpha} f(B)$ for all $B \subset A$. There are several known algorithms for USM (Feige et al., 2007; Gharan and Vondrak, 2011; Buchbinder et al., 2015). Feige et al. (2007) showed that no algorithm using only polynomially many oracle queries can achieve an approximation ratio smaller than $\alpha = 2$. For the sake of generality, we remain agnostic to the specific USM subroutine that is being used and derive an approximation ratio for RepeatedGreedy that depends on α; however, in order to obtain nearly-linear time algorithms, we restrict our attention to USM subroutines which require at most $\mathcal{O}(n)$ oracle calls. In the spirit of proposing deterministic algorithms, we further restrict our attention to deterministic USM subroutines, although a randomized subroutine may be used here as well, with appropriate probabilistic caveats in the approximation ratio. At the time of this writing, it is most natural to use the deterministic algorithm of Buchbinder et al. (2015), which yields an approximation ratio of $\alpha = 3$ and runs in linear time. We remark there that it is an interesting open problem to construct a deterministic linear time algorithm for USM which achieves the optimal $\alpha = 2$ approximation, although Buchbinder and Feldman (2018b) come close to achieving this goal. Specifically, they designed an algorithm for USM achieving $(2+\varepsilon)$-approximation using $O(n/\varepsilon)$ time.

Observation 27 RepeatedGreedy *requires* $\mathcal{O}(\ell r n)$ *oracle calls and its output* S *is independent.*

Proof We begin the proof by bounding the number of oracle calls used by RepeatedGreedy. Observe that Greedy has at most $|S| \leq r$ iterations, during which at most n calls to the value and independence oracle calls are made. This means that a single execution of Greedy requires $\mathcal{O}(rn)$ oracle calls. As discussed above, we only consider implementations of USM which require $\mathcal{O}(n)$ oracle calls, which is negligible compared to the computational requirements of Greedy. Finally, because ℓ solutions are produced by RepeatedGreedy (and thus, the algorithm makes only ℓ iterations), the total number of oracle calls is $\mathcal{O}(\ell r n)$.

Next, we prove that the output S is independent. For every $1 \leq i \leq \ell$, the set S_i is initialized as independent because the greedy algorithm returns an independent set. Moreover, the output S'_i of USM satisfies $S'_i \subseteq S_i$, and thus, S'_i is independent by the down-closed property of k-systems. The observation now follows since the output S of RepeatedGreedy is chosen as either S_i or S'_i for one of such i. ■

We now present the main runtime and approximation guarantees for RepeatedGreedy.

Theorem 28 *Suppose that* $(\mathcal{N}, \mathcal{I})$ *is* k*-system and that the number of solutions is set to* $\ell = \lfloor 1 + \sqrt{2(k+1)/\alpha} \rfloor$*. Then,* RepeatedGreedy *requires* $\mathcal{O}(\sqrt{k} r n)$ *oracle calls and produces a solution whose approximation ratio is at most* $k + \left(\sqrt{2\alpha}\right)\sqrt{k} + (\alpha+1) + \mathcal{O}(\sqrt{\alpha^3/k})$*. Moreover, when* f *is a non-negative monotone submodular function and the number of solutions is set to* $\ell = 1$*, then the approximation ratio of* RepeatedGreedy *improves to* $k+1$*.*

Observe that compared to the prior line of work by Gupta et al. (2010) and Mirzasoleiman et al. (2016) which demonstrated that RepeatedGreedy achieves $2k$-approximation after producing $\ell = k$ solutions, Theorem 28 improves upon both the runtime and the approximation guaranatees. In particular, only $\ell = \mathcal{O}(\sqrt{k})$ solutions are required, and the approximation guarantee is improved to $k + \mathcal{O}(\sqrt{k})$. Interestingly, we show in Section 6.1 that this analysis is tight even for the class of k-extendible systems. This stands in sharp contrast to SimultaneousGreedys, which is able to achieve an improved $k + \mathcal{O}(1)$ approximation for the subclass of k-extendible systems.

Furthermore, although both SimultaneousGreedys and RepeatedGreedy achieve similar asymptotic approximation factors of $k + \mathcal{O}(\sqrt{k})$ for the class of k-systems, the low order terms in the approximation factor of RepeatedGreedy are larger. More precisely, RepeatedGreedy achieves an approximation of $k + (\sqrt{2\alpha})\sqrt{k} + (1+\alpha) + \mathcal{O}(\sqrt{\alpha^3/k})$ while SimultaneousGreedys achieves an approximation of $(1 + \sqrt{k+2})^2 = k + 2\sqrt{k} +$

$3 + \mathcal{O}(\sqrt{1/k})$. While both algorithms have the same coefficient for the leading k term, RepeatedGreedy has larger coefficients in the low order terms. In particular, the lower order terms of RepeatedGreedy depend on α, the approximation ratio for USM, which will be at least 2 by the hardness result of Feige et al. (2007). Moreover, even the smallest $\mathcal{O}(\sqrt{\alpha^3/k})$ term is larger for RepeatedGreedy. While this term goes to zero as k grows, it may be non-negligible for very small k values. An explicit form for this term is derived in the proof of Theorem 28. After the proof, we analyze this term more carefully, showing that for $\alpha = 3$ and $k = 1$ it is ≈ 20.4 and for $k = 10$ the term is ≈ 2.1. On the other hand, the $\mathcal{O}(\sqrt{1/k})$ term in the approximation ratio of SimultaneousGreedys is at most $2/\sqrt{k}$.

We now begin the analysis of the approximation ratio of RepeatedGreedy for k-systems. As before, we write OPT to denote an independent set of $(\mathcal{N}, \mathcal{I})$ maximizing f. At a high level, our analysis proceeds by showing that (1) by properties of the Greedy and USM procedures, the value of the output of RepeatedGreedy is proportional to the average value of the union between a set from $\{S\}_{i=1}^{\ell}$ and OPT, and then (2) because the sets $\{S\}_{i=1}^{\ell}$ are disjoint, this aforementioned average cannot be considerably smaller than the value of OPT. More concretely, our analysis is based on three lemmata. The first of these lemmata, presented below, gives lower bounds on the objective values of the two sets S_i and S'_i produced at each iteration.

Lemma 29 *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 $f(S'_i) \geq \frac{1}{\alpha} f(S_i \cap OPT)$.*

Proof The first inequality is a direct application of Lemma 3.2 of Gupta et al. (2010), which states that a set 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. Notice that 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 $\mathcal{N}_i$ and that $C = OPT \cap \mathcal{N}_i$ is an independent set of this restricted k-system. This yields that $f(S_i) \geq \frac{1}{k+1} f(S_i \cup (OPT \cap \mathcal{N}_i))$.

Let us now explain why the second inequality of the lemma holds. Observe that $S_i \cap OPT$ is a subset of S_i. Thus, by the approximation guarantees of USM, $f(S'_i) \geq \frac{1}{\alpha} f(S_i \cap OPT)$. ■

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

Lemma 30 *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

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

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

The third lemma we need is Lemma 7 [Lemma 2.2 of Buchbinder et al. (2014)] which allows us to relate the average value of $f(S_i \cup OPT)$ to the optimal value $f(OPT)$. Together, these allow us to prove Proposition 31, which is a general approximation guarantee for RepeatedGreedy that holds for any number of iterations $\ell \geq 1$.

Proposition 31 *If $(\mathcal{N}, \mathcal{I})$ is a k-system, then the solution returned by* RepeatedGreedy *has an approximation ratio of at most $\frac{k+1+\frac{\alpha}{2}(\ell-1)}{1-1/\ell}$. Moreover, this approximation improves to $k+1+\frac{\alpha}{2}(\ell-1)$ for monotone submodular objectives.*

Proof Observe that, for every $1 \leq i \leq \ell$, we have

$$OPT \setminus \mathcal{N}_i = OPT \cap (\mathcal{N} \setminus \mathcal{N}_i) = OPT \cap \left(\cup_{j=1}^{i-1} S_i\right) = \cup_{j=1}^{i-1} \left(OPT \cap S_j\right) \tag{10}$$

where the first equality holds because $OPT \subseteq \mathcal{N}$, and the second equality follows from the removal of S_i from the ground set in each iteration of RepeatedGreedy. Using the previous lemmata and this observation, we can obtain a lower bound on the objective value of the returned solution S in terms of the average value of $f(S_i \cup OPT)$ as

$$\begin{aligned}
\frac{1}{\ell}\sum_{i=1}^{\ell} f(S_i \cup OPT) &\leq \frac{1}{\ell}\sum_{i=1}^{\ell} f(S_i \cup (OPT \cap \mathcal{N}_i)) + \frac{1}{\ell}\sum_{i=1}^{\ell} f(OPT \setminus \mathcal{N}_i) && \text{(Lemma 30)}\\
&= \frac{1}{\ell}\sum_{i=1}^{\ell} f(S_i \cup (OPT \cap \mathcal{N}_i)) + \frac{1}{\ell}\sum_{i=1}^{\ell} f\left(\cup_{j=1}^{i-1}(OPT \cap S_j)\right) && \text{(Equality (10))}\\
&\leq \frac{1}{\ell}\sum_{i=1}^{\ell} f(S_i \cup (OPT \cap \mathcal{N}_i)) + \frac{1}{\ell}\sum_{i=1}^{\ell}\sum_{j=1}^{i-1} f(OPT \cap S_j) && \text{(submodularity)}\\
&\leq \frac{k+1}{\ell}\sum_{i=1}^{\ell} f(S_i) + \frac{\alpha}{\ell}\sum_{i=1}^{\ell}\sum_{j=1}^{i-1} f(S'_j) && \text{(Lemma 29)}\\
&\leq \frac{k+1}{\ell}\sum_{i=1}^{\ell} f(S) + \frac{\alpha}{\ell}\sum_{i=1}^{\ell}\sum_{j=1}^{i-1} f(S) && \text{(definition of } S\text{)}\\
&= [k+1+\alpha(\ell-1)/2]\, f(S) \enspace .
\end{aligned}$$

Rearranging this inequality yields the following lower bound on the value of the returned solution.

$$f(S) \geq \frac{1}{k+1+\frac{\alpha}{2}(\ell-1)} \cdot \frac{1}{\ell}\sum_{i=1}^{\ell} f(S_i \cup OPT) \enspace . \tag{11}$$

In order to remove the dependence of the right hand side on the solutions S_i, we again use Lemma 7 [Lemma 2.2 of Buchbinder et al. (2014)]. In particular, consider a set $\bar{S}$ chosen uniformly at random from the ℓ constructed solutions $S_1, S_2, \dots S_\ell$. Because these solutions are disjoint by construction, an element can belong to $\bar{S}$ with probability at most ℓ^{-1}. Hence, applying Lemma 7 to the submodular function $g(S) = f(OPT \cup S)$, we get

$$\frac{1}{\ell}\sum_{i=1}^{\ell} f(S_i \cup OPT) = \mathbb{E}[f(OPT \cup \bar{S})] = \mathbb{E}[g(\bar{S})] \geq (1-\ell^{-1}) \cdot g(\varnothing) = (1-\ell^{-1}) \cdot f(OPT) \enspace . \tag{12}$$

Substituting (12) into the lower bound of (11) yields the desired approximation.

When f is monotone submodular, we may obtain an improved approximation ratio by applying monotonicity directly to the lower bound (11). In particular, applying monotonicity yields

$$\frac{1}{\ell}\sum_{i=1}^{\ell} f(S_i \cup OPT) \geq \frac{1}{\ell}\sum_{i=1}^{\ell} f(OPT) = f(OPT) \ ,$$

which yields the desired approximation in the monotone setting. ■

Note that the approximation factor derived in Proposition 31 is not a monotone function of the number of iterations ℓ in RepeatedGreedy. In this sense, we can optimize this derived approximation factor by choosing some appropriate value of ℓ. On the other hand, the true approximation factor of RepeatedGreedy can only increase as the number of iterations increases, as more solutions are produced. Thus, the non-monotonicity of our derived approximation factor in ℓ should be regarded as an artifact of our analysis and not as the actual behavior of RepeatedGreedy.

Nevertheless, we may derive bounds on the approximation factor of RepeatedGreedy for k-systems when the number of iterations is set as $\ell = \mathcal{O}(\sqrt{k})$. In particular, setting ℓ to minimize the approximation factor presented in Proposition 31 yields the approximation guarantee of Theorem 28.

Proof *(Proof of Theorem 28)*: First, we show that the number of oracle calls is at most $\mathcal{O}(\sqrt{k}rn)$. By Observation 27, the number of oracle calls is at most $\mathcal{O}(\ell rn)$ and so the result follows from our choice of the number of solutions, $\ell = \lfloor 1 + \sqrt{2(k+1)/\alpha} \rfloor = \mathcal{O}(\sqrt{k})$.

Next, we show that setting the number of solutions to $\ell = \lfloor 1 + \sqrt{2(k+1)/\alpha} \rfloor$ yields an approximation factor of at most $k + \left(\sqrt{2\alpha}\right)\sqrt{k} + (\alpha+1) + o(1)$. We begin by substituting this value of ℓ into the approximation guarantee of Proposition 31. This gives us that the approximation factor is

$$\begin{aligned}
\frac{k+1+\frac{\alpha}{2}(\ell-1)}{1-1/\ell} &= \frac{k+1+\frac{\alpha}{2}(\lfloor 1+\sqrt{2(k+1)/\alpha}\rfloor - 1)}{1-\frac{1}{\lfloor 1+\sqrt{2(k+1)/\alpha}\rfloor}} \\
&\leq \frac{k+1+\frac{\alpha}{2}(1+\sqrt{2(k+1)/\alpha} - 1)}{1-\frac{1}{\sqrt{2(k+1)/\alpha}}} \\
&= \frac{\alpha}{2}\left(\frac{2(k+1)/\alpha + \sqrt{2(k+1)/\alpha}}{1-\frac{1}{\sqrt{2(k+1)/\alpha}}}\right) \ .
\end{aligned}$$

By defining $\gamma = 2(k+1)/\alpha$, we can simplify the term inside the parenthesis, and write it as

$$\frac{\gamma + \sqrt{\gamma}}{1 - 1/\sqrt{\gamma}} = (1+\sqrt{\gamma})^2 + 1 + \frac{4}{\gamma - 1} + \frac{2}{1+\sqrt{\gamma}} \ .$$

Substituting this back into the calculation above, we have that the approximation factor is at most

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

where the term $\eta=\mathcal{O}(\sqrt{\alpha^3/k})$ decreases gradually with k and is given explicitly as

$$\eta=\sqrt{2\alpha}\left(\sqrt{k+1}-\sqrt{k}\right)+\frac{\alpha}{\sqrt{\frac{2}{\alpha}(k+1)}-1}\ .$$

Finally, we demonstrate the our analysis obtains the improved approximation ratio of $k+1$ for monotone submodular functions when running the greedy algorithm. For monotone submodular functions, Proposition 31 yields an approximation factor of $k+1+\frac{\alpha}{2}(\ell-1)$. In this case, constructing a single solution ($\ell=1$) yields the approximation factor $k+1$ in the monotone case. ■

Although the term η goes to zero for large k, it may be non-negligible for very small k. The magnitude of this η term also depends on the USM approximation ratio, α. Roughly speaking, $\eta=\mathcal{O}(\sqrt{\alpha^3/k})$ so that a decrease in α can yield a significant decrease in this sub-constant term. For example, for $\alpha=3$ and $k=1$, we have that $\eta\approx 20.4$, and if k increases to 10 then $\eta\approx 2.1$. On the other hand, if $\alpha=2$ and $k=1$, then we have that $\eta\approx 5.7$, while when k increases to 10, we get $\eta\approx 1.2$.

6.1 Tight Approximation Analysis for k-Extendible Systems

Earlier in Section 6, we proved that RepeatedGreedy achieves a $k+\mathcal{O}(\sqrt{k})$ approximation under a k-system if allowed to run for $\mathcal{O}(\sqrt{k})$ iterations. A natural question is whether, like SimultaneousGreedys, the approximation factor of RepeatedGreedy may improve for the subclass of k-extendible systems. In this section, we answer this question in the negative by showing that RepeatedGreedy achieves an approximation factor of $k+\Omega(\sqrt{k})$ for the subclass of k-extendible systems. In particular, we prove the following theorem:

Theorem 32 *The approximation ratio of* RepeatedGreedy *for the problem of maximizing a non-negative submodular function subject to a k-extendible constraint is $k+\Omega(\sqrt{k})$ regardless of the number of iterations used by the algorithm.*

We remark that the proof of Theorem 32 works even when the constraint is restricted to be a k-matchoid, which is a special case of a k-extendible constraint.

To prove Theorem 32, we construct a family of instances on which RepeatedGreedy performs poorly. Specifically, we construct a bad instance, denoted by I_k, for every integer $k \geq 1$ such that $\sqrt{k}$ is also an integer. We begin the construction by defining for every integer $1 \leq i \leq \sqrt{k}$ the sets

$$O_i = \{o_{i,j} \mid 1 \leq j \leq 1 + \sqrt{k}\} \enspace , \quad B_i = \{b_i\} \enspace , \quad \text{and} \quad D_i = \{d_{i,j} \mid 1 \leq j \leq \sqrt{k}\} \enspace .$$

Then, the ground set of the instance I_k is the set $\mathcal{N}_k = \bigcup_{i=1}^{\sqrt{k}} (O_i \cup D_i \cup B_i)$. The objective function $f_k \colon 2^{\mathcal{N}_k} \to \mathbb{R}_+$ of I_k is defined for every set $S \subseteq \mathcal{N}_k$ by $f_k(S) = \sum_{i=1}^{\sqrt{k}} g_i(S \cap (O_i \cup D_i \cup B_i))$, where g_i is an auxiliary function $g_i \colon 2^{O_i \cup D_i \cup B_i} \to \mathbb{R}_+$ given for every set $S \subseteq O_i \cup D_i \cup B_i$ by

$$g_i(S) = \begin{cases} |\{j \mid S \cap \{o_{i,j}, d_{i,j}\} \neq \varnothing\}| + \frac{2|S \cap D_i| + |S \cap O_i|}{4\sqrt{k}} & \text{if } b_i \notin S \enspace , \\ \frac{3 + 4\sqrt{k} + |S \cap \{o_{i,1+\sqrt{k}}\}| - |S \cap (O_i \cup D_i - o_{i,1+\sqrt{k}})|}{4\sqrt{k}} & \text{if } b_i \in S \enspace . \end{cases}$$

Intuitively, the elements of O_i belong to the optimal solution, and the elements of D_i are decoy elements. To understand the roles of these elements, let us first ignore the elements b_i and $o_{i,1+\sqrt{k}}$. Once these elements are dropped, we get that every element $o_{i,j} \in O_i$ has a matching decoy element $d_{i,j} \in D_i$ such that $o_{i,j}$ and $d_{i,j}$ contribute roughly the same value to g_i (i.e., once one of them is added, the other does not contribute much more). The basic plan is to choose a constraint that makes every decoy element $d_{i,j}$ exclude all the optimal elements except for $o_{i,j}$, which will make RepeatedGreedy produce in each iteration a solution of the type $\{d_{i,j}, o_{i,j}\}$. However, since every $d_{i,j}$ elements can exclude only k elements, this basic plan can (on its own) only result in an approximation ratio of $k + 1$ (this is perhaps not surprising since the omission of b_i makes g_i monotone).

To improve over the above basic plan, we use also the elements b_i and $o_{i,1+\sqrt{k}}$. Notice that once b_i is selected, the marginal contributions of all the elements of O_i become insignificant (this is possible because g_i is non-monotone). To use this property, we choose below a constraint that allows b_i to exclude the elements of $O_{i'}$ for every $i' \neq i$. Thus, RepeatedGreedy is forced to combine b_i with $O_{1+\sqrt{k}}$ in a solution, and once this is done for every $1 \leq i \leq \sqrt{k}$, enough value of the optimal solution is lost to allow the above basic plan to prove Theorem 32.

We now formalize the above arguments, and we begin by proving some properties of the function g_i.

Observation 33 *For every integer $1 \leq i \leq \sqrt{k}$, the function g_i is non-negative and submodular.*

Proof The non-negativity of g_i follows immediately from its definition since the size of the set $O_i \cup D_i - o_{i,k+1}$ is $2\sqrt{k}$. Therefore, we focus on proving that g_i is submodular. Recall that the function g_i is defined on the ground set $O_i \cup D_i \cup B_i$. Thus, g_i is submodular if $g_i(u \mid S)$ is a down-monotone function of S for every element $u \in O_i \cup D_i \cup B_i$, where $S \subseteq O_i \cup D_i \cup B_i \setminus \{u\}$. We do that by considering a few cases. Consider first the case in which $u = b_i$. In this case

$$\begin{aligned}g_i(b_i \mid S) ={}& \frac{3 + 4\sqrt{k} + |S \cap \{o_{i,1+\sqrt{k}}\}| - |S \cap (O_i \cup D_i - o_{i,1+\sqrt{k}})|}{4\sqrt{k}}\\ &- |\{j \mid S \cap \{o_{i,j}, d_{i,j}\} \neq \varnothing\}| - \frac{2|S \cap D_i| + |S \cap O_i|}{4\sqrt{k}}\\ ={}& \frac{3 + 4\sqrt{k} - (4\sqrt{k} - 1) \cdot |S \cap \{o_{i,1+\sqrt{k}}\}| - |S \cap (O_i \cup D_i - o_{i,1+\sqrt{k}})|}{4\sqrt{k}}\\ &- |\{j \mid S \cap \{o_{i,j}, d_{i,j}\} \neq \varnothing, j \neq 1 + \sqrt{k}\}| - \frac{2|S \cap D_i| + |S \cap O_i|}{4\sqrt{k}} \quad ,\end{aligned}$$

which is clearly a down-monotone function of S (the second equality holds since $d_{i,k+1}$ does not belong to the ground set $\mathcal{N}_k$, and therefore, cannot appear in S). Consider now the case in which $u = o_{i,j}$ for some integer $1 \leq j \leq \sqrt{k}$. In this case

$$g_i(o_{i,j} \mid S) = \begin{cases} \frac{1}{4\sqrt{k}} + 1 - |S \cap \{d_{i,j}\}| & \text{if } b_i \notin S \enspace , \\ -\frac{1}{4\sqrt{k}} & \text{if } b_i \in S \enspace , \end{cases}$$

which is a down-monotone function of S since the expression for the case $b_i \notin S$ is always non-negative. The next case is when $u = o_{i,1+\sqrt{k}}$, which yields

$$g_i(o_{i,1+\sqrt{k}} \mid S) = \begin{cases} 1 + \frac{1}{4\sqrt{k}} & \text{if } b_i \notin S \enspace , \\ \frac{1}{4\sqrt{k}} & \text{if } b_i \in S \enspace , \end{cases}$$

which is down-monotone. The last case to consider is the case of $u = d_{i,j}$ for some integer $1 \leq j \leq \sqrt{k}$. In this case

$$g_i(d_{i,j} \mid S) = \begin{cases} \frac{1}{2\sqrt{k}} + 1 - |S \cap \{o_{i,j}\}| & \text{if } b_i \notin S \enspace , \\ -\frac{1}{4\sqrt{k}} & \text{if } b_i \in S \enspace , \end{cases}$$

which is down-monotone since the expression for the case $b_i \notin S$ is again always non-negative. ■

One can note that f_k is non-negative and submodular since it is the sum of $\sqrt{k}$ functions having these properties. To complete the description of the instance I_k, we still need to define its constraint. To do that, let us associate each element of I_k with up to k colors from the list $\{\perp\} \cup \{(i, j) \mid 1 \leq i \leq \sqrt{k}, 1 \leq j \leq 1 + \sqrt{k}\}$. A set is feasible under our constraint if no two elements in it share a color (this constraint is a k-extendible set system because it can be represented as a k-matchoid by having one matroid for each color whose role is to allow at most a single element with that color in a feasible set). The colors of the different elements are as follows.

- An element $o_{i,j} \in \mathcal{N}_k$ has (i, j) as its single color.
- An element $d_{i,j} \in \mathcal{N}_k$ has all the colors in $\{\perp\} \cup \{(i', j') \mid 1 \leq i' \leq \sqrt{k}, 1 \leq j' \leq \sqrt{k}\} \setminus \{(i, j)\}$, which is $1 + (k - 1) = k$ different colors.

- An element $b_i \in \mathcal{N}_k$ has all the colors in $\{\perp\} \cup \{(i', j') \mid 1 \leq i' \leq \sqrt{k}, i' \neq i, 1 \leq j' \leq 1 + \sqrt{k}\}$, which is $1 + (\sqrt{k} - 1)(1 + \sqrt{k}) = k$ different colors.

The following observation shows that the optimal solution for the instance I_k has a lot of value.

Observation 34 *The value of the optimal solution for I_k is at least $k + \frac{5\sqrt{k}}{4}$.*

Proof Note that the set $\bigcup_{i=1}^{\sqrt{k}} O_i$ is a feasible solution. The value of this set according to f_k is

$$f_k\left(\bigcup_{i=1}^{\sqrt{k}} O_i\right) = \sum_{i=1}^{\sqrt{k}} g_i(O_i) = \sum_{i=1}^{\sqrt{k}} \left(1 + \frac{1}{4\sqrt{k}}\right) \cdot |O_i| = \sqrt{k} \cdot \left(1 + \frac{1}{4\sqrt{k}}\right) \cdot (1 + \sqrt{k}) \geq k + \frac{5\sqrt{k}}{4} \quad .$$

■

Our next objective is to analyze the performance of RepeatedGreedy given the input I_k. We do that using the following two lemmata.

Lemma 35 *Let L be a strict subset of $\{1, 2, \ldots, \sqrt{k}\}$, and assume that the greedy algorithm is applied to the instance I_k restricted to the ground set $\mathcal{N}_k \setminus \{b_i, o_{i,1+\sqrt{k}} \mid i \in L\}$. Then, it outputs the set $\{b_i, o_{i,1+\sqrt{k}}\}$ for some $i \notin L$.*

Proof We begin the proof by considering the marginal contribution of every element of $\mathcal{N}_k$ with respect to $\varnothing$.

- For every two integers $1 \leq i \leq \sqrt{k}$ and $1 \leq j \leq 1 + \sqrt{k}$, $f_k(o_{i,j} \mid \varnothing) = 1 + \frac{1}{4\sqrt{k}}$.
- For every two integers $1 \leq i \leq \sqrt{k}$ and $1 \leq j \leq \sqrt{k}$, $f_k(d_{i,j} \mid \varnothing) = 1 + \frac{2}{4\sqrt{k}}$.
- For every integer $1 \leq i \leq \sqrt{k}$, $f_k(b_i \mid \varnothing) = \frac{3+4\sqrt{k}}{4\sqrt{k}} = 1 + \frac{3}{4\sqrt{k}}$.

One can observe that the marginal contribution calculated above for the b_i elements is larger than the marginal contributions calculated for the other elements (and is positive), and therefore, the first element that the greedy algorithm will add to its solution will be one such element that is available in the ground set.

Assume therefore that the greedy algorithm has picked so far into its solution only the element b_i for some $i \notin L$. Due to the constraint, the only elements that can still be added to the solution once b_i is in it are the elements of O_i. Their marginal contribution with respect to $\{b_i\}$ is

- For every integer $1 \leq j \leq \sqrt{k}$, $f_k(o_{i,j} \mid \{b_i\}) = -\frac{1}{4\sqrt{k}}$.
- $f_k(o_{i,1+\sqrt{k}} \mid \{b_i\}) = \frac{1}{4\sqrt{k}}$.

Since the marginal contribution calculated for $o_{i,1+\sqrt{k}}$ is the largest (and is positive), the greedy algorithm picks $o_{i,1+\sqrt{k}}$ as the next element to add to its solution. Furthermore, since the marginal contributions of the remaining elements of O_i are already negative at this stage (and hence, will be negative in the future as well), the greedy algorithm does not pick any of them. Thus, its output set is $\{b_i, o_{i,1+\sqrt{k}}\}$, as promised. ■

Lemma 36 *Let L be a strict subset of $\{(i,j) \mid 1 \leq i,j \leq \sqrt{k}\}$, and assume that the greedy algorithm is applied to the instance I_k restricted to the ground set $\mathcal{N}_k \setminus (\{o_{i,j}, d_{i,j} \mid (i,j) \in L\} \cup \{b_i, o_{i,1+\sqrt{k}} \mid 1 \leq i \leq \sqrt{k}\}$. Then, it outputs the set $\{o_{i,j}, d_{i,j}\}$ for some $(i,j) \notin L$.*

Proof We begin the proof by considering the marginal contribution of every element of $\mathcal{N}_k \setminus \{b_i, o_{i,1+\sqrt{k}} \mid 1 \leq i \leq \sqrt{k}\}$ with respect to $\varnothing$.

- For every two integers $1 \leq i \leq \sqrt{k}$ and $1 \leq j \leq \sqrt{k}$, $f_k(o_{i,j} \mid \varnothing) = 1 + \frac{1}{4\sqrt{k}}$.
- For every two integers $1 \leq i \leq \sqrt{k}$ and $1 \leq j \leq \sqrt{k}$, $f_k(d_{i,j} \mid \varnothing) = 1 + \frac{2}{4\sqrt{k}}$.

One can observe that the marginal contribution calculated above for the $d_{i,j}$ elements is larger than the marginal contribution calculated for the other elements (and is positive), and therefore, the first element that the greedy algorithm adds to its solution is one such element that is available in the ground set.

Assume therefore that the greedy algorithm has picked so far into its solution only the element $d_{i,j}$ for some $(i,j) \notin L$. Due to the constraint, the only element that can still be added to the solution once $d_{i,j}$ is in it is $o_{i,j}$, whose marginal with respect to $\{d_{i,j}\}$ is $f_k(o_{i,j} \mid \{d_{i,j}\}) = 1/\sqrt{4k}$, which is positive. Hence, the greedy algorithm selects at this point $o_{i,j}$, and outputs the set $\{o_{i,j}, d_{i,j}\}$, as promised. ■

Combining the two last lemmata, we get the following corollary.

Corollary 37 *Regardless of number of iterations of* RepeatedGreedy *used, the only sets it can output are either subsets of $\{b_i, o_{i,1+\sqrt{k}}\}$ for some integer $1 \leq i \leq \sqrt{k}$ or subsets of $\{o_{i,j}, d_{i,j}\}$ for some integers $1 \leq i,j \leq \sqrt{k}$.*

We can now upper bound the value of the output of RepeatedGreedy.

Lemma 38 *The value of the output set of* RepeatedGreedy *given I_k is at most $1 + \frac{1}{\sqrt{k}}$.*

Proof To prove the lemma, we need to show that every set that RepeatedGreedy might output according to Corollary 37 has a value of at most $1 + \frac{1}{\sqrt{k}}$. We do that by considering every possible type of such sets.

- $f_k(\varnothing) = 0$.
- For every integer $1 \leq i \leq \sqrt{k}$, $f_k(\{b_i\}) = 1 + \frac{3}{4\sqrt{k}} < 1 + \frac{1}{\sqrt{k}}$.
- For every integer $1 \leq i \leq \sqrt{k}$, $f_k(\{b_i, o_{i,1+\sqrt{k}}\}) = 1 + \frac{4}{4\sqrt{k}} = 1 + \frac{1}{\sqrt{k}}$.
- For every two integers $1 \leq i \leq \sqrt{k}$ and $1 \leq j \leq 1+\sqrt{k}$, $f_k(\{o_{i,j}\}) = 1 + \frac{1}{4\sqrt{k}} < 1 + \frac{1}{\sqrt{k}}$.

- For every two integers $1 \leq i, j \leq \sqrt{k}$, $f_k(\{d_{i,j}\}) = 1 + \frac{2}{4\sqrt{k}} < 1 + \frac{1}{\sqrt{k}}$.
- For every two integers $1 \leq i, j \leq \sqrt{k}$, $f_k(\{o_{i,j}, d_{i,j}\}) = 1 + \frac{3}{4\sqrt{k}} < 1 + \frac{1}{\sqrt{k}}$. ■

To complete the proof of Theorem 32, it remains to observe that the ratio between the value of the optimal solution of I_k (lower bounded by Observation 34) and the maximum value of a set that RepeatedGreedy can output (upper bounded by Lemma 38) is at least

$$\frac{k + \frac{5\sqrt{k}}{4}}{1 + \frac{1}{\sqrt{k}}} = \frac{4k\sqrt{k} + 5k}{4\sqrt{k} + 4} = k + \frac{k}{4\sqrt{k} + 4} = k + \Omega(\sqrt{k}) \ .$$

6.2 Nearly Linear Time with Knapsack Constraints

In this section, we demonstrate how RepeatedGreedy may be modified to run in nearly linear time and achieve approximations for the more general problem (2), where there are m additional knapsack constraints. As before, we use the marginal gain thresholding technique of Badanidiyuru and Vondrák (2014) to ensure a nearly linear run time at the cost of an (arbitrarily) small multiplicative increase in the approximation factor. To handle knapsack constraints, we use the density threshold technique of Mirzasoleiman et al. (2016) with our improved binary search analysis. These modification techniques are identical to those used in Sections 4 and 5, and so our discussion of them in this section is considerably shorter.

These modifications are made primarily in the greedy subroutine, which we present below as ModifiedGreedy. ModifiedGreedy is similar to Greedy, but differs in two respects: rather than iteratively searching over all elements to find the one with largest marginal gain, the ModifiedGreedy iteratively decreases marginal gain thresholds and accepts any element whose marginal gain is above the threshold *and* whose density is above the density threshold. The formal details of the implementation are given below as Algorithm 8.

The ModifiedRepeatedGreedy algorithm iteratively calls ModifiedGreedy to produce a solution S_i, runs an unconstrained submodular maximization (USM) subroutine on S_i to obtain S'_i, and then removes S_i from the ground set. Finally, ModifiedRepeatedGreedy returns the best solution among the sequence $S_1, S'_1, \dots S_\ell, S'_\ell$ produced. Note that this is similar to RepeatedGreedy, the only difference being that ModifiedRepeatedGreedy calls ModifiedGreedy rather than the vanilla greedy algorithm. We present ModifiedRepeatedGreedy formally below as Algorithm 7.

Algorithm 9: ModifiedRepeatedGreedy $(\mathcal{N}, f, \mathcal{I}, \ell, \rho, \varepsilon)$

```
1 Let $\mathcal{N}_1 \leftarrow \mathcal{N}$.
2 for $i = 1$ to $\ell$ do
3     Run modified greedy procedure $S_i \leftarrow$ ModifiedGreedy$(\mathcal{N}_i, f, \mathcal{I}, \rho, \varepsilon)$
4     Filter the greedy solution $S'_i \leftarrow$ USM$(S_i)$
5     Update ground set $\mathcal{N}_{i+1} \leftarrow \mathcal{N}_i \setminus S_i$.
6 return the set $S$ maximizing $f$ among the sets $\{S_i, S'_i\}_{i=1}^{\ell}$.
```

Algorithm 8: ModifiedGreedy $(\mathcal{N}, f, \mathcal{I}, \rho, \varepsilon)$

```
1  Initialize solution $S_0 \leftarrow \varnothing$ and iteration counter $i \leftarrow 1$.
2  Let $\Delta_f = \max_{u \in \mathcal{N}} f(u)$, and initialize threshold $\tau = \Delta_f$.
3  while $\tau > (\varepsilon/n) \cdot \Delta_f$ do
4      for every element $u$ with $u \in \mathcal{N}$ such that $S_{i-1} + u \in \mathcal{I}$ do
5          if $f(u \mid S_{i-1}) \geq \max(\tau, \rho \cdot \sum_{r=1}^{m} c_r(u))$ then
6              if $c_r(S_{i-1} + u) \leq 1$ for all $1 \leq r \leq m$ then
7                  Let $u_i \leftarrow u$ .
8                  Update the solution as $S_i \leftarrow S_{i-1} + u$.
9                  Update the iteration counter $i \leftarrow i + 1$.
10     Update marginal gain $\tau \leftarrow (1 - \varepsilon) \cdot \tau$.
11 Let $u^* \leftarrow \arg\max_{u \in \mathcal{N}} f(u)$.
12 return the set $S$ maximizing $f$ among the sets $S_i$ and $\{u^*\}$.
```

In the following observation, we bound the running time of ModifiedRepeatedGreedy and prove feasibility of the returned solution.

Observation 39 ModifiedRepeatedGreedy *requires $\mathcal{O}(\ell n/\varepsilon)$ oracle calls, $\mathcal{O}(\ell nm/\varepsilon)$ arithmetic operations, and its output S is independent in $\mathcal{I}$ and satisfies the knapsack constraints.*

Proof These statements follow largely from analysis of ModifiedGreedy. Note that each iteration of the while loop of ModifiedGreedy examines each element only once so that $\mathcal{O}(n)$ oracle calls and $\mathcal{O}(mn)$ arithmetic operations are required during each iteration. As discussed in the proof of Observation 19, there are at most $\tilde{\mathcal{O}}(1/\varepsilon)$ iterations of the while loop when the input error term satisfying $\varepsilon < 1/2$. Thus, ModifiedGreedy requires a total of $\tilde{\mathcal{O}}(n/\varepsilon)$ oracle queries and $\tilde{\mathcal{O}}(nm/\varepsilon)$ arithmetic operations. Moreover, by the acceptance criteria, the solution returned by ModifiedGreedy is independent in $\mathcal{I}$ and satisfies the knapsack constraints.

ModifiedRepeatedGreedy makes ℓ calls to ModifiedGreedy and to the USM subroutine, which is assumed to run in linear time. Thus, the algorithm requires $\tilde{\mathcal{O}}(\ell n/\varepsilon)$ oracle calls and $\tilde{\mathcal{O}}(\ell nm/\varepsilon)$ arithmetic operations. Finally, the solution returned by ModifiedRepeatedGreedy is feasible with respect to independence and knapsack constraints because all the outputs of ModifiedGreedy and USM have this property. ■

The following proposition provides an approximation guarantee for the solution returned by ModifiedRepeatedGreedy. As in Section 5, we define E to be an indicator variable which takes the value 1 if the knapsack check in Line 6 of ModifiedGreedy evaluates to false at any point in the execution of ModifiedRepeatedGreedy and 0 otherwise.

Proposition 40 *If $(\mathcal{N}, \mathcal{I})$ is a k-system, then the solution S returned by* ModifiedRepeatedGreedy *satisfies the following approximation guarantees.*

$$f(S) \geq \begin{cases} \frac{1}{2}\rho & \text{if } E = 1 \ , \\ \left(\frac{1-\varepsilon}{k+1+\alpha(\ell-1)/2}\right)\Big((1 - 1/\ell - \varepsilon) f(OPT) - \rho m\Big) & \text{if } E = 0 \ . \end{cases} \tag{13}$$

Moreover, when f is monotone, these approximation guarantees improve to

$$f(S) \geq \begin{cases} \frac{1}{2}\rho & \text{if } E = 1 \ , \\ \left(\frac{(1-\varepsilon)}{k+1+\alpha(\ell-1)/2}\right)\Big((1 - \varepsilon) f(OPT) - \rho m\Big) & \text{if } E = 0 \ . \end{cases} \tag{14}$$

The proof of Proposition 40 is similar to that of Proposition 31, except that we cannot use the analysis of Gupta et al. (2010) for the approximation ratio of Greedy subject to a k-system. Instead, we must use an analysis which takes into account the marginal gain and density thresholding techniques. Because the main proof ideas involving the repeated greedy technique are presented in Section 6 and the marginal gain threshold and density ratio threshold techniques already appear in existing works, we defer the proof of Proposition 40 to Appendix B.

We would like to choose a density parameter ρ to maximize the lower bound on the returned objective value in Proposition 40. As in Section 5, we propose a binary search approach on the density parameter ρ using the knapsack rejection indicator E. We sketch this idea again here for completeness.

Suppose that the f is a general non-monotone submodular function—the monotone case may be handled similarly. Choosing the density parameter $\rho^* = 2(1-\varepsilon)\left(\frac{1-1/\ell-\varepsilon}{k+2m+1+\alpha(\ell-1)/2}\right) \triangleq \beta \cdot f(OPT)$ approximately maximizes the lower bound (13) presented in Proposition 40, which yields an approximation guarantee of

$$f(S) \geq (1-\varepsilon)\left(\frac{1 - 1/\ell - \varepsilon}{k + 2m + 1 + \alpha(\ell-1)/2}\right) f(OPT) \ .$$

Although the term β is known, we do not know the optimal objective value $f(OPT)$ and so there is no way for us to know the value of the (approximately) optimal density parameter ρ^*. However, the submodularity of f implies that the optimal objective value lies within the interval $\Delta_f \leq f(OPT) \leq r \cdot \Delta_f$, and thus, the optimal density parameter ρ^* lies in the interval $\beta \cdot \Delta_f \leq \rho^* \leq \beta \cdot (r \cdot \Delta_f)$. Mirzasoleiman et al. (2016) proposed running a multiplicative grid search over this interval, where the repeated greedy algorithm is run with each ρ in this grid. This grid search technique yields an approximation factor which is only $(1+\delta)$ times larger than if we had used the optimal ρ^* and requires $\mathcal{O}(1/\delta)$ calls to the repeated greedy algorithm. As in Section 5, we improve upon this technique by proposing a binary search method which uses the knapsack rejection indicator E. The same approximation factor is achieved, but only $\tilde{\mathcal{O}}(1)$ calls to the repeated greedy algorithm are required, which is an exponential decrease compared to the "brute-force" grid search.

We refer to this binary search routine for calling ModifiedRepeatedGreedy with different values of the density parameter as DensitySearchRG. This is formally outlined below as Algorithm 10.

Algorithm 10: DensitySearchRG $(\mathcal{N}, f, \mathcal{I}, \ell, \delta, \varepsilon, \beta)$

```
1  Initialize upper and lower bounds $k_\ell = 1$, $k_u = \lceil \frac{1}{\delta} \log n \rceil$.
2  Let $\Delta_f = \max_{u \in \mathcal{N}} f(u)$, and initialize iteration counter $i \leftarrow 1$.
3  while $|k_u - k_\ell| > 1$ do
4      Set middle bound $k_i = \left\lceil \frac{k_\ell + k_u}{2} \right\rceil$.
5      Set density ratio $\rho_i \leftarrow \beta \cdot \Delta_f (1+\delta)^{k_i}$.
6      Obtain set $S_i \leftarrow$ ModifiedRepeatedGreedy$(\mathcal{N}, f, \mathcal{I}, \ell, \rho_i, \varepsilon)$.
7      if $E_i = 0$ then
8          Increase lower bound $k_\ell \leftarrow k_i$.
9      else
10         Decrease upper bound $k_u \leftarrow k_i$.
11     Update iteration counter $i \leftarrow i + 1$.
12 Set density ratio $\rho_i \leftarrow \beta \cdot \Delta_f (1+\delta)^{k_\ell}$.
13 Obtain set $S_i \leftarrow$ ModifiedRepeatedGreedy$(\mathcal{N}, f, \mathcal{I}, \ell, \rho_i, \varepsilon)$.
14 return the set $S$ maximizing $f$ among the sets $\{S_1, \dots S_T\}$.
```

The approximation guarantees of DensitySearchRG are given below in Proposition 41. The proof is omitted because it is nearly identical to that of Proposition 24. In particular, the proof of Proposition 41 uses Proposition 40 in the same way that Proposition 23 is used in the proof of Proposition 24.

Proposition 41 DensitySearchRG *makes* $\tilde{\mathcal{O}}(1)$ *calls to* ModifiedRepeatedGreedy. *If* $(\mathcal{N}, \mathcal{I})$ *is a* k*-system and* $\beta = 2(1-\varepsilon)\left(\frac{1-1/\ell-\varepsilon}{k+2m+1+\alpha(\ell-1)/2}\right)$, *then the solution* S *returned by* DensitySearchRG *satisfies*

$$f(S) \geq (1-\delta)(1-\varepsilon)\left(\frac{1-1/\ell-\varepsilon}{k+2m+1+\alpha(\ell-1)/2}\right) f(OPT)$$
$$\geq (1-\delta)(1-2\varepsilon)^2\left(\frac{1-1/\ell}{k+2m+1+\alpha(\ell-1)/2}\right) f(OPT)$$

when the number of iterations ℓ *is at least* 2. *Moreover, if* f *is monotone and* $\beta = \frac{2(1-\varepsilon)^2}{k+2m+1+\alpha(\ell-1)/2}$ *then this lower bound improves to*

$$f(S) \geq (1-\delta)(1-\varepsilon)^2\left(\frac{1}{k+2m+1+\alpha(\ell-1)/2}\right) f(OPT)$$

for any number of iterations ℓ.

By setting the number of solutions ℓ to maximize the lower bounds in Proposition 41, we may obtain the following result.

Theorem 42 *Suppose that* $(\mathcal{N}, \mathcal{I})$ *is a* k*-system, the number of iterations is set to* $\ell = \lfloor 1 + \sqrt{2(k+2m+1)/\alpha} \rfloor$, *and the two error terms are set to be equal (i.e.,* $\varepsilon = \delta \in (0, 1/2)$*). Then,* DensitySearchRG *requires* $\tilde{\mathcal{O}}(\sqrt{k+m} \cdot n/\varepsilon)$ *oracle calls and* $\tilde{\mathcal{O}}(\sqrt{k+m} \cdot mn/\varepsilon)$ *arithmetic operations and produces a solution whose approximation ratio is at most*

$$(1-2\varepsilon)^{-3}\Big[k + 2m + (\sqrt{2\alpha})\sqrt{k+2m} + (\alpha+1) + o(1)\Big] \ .$$

Moreover, when f is non-negative monotone submodular and the number of iterations is set to $\ell = 1$, then the approximation ratio improves to $(1-\varepsilon)^{-3}(k + 2m + 1)$.

As discussed in Section 5, the $(1-2\varepsilon)^{-1}$ multiplicative factor can always be made into a $(1+\varepsilon')$ approximation factor by setting $\varepsilon' = c\cdot\varepsilon$ for some constant $c < 1$. We omit the proof of Theorem 42, as the analysis is essentially the same as in the proof of Theorem 28, except that k is replaced now by $k + 2m$. Indeed, one of the interpretations of Theorem 42 is that the ModifiedRepeatedGreedy algorithm achieves approximation guarantees similar to RepeatedGreedy, except that the independence parameter k is replaced with $k + 2m$ to account for the additional knapsack constraints, while running in nearly linear time.

7. Hardness Results

In this section, we present hardness results which complement our algorithmic contributions. In particular, we study the hardness of maximizing linear functions and monotone submodular functions over k-extendible systems. These hardness results demonstrate that the approximation guarantees of SimultaneousGreedys for submodular maximization over k-extendible systems are nearly optimal (up to low order terms) amongst all polynomial time algorithms. We emphasize here that the following hardness results are information theoretic, and thus, independent of computational complexity hypotheses such as $P \neq NP$. The first hardness result regards the approximability of maximizing a linear function over a k-extendible system.

Theorem 43 *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 $\varepsilon > 0$.*

The second hardness result regards the approximability of maximizing a monotone submodular function over a k-extendible system.

Theorem 44 *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 $\varepsilon > 0$.*

Recall that SimultaneousGreedys achieves an approximation ratio of $(k+1)^2/k = k + 2 + 1/k$ for maximizing a submodular function over a k-extendible system and that this approximation ratio improves to $k+1$ when the objective is monotone. The hardness result of Theorem 44 shows that achieving an approximation ratio better than $(1-e^{-1/k})^{-1} - \varepsilon \geq k + 1/2 - \varepsilon$ for monotone objectives requires exponentially many queries to the value and independence oracles. Hence, the gap between the achieves approximation ratio and the hardness result is a small constant. In this sense, the approximation achieved by SimultaneousGreedys and its variants for maximizing over a k-extendible system is near-optimal amongst all algorithms which query the oracles polynomially many times.

SimultaneousGreedys has an approximation guarantee of $k + \mathcal{O}(\sqrt{k})$ for the more general class of k-systems, and so it is natural to wonder whether this approximation is also near-optimal amongst polynomial time algorithms. Since k-extendible systems are a subclass of k-systems, the hardness results presented here also apply to k-systems; however, the gap between the $k + \mathcal{O}(\sqrt{k})$ approximation and the $k + 1/2 - \varepsilon$ hardness is larger in this case. Indeed, it is an open question whether the additive $\mathcal{O}(\sqrt{k})$ term is necessary for any polynomial time algorithm which maximizes a non-monotone submodular objective over a k-system or whether this factor may be improved.

The proof of Theorems 43 and 44 consists of two steps which are organized into two respective sections. In Section 7.1, we define two k-extendible systems which are indistinguishable in polynomial time. The inapproximability result for linear objectives follows from the indistinguishability of these systems and the fact that the sizes of their maximal sets are very different. In Section 7.2, we 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.

7.1 Hardness for Linear Functions Over k-Extendible Systems

In this section, we construct two k-extendible systems which, after a random permutation is applied to the ground set, are indistinguishable using polynomially many queries with high probability. Moreover, the size of the largest base is significantly different between these two systems.

First, we construct a k-extendible system $\mathcal{M}(k,h,m) = (\mathcal{N}_{k,h,m}, \mathcal{I}_{k,h,m})$, which is parameterized by three positive integers k, h and m such that h is an integer multiple of $2k$. The ground set of the system consists of h groups of elements, each of size km. More formally, the ground set 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 if and only if it obeys the following inequality:

$$g(|S \cap H_1(k,m)|) + |S \setminus H_1(k,m)| \leq m \enspace ,$$

where the function g is a piece-wise linear function defined by

$$g(x) = \min\left\{x, \frac{2km}{h}\right\} + \max\left\{\frac{x - 2km/h}{k}, 0\right\} = \begin{cases} x & \text{if } x \leq 2km/h \\ \frac{x}{k} + (1 - \frac{1}{k})\frac{2km}{h} & \text{if } x \geq 2km/h \end{cases} .$$

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.

For notational clarity, we drop the reference to the underlying parameters k, h, and m in the definition of the set systems throughout the rest of the section. That is, we write $\mathcal{M}$ and H_i instead of the more burdensome $\mathcal{M}(k,h,m)$ and $H_i(k,m)$.

Lemma 45 *For every choice of h and m, $\mathcal{M}$ is a k-extendible system.*

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

Consider an arbitrary independent set $C \in \mathcal{I}$, an independent extension D of C and an element $u \notin D$ for which $C + u \in \mathcal{I}$. We need to find a subset $Y \subseteq D \setminus C$ of size at most k such that $D \setminus Y + u \in \mathcal{I}$. 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

$$\Sigma(S) = g(|S \cap H_1) + |S \setminus H_1|.$$

By definition, $\Sigma(D) \leq m$ because $D \in \mathcal{I}$. 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

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

which implies that $D \setminus Y + u \in \mathcal{I}$. ■

Let us now show that $\mathcal{M}$ contains a large independent set.

Observation 46 *$\mathcal{M}$ 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.*

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 $H_1 \subseteq \mathcal{N}$ since $s \leq km$. Also,

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

Since S contains only elements of H_1, its independence follows from the above inequality. ■

Let us now define our second k-extendible system $\mathcal{M}' = (\mathcal{N}, \mathcal{I}')$. The ground set of this system is the same as the ground set of $\mathcal{M}$, but a set $S \subseteq \mathcal{N}$ 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 $\mathcal{M}$ and $\mathcal{M}'$ is at least

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

Our plan is to show that it takes exponential time to distinguish between the systems $\mathcal{M}$ and $\mathcal{M}'$, 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 $\mathcal{M}$ or $\mathcal{M}'$ after a random permutation was applied to the ground set. We prove below that with high probability the algorithm fails to distinguish between the two possible inputs. Notice that by Yao's lemma (Yao, 1977; Borodin and El-Yaniv, 1998), 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 $\mathcal{M}'$, 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 $\mathcal{M}'$ depends only on its size, and thus, the algorithm will take the same execution path given every permutation. If the same algorithm now gets $\mathcal{M}$ 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 $\mathcal{M}'$) 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 $\mathcal{M}'$. 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 $\mathcal{M}$ and $\mathcal{M}'$.

Lemma 47 *Following the application of the random ground set permutation, the probability that a set S is independent in $\mathcal{M}$ but not in $\mathcal{M}'$, 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 $\mathcal{M}$ and $\mathcal{M}'$ of a random set of size $|S|$. If $|S| > km$, then the set is never independent in either $\mathcal{M}$ or $\mathcal{M}'$, and if $|S| \leq m$, then the set is always independent in both $\mathcal{M}$ and $\mathcal{M}'$. Thus, the interesting case is when $m < |S| \leq km$.

Let $X = |S \cap H_1|$. 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

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

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

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 $\mathcal{N}$ is $n = mkh = O(m)$. 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 distinguish between the cases in which it gets as input $\mathcal{M}$ or $\mathcal{M}'$.

Using the above results, we are now ready to prove Theorem 43.

Proof *(Proof of Theorem 43)*: Consider an algorithm that needs to maximize the cardinality function over the k-extendible system $\mathcal{M}$ after the random permutation was applied, and let T be its output set. Notice that T must be independent in $\mathcal{M}$, and thus, its size is always upper bounded by mk. Moreover, since the algorithm fails, with high probability, to distinguish between $\mathcal{M}$ and $\mathcal{M}'$, T is with high probability also independent in $\mathcal{M}'$, and thus, has a size of at most m. Therefore, the expected size of T cannot be larger than $m + o(1)$ (formally, this $o(1)$ terms represents an expression that goes to 0 as m increases for any given choice of k and h).

On the other hand, Lemma 46 shows that $\mathcal{M}$ contains an independent set of size at least $mk(1 - 2k/h)$. Thus, the approximation ratio of the algorithm is no better than

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

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

7.2 Hardness for Submodular Functions Over k-Extendible Systems

In this section, we prove Theorem 44, which is a stronger inapproximability result for maximizing monotone submodular functions over k-extendible systems. As in the previous section, we construct two problem instances which have different optimal values but—after a permutation of the ground set—are indistinguishable using only polynomially many oracle queries. We use again the two k-extendible systems $\mathcal{M}$ and $\mathcal{M}'$, parametrized by h, m and k, defined in the last section. The additional technical construction of this section is two submodular functions f and g which take very different optimal values over the two extendible systems. To construct these two submodular functions, we use the symmetry gap technique developed by Vondrák (2013).

The symmetry gap technique is a general method for constructing hard instances for submodular optimization; however, we present a relatively self-contained version of this technique, appealing only to the main technical construction from Vondrák (2013). The rest of this paragraph is a high level roadmap for the construction of the submodular functions f and g based on the symmetry gap technique. Recall that the common ground set $\mathcal{N}$ of our independence systems is the union of h disjoint sets of elements, i.e., $\mathcal{N} = \cup_{i=1}^{h} H_i$. We begin the construction of f and g by defining an initial function $q\colon 2^{[h]} \to \mathbb{R}_+$, which assigns a non-negative value to each subset X of $[h]$. A key aspect of this initial function is that it has a desired symmetry property. In our context, that means that the value of q depends only on the cardinality of its input, i.e., $|X|$. Next, we consider the multilinear extension of the initial function, which is denoted by $Q\colon [0,1]^h \to \mathbb{R}_+$ and is defined as

$$Q(x) = \sum_{X \subseteq [h]} q(X) \prod_{i \in X} x_i \prod_{i \notin X} (1 - x_i) \enspace .$$

Note that Q is a function on vectors of the hypercube $[0,1]^h$. The next step is to apply a well-chosen perturbation to this multilinear extension Q to obtain a function $F\colon [0,1]^h \to \mathbb{R}_+$

$$\begin{tikzcd}
\boxed{\begin{array}{c} q\colon 2^{[h]} \to \mathbb{R}_+ \\ \text{initial function} \end{array}} \arrow[r] & \boxed{\begin{array}{c} Q\colon [0,1]^h \to \mathbb{R}_+ \\ \text{relaxation} \end{array}} \arrow[r] & \boxed{\begin{array}{c} F\colon [0,1]^h \to \mathbb{R}_+ \\ \text{perturbation} \end{array}} \arrow[r] \arrow[d] & \boxed{\begin{array}{c} f\colon 2^{\mathcal{N}} \to \mathbb{R}_+ \\ \text{hardness construction} \end{array}} \\
 & & \boxed{\begin{array}{c} G\colon [0,1]^h \to \mathbb{R}_+ \\ \text{symmetrization} \end{array}} \arrow[r] & \boxed{\begin{array}{c} g\colon 2^{\mathcal{N}} \to \mathbb{R}_+ \\ \text{hardness construction} \end{array}}
\end{tikzcd}$$

Figure 1: Construction of the objective functions f and g.

and then symmetrize F to obtain a second function $G\colon [0,1]^h \to \mathbb{R}_+$. At this point, we construct the desired set functions f and g on the original ground set $\mathcal{N}$ by mapping sets $S \in \mathcal{N}$ to vectors $x \in [0,1]^h$. This mapping depends on the number of elements of S that are in each of the partitions $H_1 \dots H_h$ of the ground set. Specifically, we define the mapping as

$$x(S) = \left(\frac{|S \cap H_1|}{|H_1|}, \frac{|S \cap H_2|}{|H_2|}, \dots \frac{|S \cap H_h|}{|H_h|} \right) \quad ,$$

and the set functions f and g are defined as $f(S) = F(x(S))$ and $g(S) = G(x(S))$, respectively. A diagram that summarizes this construction appears in Figure 1. The key technical lemma of Vondrák (2013) shows that, by picking an appropriate perturbation and symmetrization method, we can guarantee that the set functions f and g are both monotone submodular, g depends only on the cardinality of its input, and yet the two functions have similar values on most inputs.

Now with the roadmap complete, we begin our construction of the functions f and g. We define the initial function $q\colon 2^{[h]} \to \mathbb{R}_+$ as

$$q(X) = \min\{|X|, 1\} \quad .$$

Let $Q : [0,1]^h \to \mathbb{R}_+$ be the mutlilinear extension of q. One can observe that $Q(x) = 1 - \prod_{i \in [h]} (1 - x_i)$ for every vector $x \in [0,1]^h$. Furthermore, for every such vector x, we define its *symmetrization* as $\bar{x} = (\|x\|_1 / h) \cdot \mathbf{1}_{[h]}$ ($\mathbf{1}_{[h]}$ represents here the all ones vector in $[0,1]^h$). The next lemma is a direct application of Lemma 3.2 in (Vondrák, 2013), but simplified for our setting. It follows from the symmetry in the initial function q, namely that it is invariant under any permutation of the elements of $[h]$.

Lemma 48 *For every $\varepsilon' > 0$ there exists $\delta_h > 0$ and two functions $F, G : [0,1]^h \to \mathbb{R}_+$ with the following properties.*

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

The first property formally states the sense in which F is a perturbation of Q; namely, that their values differ only by an ε' amount for all vectors in the unit cube. The second property formally states the sense in which the function G is a symmetrization of F; namely, that evaluation of G at x is obtained by evaluating F at the symmetrization $\bar{x}$. The third property states that the two functions are equal on input vectors which are nearly symmetrized. The final property is used below to show the monotonicity and submodularity of the set functions f and g (which are more formally constructed below).

Recall that the mapping from sets $S \subseteq \mathcal{N}$ to vectors $x \in [0,1]^h$ is defined using the partition of $\mathcal{N}$ into the sets $H_1, H_2, \dots H_h$ as

$$x(S) = \left(\frac{|S \cap H_1|}{|H_1|}, \frac{|S \cap H_2|}{|H_2|}, \dots \frac{|S \cap H_h|}{|H_h|} \right) \enspace .$$

The set functions are defined as $f(S) = F(x(S))$ and $g(S) = G(x(S))$. The next lemma uses the final property of Lemma 48 to argue that f and g are both monotone and submodular. Its proof is essentially identical to the proof of Lemma 3.1 of (Vondrák, 2013). Nevertheless, we include it here for completeness.

Lemma 49 *The set functions f and g defined as above are monotone and submodular.*

Proof We only show that f is monotone submodular, as the proof that g is monotone submodular is identical. We begin by showing the monotonicity of f. To this end, the main technical condition on F that we need is:

$$F(w) \leq F(y) \quad \text{for all } w, y \in [0,1]^h \text{ satisfying } w \preceq y \enspace , \tag{15}$$

where $\preceq$ denotes the component-wise partial ordering. To see that this condition holds, consider the line segment between w and y, given by $v(t) = (1-t) \cdot w + t \cdot y$, where $t \in [0,1]$ is a parametrization of the line segment. Since $w \preceq y$, we have that the coordinates of $y - w$ are all non-negative. Additionally, by Lemma 48, we have that F is differentiable everywhere and $\frac{\partial F}{\partial x_u} \geq 0$ everywhere for every $u \in [h]$. This means that each coordinate of the gradient $\nabla F(v(t))$ is non-negative for each $t \in [0,1]$. Thus, the following inner product is non-negative:

$$\langle \nabla F(v(t)), y - w \rangle \geq 0 \quad \text{for all } t \in [0,1] \enspace .$$

Using this and the fundamental theorem of calculus, we obtain

$$F(y) - F(w) = \int_{t=0}^{1} \langle \nabla F(v(t)), y - w \rangle dt \geq 0 \enspace .$$

Monotonicity of f now follows by observing that for sets $A \subseteq B$, the corresponding vectors satisfy $x(A) \preceq x(B)$, and hence, using (15), we have

$$f(A) = F(x(A)) \leq F(x(B)) = f(B) \enspace .$$

Next, we show that f is submodular. To this end, the main technical condition that we need on F is that for all $w, y \in [0,1]^h$ and $z \succeq 0$ satisfying $w \preceq y$ and $w + z, y + z \in [0,1]^h$,

$$F(w + z) - F(w) \geq F(y + z) - F(y) \enspace . \tag{16}$$

To see that this technical condition holds, consider the line segment between w and $w+z$, which is given by $w(t) = w + t \cdot z$, where $t \in [0,1]$ is a parameterization of this line segment. Similarly, $y(t) = y + t \cdot z$ is the line segment between y and $y+z$. Since $w+z, y+z \in [0,1]^h$, all points on these two line segments are also in the unit cube and so the function F, its gradient ∇F and its Hessian $\nabla^2 F$ are all well-defined on these points. Additionally, note that for each t, $w(t) \preceq y(t)$ so that the vector $y(t) - w(t)$ has non-negative coordinates. Additionally, Lemma 48 states that $\frac{\partial^2 F}{\partial x_u \partial x_v} \leq 0$ almost everywhere for every pair $u, v \in [h]$. This means that the Hessian matrix $\nabla^2 F$ has non-positive entries, and thus,

$$(y(t) - w(t))^T \left[\nabla^2 F(v)\right] (y(t) - w(t)) \leq 0$$

for almost all $t \in [0,1]$. Define now

$$v_t(s) = (1-s) \cdot w(t) + s \cdot y(t)$$

to be the line segment between $w(t)$ and $y(t)$, and for notation convenience, define the function $H(v) = \langle \nabla F(v), z \rangle$. Using the fundamental theorem of calculus along with the chain rule and the properties of the Hessian matrix $\nabla^2 F$ above, the above observations yield

$$\begin{aligned}\langle \nabla F(y(t)), z \rangle - \langle \nabla F(w(t)), z \rangle &= H(y(t)) - H(w(t)) \\ &= \int_{s=0}^{1} \langle \nabla H(v_t(s)), y(t) - w(t) \rangle ds && \text{(f.t. of calculus)} \\ &= \int_{s=0}^{1} (y(t) - w(t))^T \left[\nabla^2 F(v_t(s))\right] (y(t) - w(t)) ds && \text{(chain rule)} \\ &\leq 0 \ ,\end{aligned}$$

so that $\langle \nabla F(w(t)), z \rangle \geq \langle \nabla F(y(t)), z \rangle$ for all $t \in [0,1]$. To prove (16), it only remains to combine the last result with the fundamental theorem of calculus and obtain

$$F(w+z) - F(w) = \int_{t=0}^{1} \langle \nabla F(w(t)), t \cdot z \rangle dt \geq \int_{t=0}^{1} \langle \nabla F(y(t)), t \cdot z \rangle dt = F(y+z) - F(y) \ .$$

Now we can use Inequality (16) to prove submodularity of f. First, observe that for sets $A \subseteq B$, the corresponding vectors $x(A)$ and $x(B)$ satisfy $x(A) \preceq x(B)$. Moreover, for any element $e \notin B$, $x(A+e) = x(A) + x(e)$ and $x(B+e) = x(B) = x(e)$. Thus, setting $w = x(A)$, $y = x(B)$ and $z = x(e)$, we have that

$$\begin{aligned}f(A+e) - f(A) &= F(x(A) + x(e)) - F(x(A)) \\ &\geq F(x(B) + x(e)) - F(x(B)) \\ &= f(B+e) - f(B) \ ,\end{aligned}$$

which establishes the submodularity of f. ■

To define our problem instances, we associate the monotone submodular objective f with the k-extendible system $\mathcal{M}$ and the monotone submodular objective g with the k-extendible systems $\mathcal{M}'$. Let us now bound the maximum values of the resulting submodular optimization problems.

Lemma 50 *The maximum value of an independent set in $\mathcal{M}$ with respect to the objective f is at least $1 - 2k/h - \varepsilon'$, and no more than $1 + \varepsilon'$.*

Proof Observation 46 guarantees the existence of an independent set $S \subseteq H_1$ in $\mathcal{M}$ of size $s \geq k(m - 2km/h)$. Using the first property of Lemma 48 and evaluating Q at the corresponding vector $x(S)$, we have that the objective value associated with this set is

$$f(S) = F(x(S)) \geq Q(x(S)) - \varepsilon' = \frac{s}{km} - \varepsilon' \geq \frac{k(m - 2km/h)}{km} - \varepsilon' = 1 - 2k/h - \varepsilon' \ .$$

This completes the proof of the first part of the lemma. To see that the second part also holds, we observe that q (and therefore, also Q) never takes values larger than 1; and thus, by the first property of Lemma 48, for every set $S' \subseteq \mathcal{N}$,

$$f(S') = F(x(S')) \leq Q(x(S')) + \varepsilon' \leq 1 + \varepsilon' \ . \qquad \blacksquare$$

Lemma 51 *The maximum value of a set in $\mathcal{M}'$ with respect to the objective g is at most $1 - e^{-1/k} + h^{-1} + \varepsilon'$.*

Proof The objective g is monotone, and thus, the maximum value set in $\mathcal{M}'$ must be of size m. Using the second and first properties of Lemma 48 and evaluating Q, we have that for every set S of size m, we get that

$$\begin{aligned} g(S) &= G(x(S)) \\ &= F(\overline{x(S)}) \\ &= F((kh)^{-1} \cdot \mathbf{1}_h) \leq Q((kh)^{-1} \cdot \mathbf{1}_h) + \varepsilon' \\ &= 1 - \left(1 - \frac{1}{kh}\right)^h + \varepsilon' \\ &\leq 1 - e^{-1/k}\left(1 - \frac{1}{k^2 h}\right) + \varepsilon' \\ &\leq 1 - e^{-1/k} + h^{-1} + \varepsilon' \ , \end{aligned}$$

where the first inequality holds because the inequality $(1 + x/n)^n \geq e^x(1 - x^2/n)$ holds whenever $n \geq 1$ and $|x| \leq n$ (here $x = -1/k$ and $n = h$), and the last inequality holds because $e^{-k}/k^2 \leq 1$ for every $k \geq 1$. $\blacksquare$

As before, our plan is to show that after a random permutation is applied to the ground set it is difficult to distinguish between the problem instances f with $\mathcal{M}$ and g with $\mathcal{M}'$. 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 47 holds regardless of the objective function. Thus, $\mathcal{M}$ and $\mathcal{M}'$ are still polynomially indistinguishable. Additionally, the next lemma shows that their associated objective functions are also polynomially indistinguishable.

Lemma 52 *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 We begin by showing that $f(S) = g(S)$ for all sets S which are made up of roughly the same number of elements from each of the partitions $H_1, H_2, \ldots H_h$. More precisely, define $X_i = |S \cap H_i(k,m)|$. We claim that if a set S satisfies $\left|X_i - \frac{|S|}{h}\right| < mk \cdot \sqrt{\frac{\delta_h}{h}}$ for every $1 \leq i \leq h$, then $f(S) = g(S)$. Note that under this condition, the norm of the difference between $x(S)$ and its symmetrization $\overline{x(S)}$ is at most

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

and thus by the third property of Lemma 48, we have that $F(x(S)) = G(x(S))$, which implies that $f(S) = g(S)$ by construction of these set functions.

We now show that following the application of a random permutation to the ground set, any given set S satisfies $\left|X_i - \frac{|S|}{h}\right| < mk \cdot \sqrt{\frac{\delta_h}{h}}$ for every $1 \leq i \leq h$ with high probability, and thus, $f(S) = g(S)$ with high probability. 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 $|S|$. 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 $2h \cdot e^{-2mk\delta_h/h^2}$. It follows now from the first part of the proof that $f(S) \neq g(S)$ with probability at most $2h \cdot e^{-2mk\delta_h/h^2}$. ■

Consider a polynomial time deterministic algorithm that gets either $\mathcal{M}$ with its corresponding objective f or $\mathcal{M}'$ with its corresponding objective g after a random permutation was applied to the ground set. Consider first the case that the algorithm gets $\mathcal{M}'$ and its corresponding objective g. 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 the value and independence oracles for every set that it checks. As before, one can observe that the sets which are queried do not depend on the permutation because the independence of a set in $\mathcal{M}'$ and its value with respect to g 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 $\mathcal{M}$ 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 oracle queries (different than what is expected for $\mathcal{M}'$) or it will finish all the queries. Note that in the later case the algorithm must return the same answer that it would have returned had it been given $\mathcal{M}'$.

By the union bound, Lemmata 47 and 52 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 (Yao, 1977; Borodin and El-Yaniv, 1998) this observation extends also to polynomial time randomized algorithms.

Using these ideas, we are now ready to prove Theorem 44.

Proof *(Proof of Theorem 44)*: Consider an algorithm that seeks to maximize $f(S)$ over the k-extendible system $\mathcal{M}$ after the random permutation was applied, and let T be its output set. Moreover, the algorithm fails, with high probability, to distinguish between $\mathcal{M}$ and $\mathcal{M}'$. Thus, with high probability T is independent in $\mathcal{M}'$ and has the same value under both objective functions f and g which implies by Lemma 51 that

$$f(S) = g(S) \leq 1 - e^{-1/k} + h^{-1} + \varepsilon'.$$

Since Lemma 50 shows that even in the rare case in which the algorithm does mamange to distinguish between the functions, still $f(S) \leq 1 + \varepsilon'$, this implies

$$\mathbb{E}[f(T)] \leq 1 - e^{-1/k} + h^{-1} + \varepsilon' + o(1) \enspace ,$$

where the $o(1)$ term represents a value that goes to zero when m goes to infinity assuming k and h are kept constant.

On the other hand, Lemma 50 shows that $\mathcal{M}$ contains an independent set S whose objective value $f(S)$ is at least $1-2k/h-\varepsilon'$. Thus, the approximation ratio of the algorithm is no better than

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

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 $\varepsilon > 0$. ■

8. Practical Considerations and the `SubmodularGreedy.jl` package

In this section, we discuss practical considerations for practitioners interested in using SimultaneousGreedys, RepeatedGreedy, and their variants. We also present `SubmodularGreedy.jl`, an open source Julia package which implements these simultaneous and repeated greedy techniques, their variants, and practical heuristics.

8.1 Practical Considerations: Simultaneous or Repeated?

In this paper, we have proposed simultaneous and repeated greedy techniques. The most natural question is: which algorithmic technique is better in practice? Given enough computational resources, executing both algorithms for a variety of parameter settings is most likely to yield the best solution. Still, it is of practical interest to know which of the two algorithms will generally perform better. One may be tempted to judge the effectiveness of these algorithms by comparing the approximation ratios which we derived in the preceding sections. This line of thinking would suggest that SimultaneousGreedys is clearly the better choice in practice; however, we caution practitioners against making these judgments based solely on the approximation ratios as they are based on worst-case analysis and may not reflect typical problem instances.

We argue that RepeatedGreedy may be more reasonable to use in practice because it requires less parameter tuning and is guaranteed to return a solution which is at least as good as the greedy solution. Both SimultaneousGreedys and RepeatedGreedy require setting the main parameter ℓ, which is roughly the number of candidate solutions produced in both algorithms. Although our worst-case analysis yields natural choices of ℓ based on properties of the independence system, it is ultimately a parameter which is to be set by the user. The set of solutions produced by SimultaneousGreedys will generally be quite different for varying ℓ and so the resulting approximation performance of the algorithm really does depend on the choice of ℓ. In contrast, RepeatedGreedy produces the same sequence of solutions as ℓ increases, and so the approximation can only improve as ℓ increases. In this sense, ℓ is simpler to tune when using RepeatedGreedy. In Section 9, we observe that the quality of the solution returned by SimultaneousGreedys varies non-monotonically with the number of solutions ℓ. One way to address this is to run SimultaneousGreedys for a sequence of $\ell = 1, 2, \ldots, \ell_{\max}$, and return the best solution.

Another reason to use RepeatedGreedy in practice is that it is at least as effective as the greedy algorithm; on the other hand, SimultaneousGreedys may perform worse than the greedy algorithm if ℓ is set too large. Finally, we conjecture that the approximation ratio of RepeatedGreedy adapts to the curvature of the submodular objective function for all values of ℓ, while the approximation ratio of SimultaneousGreedys adapts to curvature only for a restricted set of ℓ values; however, this is beyond the scope of the current paper.

The so-called "lazy greedy" search (Minoux, 1978) is a well-known heuristic for running iterative greedy searches which dramatically speeds up any greedy-based algorithm for submodular optimization, including the simultaneous and repeated greedy algorithms presented here. Rather than computing the marginal gain of each element in the ground set, the lazy greedy approach for greedy search exploits submodularity by maintaining a priority queue of each element along with its previously queried marginal gain. Because the marginal gain of an element is non-increasing as the solution set grows, previously queried marginal gains are an upper bound for the current marginal gain. In this way, the lazy greedy approach (typically) results in only a few oracle queries until the element with the top marginal gain is found. Although the lazy greedy approach does not improve worst-case runtime, it greatly improves the runtime for many practical instances. Moreover, the lazy greedy approach may be used together with the marginal gain thresholding technique for improved performance gains. When using a simultaneous greedy algorithm, the lazy greedy priority queue should be modified to include element-solution pairs.

8.2 The `SubmodularGreedy.jl` package

Our final main contribution in this paper is `SubmodularGreedy.jl`, an open source Julia package which implements the simultaneous and repeated greedy algorithms described here, along with their nearly linear time and knapsack variants. We have written the package so that it is easy to use "out-of-the-box", requiring little to no knowledge of the algorithmic variants such as marginal gain thresholding and density ratio thresholding. The package is available at this URL[4] and the installation requires only one line of code using the Julia package manager. Below, we highlight a few of the design decisions in the package:

- **Supported Algorithms**: The `SubmodularGreedy.jl` package supports all algorithms presented in this paper, including SimultaneousGreedys, RepeatedGreedy, and their nearly linear-time and knapsack variants. Additionally, the SampleGreedy algorithm of Feldman et al. (2017) is also included.

- **Oracle Models**: Our implementations run in the *oracle model.* That is, the user provides a value oracle which returns $f(S)$ given S and a independence oracle which, given S, determines whether or not $S \in \mathcal{I}$. In this way, faster implementations of these oracles will result in faster run time of our algorithms.

- **Default Parameter Settings**: The various parameters of the algorithms are set by default to values suggested by our worst case analysis. The user may specify whether the independence system is k-extendible or a k-system and whether the objective is monotone, and the number of candidate solutions ℓ is set automatically. Additionally, the β scaling terms used in the binary density search are also set based on the analysis in this paper. However, the user may override any of these default parameter settings in favor of their own.

- **Lazy Greedy Implementations**: All of our implementations feature a lazy greedy approach (discussed above in Section 8.1) for improved practical performance. More-

4. https://github.com/crharshaw/SubmodularGreedy.jl

over, the simultaneous greedy algorithm features a lazy greedy priority queue whose keys are element-solution pairs, which is more appropriate for this setting.

The following functions are available in the `SubmodularGreedy.jl` package. A more comprehensive description of these functions is contained in the documentation of the package. Additionally, we have included a tutorial of the package as a Jupyter notebook.

- `simultaneous_greedys`: A fast implementation of SimultaneousGreedys using approximate greedy search and lazy evaluations. If knapsack constraints are given, the density threshold technique is used with default parameter settings.
- `repeated_greedy`: A fast implementation of RepeatedGreedy using approximate greedy search and lazy evaluations. If knapsack constraints are given, the density threshold technique is used with default parameter settings.
- `sample_greedy`: An implementation of the SampleGreedy algorithm of Feldman et al. (2017) using lazy evaluations.
- `greedy`: A fast implementation of the Greedy algorithm using approximate greedy search and lazy evaluations. If knapsack constraints are given, the density threshold technique is used with default parameter settings.
- `deterministic_usm`: An implementation of the deterministic linear-time USM algorithm of Buchbinder et al. (2014).

9. Experiments

In this section, we demonstrate the efficacy of our proposed algorithms on two movie recommendation settings using a real dataset. The `SubmodularGreedy.jl` package contains all implementations of algorithms used in this experimental section.

9.1 MovieLens 20M Dataset

In our experiments, we use data from the MovieLens 20M Dataset, which features 20 million ratings of 27,000 movies by 138,000 users. For each movie, we construct a corresponding feature vector v_i by using a low-rank matrix completion technique on the user reviews, as proposed by Lindgren et al. (2015). The feature vectors are a low dimensional representation of the movies, based on the available user reviews. For a pair of movies i and j, we use the feature vectors to construct a similarity score

$$s_{i,j} = \exp\left(-\sigma^2(1 - \cos(v_i, v_j))\right) \quad ,$$

where $\cos(v_i, v_j) = \langle v_i, v_j \rangle / (\|v_i\| \|v_j\|)$ is the cosine similarity and $\sigma > 0$ is a user-defined bandwidth parameter which controls the decay of this similarity. In this way, the similarity scores are based on the rating behavior of the users in the MovieLens 20M dataset. We remark that the similarity scores are in the range $[0, 1]$, where $s_{ij} = 1$ only if v_i is a scaled multiple of v_j.

The MovieLens dataset also contains, for each movie, a list of genres that the movie belongs to. There are 17 total genres, including Action, Drama, Comedy, Thriller, Musical, and Western, to name a few. We emphasize that each movie belongs to at least one genre, but typically several genres. By scraping the Internet Movie Database (IMDb), metadata on the movies is collected, including the release year and the average IMDb user rating. Metadata is collected for $n = 10,473$ movies and so this is the size of the ground set.

In both experiments, we use the following non-monotone submodular objective function.

$$f(S) = \frac{1}{n}\left[\sum_{i\in\mathcal{N}}\sum_{j\in S} s_{i,j} - \lambda\cdot\sum_{i\in S}\sum_{j\in S} s_{i,j}\right] ,$$

where $\lambda \in [0,1]$ is a user-defined penalty term. The first term captures the extent to which the set S summarizes the entirety of movies in the ground set, while the second term penalizes sets S which have a lot of self-similarity. When $\lambda = 1$, the objective function recovers the graph-cut function on the graph in which edges weights are the normalized similarities, i.e., $s_{i,j}/n$.

9.2 Experiment 1: Movie Recommendation with Genre Limitations

In the first experiment, we aim to provide a user with a movie summarization set in which no genre appears too frequently. This modeling formulation is most suitable for a user who wants a diverse selection of movies from the dataset, in terms of both the MovieLens user ratings and the genres.

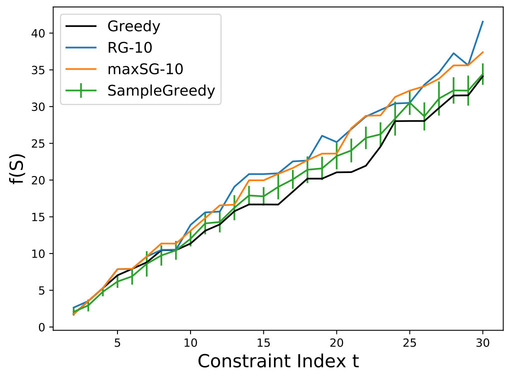
Let $\mathcal{G}$ denote the set of movie genres. For each movie $e \in \mathcal{N}$, let $G_e \subseteq \mathcal{G}$ denote the genres that movie e belongs to. For each genre $g \in \mathcal{G}$, let d_g be a non-negative integer. We define the *genre-limiting* constraint set $(\mathcal{I}, \mathcal{N})$, where $S \in \mathcal{I}$ if

$$|\{e \in S : g \in G_e\}| \leq d_g \text{ for each genre } g \in \mathcal{G}.$$

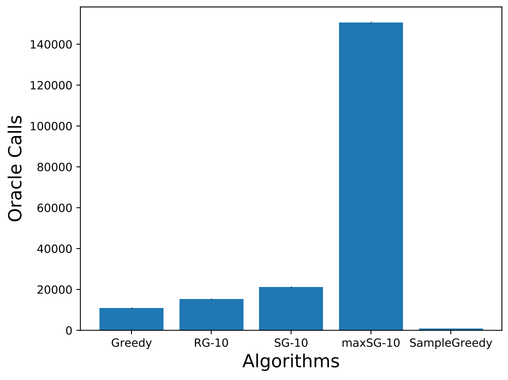
In other words, the solution set S contains at most d_g movies belonging to genre $g \in \mathcal{G}$. One can verify that this constraint set is a k-extendible system, where $k = |\mathcal{G}|$. In particular, the genre limiting constraint is the intersection of $|\mathcal{G}|$ partition matroids.

In this experiment, we consider a sequence of problem instances, defined by a sequence of constraint sets. For each genre $g \in \mathcal{G}$, we define a *genre fraction limit* $q_g \in [0,1]$. For an integer t, we define genre limits according to the genre fraction limits by $d_g = \text{Round}(t \cdot q_g)$. In this way, we define a sequence of growing constraint sets which are indexed by integers $t \in \mathbb{N}$. The choice of genre fraction limits encode a user's desired fraction of genres in the summary, while the integer t roughly determines the size of the summary. In our experiment, we choose the genre fraction limits of most genres to reflect the total fraction of movies belonging to the genre, i.e., $q_g = |\{e \in \mathcal{N} : g \in G_e\}|/n$. The exceptions are Crime, Drama and Thriller which have slightly higher genre fraction limits and Animation, Children, Romance, and Horror which have slightly lower genre fraction limits. These modified genre fraction limits may be understood to represent a particular user's personal interest. In our experiments, the modified genre fraction limits introduce local minima in which the greedy algorithm, but not the more sophisticated variants, get stuck.

We compare SimultaneousGreedys, RepeatedGreedy, Greedy, and Sample-Greedy for the sequence of problem instances in this experiment. We run these algorithms



(a)



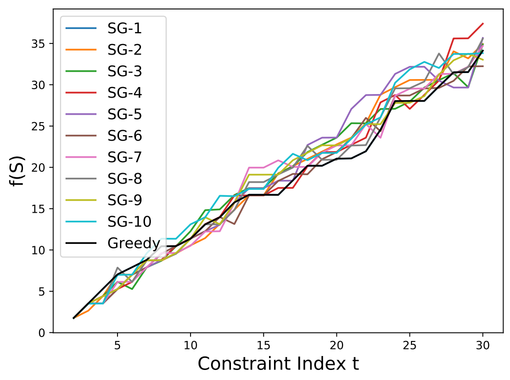
(b)

Figure 2: A comparison of objective value and runtime of Greedy, Sample Greedy, Repeated Greedy, and Simultaneous Greedys for problem instances in Experiment 1. Linear time implementations with $\varepsilon \in \{0.1, 0.01\}$ were run, but not reported here because the execution path did not change. Fig 2a plots the objective value attained by the algorithms against the constraint index. Fig 2b plots the number of oracle calls for index $t = 30$.

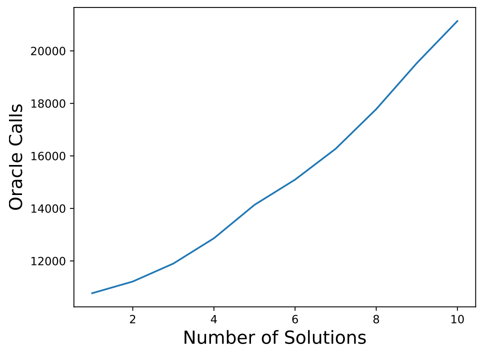
for problem instances with indices $t = 2, 3, \ldots, 30$. For each algorithm, we record the objective value of the returned solution and the required number of oracle calls for each of the problem instances. As recommended in Section 8.1, we take the maximum of SimultaneousGreedys over setting $\ell = 1, 2, \ldots, 10$. For comparison, we set the number of solutions to $\ell = 10$ when running RepeatedGreedy. We ran the linear time implementations with $\varepsilon \in \{0.01, 0.1\}$, but the execution paths of the algorithms remained unchanged (compared to the non-linear time implementations); this is likely a result of the lazy greedy implementation. For this reason, the linear time implementations are not included in these results. We ran SampleGreedy for 20 iterations. Figure 2 contains the results of this experiment.

As we see in Figure 2a, RepeatedGreedy and taking the maximum of $\ell = 1, 2, \ldots 10$ of SimultaneousGreedys return higher quality solutions than the greedy algorithm. In fact, RepeatedGreedy and SimultaneousGreedys return solutions with larger value than the expected value of the solution returned by SampleGreedy. Figure 2b shows the number of oracle calls made by the various algorithms. SimultaneousGreedys and RepeatedGreedy require more oracle calls that Greedy and taking the maximum over $\ell = 1, 2, \ldots, 10$ increases this cost. On the other hand, the expected cost of SampleGreedy is considerably lower than the other algorithms.

Figure 3 demonstrates the behavior of SimultaneousGreedys as the number of solutions ℓ is varied. Figure 3a shows the objective value attained by SimultaneousGreedys while varying the number of solutions $\ell = 1, 2, \ldots, 10$. For most values of ℓ, the attained objective value is larger than that of the greedy algorithm; however, we see that there is no value of ℓ which consistently returns the highest value of the objective function. As we see in Figure 3b, the number of oracle calls increases with the number of solutions, which is to be expected.



(a)



(b)

Figure 3: A comparison of objective value and runtime of Simultaneous Greedys when the number of solutions ℓ is varied, for problem instances in Experiment 1. Fig 3a plots the objective values attained by the various Simultaneous Greedys executions against the constraint index. Fig 3b plots the number of oracle calls for index $t = 30$ against the number of solutions ℓ.

9.3 Experiment 2: Movie Recommendations with Release Dates and Rating Budget

In the second experiment, we aim to provide a user with a movie summarization set in which movies are far apart in release date and not too many highly rated movies appear. Our modeling formulation is most suitable for a film watching party based on poorly rated films or "cult classics' throughout the years'.

In this experiment, we use release date constraint and a rating budget, defined as follows. For each movie $e \in \mathcal{N}$, let y_e denote the release year of the movie. We define the *release date constraint* $(\mathcal{I}, \mathcal{N})$, where $S \in \mathcal{I}$ if

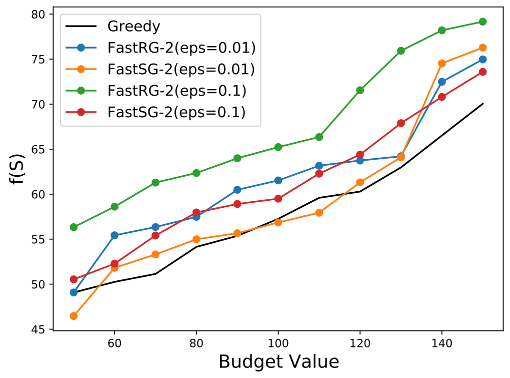
$$|y_e - y_u| \geq 1 \text{ for all pairs } e, u \in S \enspace .$$

In other words, no two movies in a feasible solution set may be released in the same year. This independence set is 2-extendible, as adding a movie e to the current solution requires the removal of up to 2 other movies that already belong to the set: one that appears up to a year after y_e and one that appears up to a year before y_e.

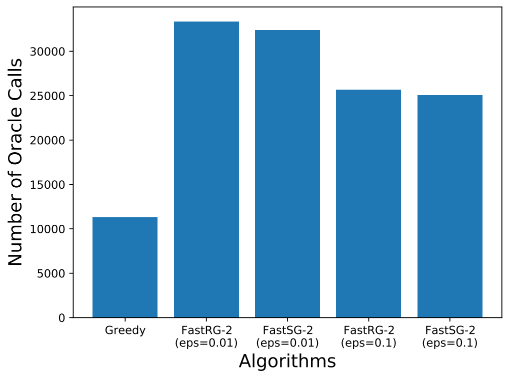
For each movie $e \in \mathcal{N}$, we let r_e denote the rating of the movie, according to the IMDb. The ratings take real values between 1 and 10. The *rating budget constraint* is that

$$\sum_{e \in S} \max(r_e - 5.0, 0) \leq \beta \enspace ,$$

where β is a user-defined *rating budget*. A set S satisfies the rating budget constraint so long as it does not contain too many highly rated movies; indeed, this constraint does not penalize movies which have a rating less than 5.0. Observe that the rating budget is a knapsack constraint with coefficients $c_e = \max(r_e - 5.0, 0)$.



(a)



(b)

Figure 4: A comparison of objective value and runtime of Greedy, Repeated Greedy, and Simultaneous Greedys for problem instances in Experiment 2. The linear time and knapsack variants of Repeated Greedy and Simultaneous Greedys are displayed here. Fig 4a plots the objective values attained by the algorithms against the budget value. Fig 4b plots the number of oracle calls for budget value $\beta = 150$

We compare DensitySearchSGS, DensitySearchRG, and Greedy for maximizing the diverse summarization objective over the release date and rating budget constraints. Recall that DensitySearchSGS and DensitySearchRG incorporate the density threshold and density search techniques for handling the knapsack objective, while Greedy incorporates the knapsack constraint into the independence constraint. For both DensitySearchSGS and DensitySearchRG, we set the number of solutions to $\ell = 2$ and we use values $\delta = \varepsilon \in \{0.1, 0.01\}$.

The results of the second experiment are summarized in Figure 4. In Figure 4a, we see that DensitySearchSGS and DensitySearchRG typically yield solutions with larger objective value than Greedy. This improvement may be attributed to the density thresholding technique, where an element with high marginal gain may not be chosen if its knapsack cost is relatively larger. Interestingly, larger values of the error term ε yields solutions with larger objective values. This is likely due to increased variability in the execution path of the algorithm, leading to more diverse solutions being constructed. In Figure 4b, we see that the density search techniques are more expensive than the greedy algorithm, especially after the lazy greedy implementation. However, the cost of the density search techniques decreases as the error term ε increases, due (in part) to fewer calls to the fixed-density subroutine.

10. Conclusion

In this paper, we have presented SimultaneousGreedys, a new algorithmic technique for constrained submodular maximization. In addition, we have improved the analysis of RepeatedGreedy, showing that fewer repeated iterations yield a better approximation than what was previously known to be possible. We have shown that both greedy-based techniques can accommodate several variants, including a nearly-linear implementation and the handling of additional knapsack constraints. Perhaps most surprisingly, the simple SimultaneousGreedys algorithmic technique provides the tightest known approximation guarantees across a mix and match of many settings: k-system constraints, k-extendible constraints, m additional knapsack constraints, non-monotone objectives, and monotone objectives. We have provided two kinds of negative results: the first is hardness results demonstrating that, for several of these settings, no efficient algorithm can achieve a significantly better approximation ratio. The second is a result which shows that our analysis of RepeatedGreedy is tight in the sense that it cannot be improved for the subclasses considered here.

We also provided practical insights, arguing that although SimultaneousGreedys has better worst-case approximation guarantees, RepeatedGreedy is often better suited for practical applications. Implementations of all the algorithms considered in this paper appear in `SubmodularGreedy.jl`, an open source Julia package which is available for download at this URL[5]. We hope that these simple, yet theoretically sound, techniques becomes a standard in the toolbox of practitioners across a variety of disciplines. In a larger sense, we hope that this technique may aid the flexibility of the submodular optimization framework as more exciting applications continue to emerge.

Acknowledgments

We graciously thank Erik Lindgren and Ehsan Kazemi for sharing the feature vectors and the scraped IMDb meta-data used in our experiments. The work of Moran Feldman was supported in part by ISF grants no. 1357/16 and 459/20. This work was supported in part by an NSF Graduate Research Fellowship (DGE1122492) awarded to Christopher Harshaw. Amin Karbasi is partially supported by NSF (IIS- 1845032), ONR (N00014-19-1-2406), and TATA Sons Private Limited.

5. https://github.com/crharshaw/SubmodularGreedy.jl

Appendix A. Proof of Proposition 23 (KnapsackSGS)

In this section, we prove Proposition 23, which is the main technical lemma behind the SimultaneousGreedys variants FastSGS and KnapsackSGS. The proposition is a meta-analysis that reduces the conditions of approximation to simple combinatorial statements relating the constructed solutions to OPT. Furthermore, the proposition and its proof mirror Proposition 4, which provided a similar meta-analysis for approximation guarantees of SimultaneousGreedys. We remind the reader that the constructions in Section 3.2 and Section 3.3 demonstrate how these conditions are satisfied for k-extendible systems and k-systems, respectively.

We begin by restating the proposition.

Proposition 23 *Suppose that there exists sets $O_i^{(j)}$ for every iteration $0 \leq i \leq T$ and solution $1 \leq j \leq \ell$ and a value p which satisfy the following properties:*

- $O_0^{(j)} = OPT$ *for every solution* $1 \leq j \leq \ell$.
- $S_i^{(j)} + u \in \mathcal{I}$ *for every iteration* $0 \leq i \leq T$, *solution* $1 \leq j \leq \ell$, *and element* $u \in O_i^{(j)}$.
- $O_i^{(j)} \subseteq O_{i-1}^{(j)} \cap \mathcal{N}_i$ *for every iteration* $1 \leq i \leq T$ *and solution* $1 \leq j \leq \ell$.
- $(S_T^{(j)} \setminus S_i^{(j)}) \cap OPT \subseteq O_i^{(j)}$ *for every iteration* $0 \leq i \leq T$ *and solution* $1 \leq j \leq \ell$.
- $\sum_{j=1}^{\ell} |O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})| \leq p$ *for every iteration* $1 \leq i \leq T$.

Then, the solution S produced by KnapsackSGS *satisfies the following approximation guarantees:*

$$f(S) \geq \begin{cases} \frac{1}{2}\rho & \text{if } E = 1 \ , \\ \frac{1-\epsilon}{p+1} \cdot \Big(\left(1 - \ell^{-1} - \varepsilon\right) f(OPT) - m\rho \Big) & \text{if } E = 0 \ . \end{cases} \tag{7}$$

Moreover, when f is monotone, these approximation guarantees improve to

$$f(S) \geq \begin{cases} \frac{1}{2}\rho & \text{if } E = 1 \ , \\ \frac{1-\epsilon}{p+1} \cdot \Big(\left(1 - \varepsilon\right) f(OPT) - m\rho \Big) & \text{if } E = 0 \ . \end{cases} \tag{8}$$

As before, at each iteration $1 \leq t \leq T$, the set $O_i^{(j)}$ contains the elements of OPT which maintain feasbility in the independence system when added to solution $S_i^{(j)}$. It may be the case, however, that some of these elements of $O_i^{(j)}$ are infeasible to add to the corresponding solution with respect to the knapsack constraints. We note also that there are several differences between the proof of Proposition 23 and the proof of the earlier Proposition 4. The most significant difference is that there is now a case analysis depending on whether or not Line 7 of KnapsackSGS ever evaluates to false, which is denoted by the indicator variable E. If $E = 1$, then a simple argument lower bounds the quality of the returned solution; and if $E = 0$, then we obtain an approximation guarantee using similar techniques to those used in the proof of Proposition 4.

In the case of $E = 0$, the proof techniques differ in a few ways: first, the elements of $O_i^{(j)}$ are typically broken up into two groups: those with high density with respect to the current solution and those with low density. In the analysis, the two groups of elements are considered separately. Second, the greedy search is now approximate (up to a factor $(1 - \varepsilon)$), and so this factor carries through the analysis. Finally, the remaining elements of $O_T^{(j)}$ after termination may have positive marginal gain when added to the constructed solution, but the gain is sufficiently small so that it does not greatly decrease the quality of the constructed solution.

Before continuing, let us set up some notation to split the elements of OPT that we throw away into high and low density. Recall that at each iteration i, $O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})$ are the elements of OPT which must be removed so that element u_i may be added to solution $S_i^{(j_i)}$. Of these elements we must throw away, we will distinguish between those with high density and those with low density. In particular, we will define $\mathcal{H}_i^{(j)}$ to be those elements of high density with respect to solution $S_i^{(j)}$ and $\mathcal{L}_i^{(j)}$ to be those elements of low density with respect to solution $S_i^{(j)}$. More formally, for any solution $1 \leq j \leq \ell$ and iteration $1 \leq i \leq T$, we define the sets

$$\mathcal{H}_i^{(j)} = \left\{ u \in O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)}) : f(u \mid S_i^{(j)}) \geq \rho \cdot \sum_{r=1}^{m} c_r(u) \right\}$$
$$\mathcal{L}_i^{(j)} = \left[O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)}) \right] \setminus \mathcal{H}_i^{(j)} \enspace .$$

The following lemma is the first step towards proving Proposition 23. Intuitively, this lemma shows that as the iteration i increases, the decrease in the value of $f(O_i^{(j)} \mid S_i^{(j)})$ is transferred, at least to some extent, to $S_i^{(j)}$.

Lemma 53 *Given the conditions of Proposition 23, if $E = 0$, then for every iteration $0 \leq i \leq T$,*

$$\frac{(p+1)}{(1-\varepsilon)} \cdot \sum_{j=1}^{\ell} f(S_i^{(j)}) + \sum_{j=1}^{\ell} f(O_i^{(j)} \mid S_i^{(j)}) \geq \sum_{j=1}^{\ell} f(OPT \cup S_i^{(j)}) - \rho \sum_{t=1}^{i} \sum_{j=1}^{\ell} \sum_{r=1}^{m} c_r(\mathcal{L}_i^{(j)}) \enspace .$$

Proof We prove the lemma by induction on the iterations $i = 0, 1, \ldots, T$. The base case is the case of $i = 0$, corresponding to the initialization of the algorithm. Recall that the solutions are initialized to be empty, i.e., $S_0^{(j)} = \varnothing$ for every $j \in [\ell]$. This, together with non-negativity of f, implies that

$$\begin{aligned}
\sum_{j=1}^{\ell} f(OPT \cup S_0^{(j)}) &= \sum_{j=1}^{\ell} f(OPT \cup \varnothing) && \text{(by the initialization } S_0^{(j)} = \varnothing) \\
&= \sum_{j=1}^{\ell} f(\varnothing) + \sum_{j=1}^{\ell} f(OPT \mid \varnothing) && \text{(rearranging terms)} \\
&\leq \frac{(p+1)}{(1-\varepsilon)} \cdot \sum_{j=1}^{\ell} f(\varnothing) + \sum_{j=1}^{\ell} f(OPT \mid \varnothing) && (f(\varnothing) \geq 0 \text{ by the non-negativity}) \\
&\leq \frac{(p+1)}{(1-\varepsilon)} \cdot \sum_{j=1}^{\ell} f(S_0^{(j)}) + \sum_{j=1}^{\ell} f(O_0^{(j)} \mid S_0^{(j)}) \enspace . && \text{(by the initialization } S_0^{(j)} = \varnothing)
\end{aligned}$$

This establishes the base case as the right term appearing on the right hand side of the lemma's inequality is zero when $i = 0$.

Assume now that the lemma holds for all iterations $i - 1 \geq 0$, and let us prove it for iteration i. Recall that only the solution $S_i^{(j_i)}$ is modified during iteration i. Thus, we have that the change in iteration i in the first sum in the guarantee of the lemma is

$$\frac{(p+1)}{(1-\varepsilon)} \cdot \sum_{j=1}^{\ell} f(S_i^{(j)}) - \frac{(p+1)}{(1-\varepsilon)} \cdot \sum_{j=1}^{\ell} f(S_{i-1}^{(j)}) = \frac{(p+1)}{(1-\varepsilon)} \cdot f(u_i \mid S_{i-1}^{(j_i)}) \ . \tag{17}$$

Bounding the change in the second sum in the guarantee is more involved, and is done in three steps. The first step is the following inequality.

$$\begin{aligned}
&\sum_{j=1}^{\ell} f(O_{i-1}^{(j)} \mid S_{i-1}^{(j)}) - \sum_{j=1}^{\ell} f(O_{i-1}^{(j)} \mid S_i^{(j)}) && (18)\\
&\qquad = f(O_{i-1}^{(j_i)} \mid S_{i-1}^{(j_i)}) - f(O_{i-1}^{(j_i)} \mid S_i^{(j_i)}) && (\text{only } S_i^{(j_i)} \text{ is modified})\\
&\qquad = f(u_i \mid S_{i-1}^{(j_i)}) - f(u_i \mid O_{i-1}^{(j_i)} \cup S_{i-1}^{(j_i)}) && (\text{rearranging terms})\\
&\qquad \leq f(u_i \mid S_{i-1}^{(j_i)}) - f(u_i \mid OPT \cup S_{i-1}^{(j_i)}) \ ,
\end{aligned}$$

where the inequality may be proved by considering two cases. First, suppose that $u_i \in O_{i-1}^{(j_i)} \cup S_{i-1}^{(j_i)}$. In this case, the inequality holds with equality, because $O_{i-1}^{(j_i)} \subseteq OPT$ by assumption. Consider now the case in which $u_i \notin O_{i-1}^{(j_i)} \cup S_{i-1}^{(j_i)}$. In this case, our assumption that $(S_T^{(j_i)} \setminus S_{i-1}^{(j_i)}) \cap OPT \subseteq O_{i-1}^{(j_i)}$ implies $u_i \notin (S_T^{(j_i)} \setminus S_{i-1}^{(j_i)}) \cap OPT$, which implies in its turn $u_i \notin OPT$ since $u_i \in S_i^{(j_i)} \subseteq S_T^{(j_i)}$ and $u_i \in \mathcal{N}_{i-1} \subseteq \mathcal{N} \setminus S_{i-1}^{(j_i)}$. Therefore, we get that in this case that Inequality (18) holds due to the submodularity of f (recall that $O_{i-1}^{(j_i)} \subseteq OPT$ by our assumption).

For the second step in the proof of the above mentioned bound, we use submodularity to bound the marginal gain $f(O_{i-1}^{(j)} \mid S_i^{(j)})$ using sums of marginal gains of single elements. Observe that

$$\begin{aligned}
\sum_{j=1}^{\ell} f(O_{i-1}^{(j)} \mid S_i^{(j)}) &\leq \sum_{j=1}^{\ell} f(O_i^{(j)} \mid S_i^{(j)}) + \sum_{j=1}^{\ell} \sum_{u \in O_{i-1}^{(j)} \setminus O_i^{(j)}} f(u \mid S_i^{(j)}) && (\text{submodularity, } O_i^{(j)} \subseteq O_{i-1}^{(j)})\\
&= \sum_{j=1}^{\ell} f(O_i^{(j)} \mid S_i^{(j)}) + \sum_{j=1}^{\ell} \sum_{u \in O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})} f(u \mid S_i^{(j)}) \ . && (U_i^{(j)} \subseteq S_i^{(j)})
\end{aligned}$$

The third step is to analyze the inner sum above by partitioning the elements $u \in O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})$ based on their density, i.e. into the two sets $\mathcal{H}_i^{(j)}$ and $\mathcal{L}_i^{(j)}$ (recall that these two sets are indeed a partition of $O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})$).

$$\sum_{u \in O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})} f(u \mid S_i^{(j)}) = \sum_{u \in \mathcal{H}_i^{(j)}} f(u \mid S_i^{(j)}) + \sum_{u \in \mathcal{L}_i^{(j)}} f(u \mid S_i^{(j)})$$

The second sum may be bounded by virtue of the low density of its elements, as

$$\sum_{u \in \mathcal{L}_i^{(j)}} f(u \mid S_i^{(j)}) \leq \sum_{u \in \mathcal{L}_i^{(j)}} \rho \cdot \sum_{r=1}^{m} c_r(u) = \rho \sum_{r=1}^{m} c_r(\mathcal{L}_i^{(j)}) \ .$$

Recall now that by the approximate greedy search, the element-solution pair $(u_i, S_i^{(j_i)})$ has the property that $f(u_i \mid S_i^{(j_i)}) \geq (1-\varepsilon) f(u \mid S_i^{(j)})$ for all element-solution pairs $(u, S_i^{(j)})$ where $S_i^{(j)} + u$ is feasible with respect to independence system, and u has high density with respect to $S_i^{(j)}$. In particular, we have that $f(u_i \mid S_i^{(j_i)}) \geq (1-\varepsilon) f(u \mid S_i^{(j)})$ for all $u \in \mathcal{H}_i^{(j)}$. This yields an upper bound on the first sum,

$$\sum_{u \in \mathcal{H}_i^{(j)}} f(u \mid S_i^{(j)}) \leq \sum_{u \in \mathcal{H}_i^{(j)}} (1-\varepsilon)^{-1} f(u_i \mid S_i^{(j_i)}) = (1-\varepsilon)^{-1} f(u_i \mid S_i^{(j_i)}) \cdot |\mathcal{H}_i^{(j)}| \enspace .$$

Combining the upper bounds we have obtained on the sums corresponding to $\mathcal{L}_i^{(j)}$ and $\mathcal{H}_i^{(j)}$ yields

$$\begin{aligned}
\sum_{j=1}^{\ell} \sum_{u \in O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})} f(u \mid S_i^{(j)}) &\leq \rho \sum_{j=1}^{\ell} \sum_{r=1}^{m} c_r(\mathcal{L}_i^{(j)}) + \sum_{j=1}^{\ell} (1-\varepsilon)^{-1} f(u_i \mid S_i^{(j_i)}) \cdot |\mathcal{H}_i^{(j)}| \\
&= \rho \sum_{j=1}^{\ell} \sum_{r=1}^{m} c_r(\mathcal{L}_i^{(j)}) + (1-\varepsilon)^{-1} f(u_i \mid S_i^{(j_i)}) \cdot \sum_{j=1}^{\ell} |\mathcal{H}_i^{(j)}| \qquad \text{(rearranging)} \\
&\leq \rho \sum_{j=1}^{\ell} \sum_{r=1}^{m} c_r(\mathcal{L}_i^{(j)}) + (1-\varepsilon)^{-1} f(u_i \mid S_i^{(j_i)}) \cdot \sum_{j=1}^{\ell} |O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})| \\
&\leq \rho \sum_{j=1}^{\ell} \sum_{r=1}^{m} c_r(\mathcal{L}_i^{(j)}) + \frac{p}{1-\varepsilon} \cdot f(u_i \mid S_i^{(j_i)}) \enspace ,
\end{aligned}$$

where the cardinality bound $|\mathcal{H}_i^{(j)}| \leq |O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})|$ in second inequality follows from the containment $\mathcal{H}_i^{(j)} \subseteq O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})$ and the last inequality follows from the final condition of the proposition which states that $\sum_{j=1}^{\ell} |O_{i-1}^{(j)} \setminus (O_i^{(j)} \cup U_i^{(j)})| \leq p$. Together with the inequality from this second step, this yields

$$\sum_{j=1}^{\ell} f(O_{i-1}^{(j)} \mid S_i^{(j)}) \leq \sum_{j=1}^{\ell} f(O_i^{(j)} \mid S_i^{(j)}) + \rho \sum_{j=1}^{\ell} \sum_{r=1}^{m} c_r(\mathcal{L}_i^{(j)}) + \frac{p}{1-\varepsilon} \cdot f(u_i \mid S_i^{(j_i)}) \enspace . \tag{19}$$

The remainder of the proof consists of combining the three inequalities (17), (18) and (19) with the induction hypothesis, as follows.

$$\begin{aligned}
&\frac{(p+1)}{(1-\varepsilon)} \cdot \sum_{j=1}^{\ell} f(S_i^{(j)}) + \sum_{j=1}^{\ell} f(O_i^{(j)} \mid S_i^{(j)}) \\
&\quad\geq \left[\frac{(p+1)}{(1-\varepsilon)} \cdot \sum_{j=1}^{\ell} f(S_{i-1}^{(j)}) + \frac{(p+1)}{(1-\varepsilon)} \cdot f(u_i \mid S_{i-1}^{(j_i)})\right] \\
&\qquad + \left[\sum_{j=1}^{\ell} f(O_{i-1}^{(j)} \mid S_i^{(j)}) - \frac{p}{(1-\varepsilon)} \cdot f(u_i \mid S_{i-1}^{(j_i)}) - \rho \sum_{j=1}^{\ell}\sum_{r=1}^{m} c_r(\mathcal{L}_i^{(j)})\right] \\
&\quad\geq \frac{(p+1)}{(1-\varepsilon)} \cdot \sum_{j=1}^{\ell} f(S_{i-1}^{(j)}) + \sum_{j=1}^{\ell} f(O_{i-1}^{(j)} \mid S_i^{(j)}) + f(u_i \mid S_{i-1}^{(j_i)}) - \rho \sum_{j=1}^{\ell}\sum_{r=1}^{m} c_r(\mathcal{L}_i^{(j)}) \\
&\quad\geq \frac{(p+1)}{(1-\varepsilon)} \cdot \sum_{j=1}^{\ell} f(S_{i-1}^{(j)}) + \sum_{j=1}^{\ell} f(O_{i-1}^{(j)} \mid S_{i-1}^{(j)}) + f(u_i \mid OPT \cup S_{i-1}^{(j_i)}) - \rho \sum_{j=1}^{\ell}\sum_{r=1}^{m} c_r(\mathcal{L}_i^{(j)}) \\
&\quad\geq \sum_{j=1}^{\ell} f(OPT \cup S_{i-1}^{(j)}) - \rho \sum_{t=1}^{i-1}\sum_{j=1}^{\ell}\sum_{r=1}^{m} c_r(\mathcal{L}_i^{(j)}) + f(u_i \mid OPT \cup S_{i-1}^{(j_i)}) - \rho \sum_{j=1}^{\ell}\sum_{r=1}^{m} c_r(\mathcal{L}_i^{(j)}) \\
&\quad= \sum_{j=1}^{\ell} f(OPT \cup S_{i-1}^{(j)}) + f(u_i \mid OPT \cup S_{i-1}^{(j_i)}) - \rho \sum_{t=1}^{i}\sum_{j=1}^{\ell}\sum_{r=1}^{m} c_r(\mathcal{L}_i^{(j)}) \\
&\quad= \sum_{j=1}^{\ell} f(OPT \cup S_i^{(j)}) - \rho \sum_{t=1}^{i}\sum_{j=1}^{\ell}\sum_{r=1}^{m} c_r(\mathcal{L}_i^{(j)})
\end{aligned}$$

where the first inequality follows from (17) and (19), the second inequality holds since $f(u_i \mid S_{i-1}^{(j_i)})$ is guaranteed to be non-negative, the third inequality follows from (18), and the fourth inequality follows by induction. ■

Corollary 54 *Given the conditions of Proposition 23, if $E = 0$, then the solutions constructed by* KnapsackSGS *satisfy the lower bound*

$$\frac{(p+1)}{(1-\varepsilon)} \sum_{j=1}^{\ell} f(S_T^{(j)}) \geq \sum_{j=1}^{\ell} f(OPT \cup S_T^{(j)}) - \varepsilon\ell\Delta_f - \rho\ell m \enspace .$$

Proof Our first step is to show that $f(O_T^{(j)} \mid S_T^{(j)})$ is negligable for every solution $1 \leq j \leq \ell$. To this end, consider any fixed solution $S_T^{(j)}$ for $1 \leq j \leq \ell$. By the termination conditions of KnapsackSGS, each element $u \in \mathcal{N}_T$ satisfies

$$f(u \mid S_T^{(j)}) < \max\left((\varepsilon/n) \cdot \Delta_f, \rho \cdot \sum_{r=1}^{m} c_r(u)\right) \enspace . \tag{20}$$

In particular, this holds for each $u \in O_T^{(j)}$, as the set $O_T^{(j)}$ is contained in $\mathcal{N}_T$. We now partition the set $O_T^{(j)}$ into two groups: the elements with high density and the elements of low density. More formally, let $\mathcal{L}_{T+1}^{(j)}$ be the elements in $O_T^{(j)}$ with low density,

$$\mathcal{L}_{T+1}^{(j)} = \left\{ u \in O_T^{(j)} : f(u \mid S_T^{(j)}) < \rho \cdot \sum_{r=1}^{m} c_r(u) \right\},$$

and define $\mathcal{H}_{T+1}^{(j)} = O_T^{(j)} \setminus \mathcal{L}_{T+1}^{(j)}$ to be the high density elements. We claim that adding any high density element in $\mathcal{H}_{T+1}^{(j)}$ to the solution $S_T^{(j)}$ has a marginal gain of at most $(\varepsilon/n) \cdot \Delta_f$. To see this, observe that because the element u has high density, (20) implies that $f(u \mid S_T^{(j)}) < (\varepsilon/n) \cdot \Delta_f$.

Using the above observations, we can now bound the marginal gain of adding $O_T^{(j)}$ to $S_T^{(j)}$ as follows.

$$\begin{aligned}
f(O_T^{(j)} \mid S_T^{(j)}) &\leq \sum_{u \in O_T^{(j)}} f(u \mid S_T^{(j)}) && \text{(submodularity)} \\
&= \sum_{u \in \mathcal{H}_{T+1}^{(j)}} f(u \mid S_T^{(j)}) + \sum_{u \in \mathcal{L}_{T+1}^{(j)}} f(u \mid S_T^{(j)}) && \text{(partitioning the sum)} \\
&\leq \sum_{u \in \mathcal{H}_{T+1}^{(j)}} \frac{\varepsilon}{n} \cdot \Delta_f + \sum_{u \in \mathcal{L}_{T+1}^{(j)}} \rho \cdot \sum_{r=1}^{m} c_r(u) && \text{(above bound)} \\
&= \frac{|\mathcal{H}_{T+1}^{(j)}|}{n} \cdot \varepsilon \Delta_f + \rho \sum_{r=1}^{m} c_r(\mathcal{L}_{T+1}^{(j)}) \\
&\leq \varepsilon \Delta_f + \rho \sum_{r=1}^{m} c_r(\mathcal{L}_{T+1}^{(j)}).
\end{aligned}$$

Substituting the above bound into the guarantee of Lemma 53 for the final iteration $i = T$ implies

$$\begin{aligned}
\frac{(p+1)}{(1-\varepsilon)} \cdot \sum_{j=1}^{\ell} f(S_T^{(j)}) &\geq \sum_{j=1}^{\ell} f(OPT \cup S_i^{(j)}) - \sum_{j=1}^{\ell} f(O_T^{(j)} \mid S_T^{(j)}) - \rho \sum_{t=1}^{T} \sum_{j=1}^{\ell} \sum_{r=1}^{m} c_r(\mathcal{L}_t^{(j)}) \\
&\geq \sum_{j=1}^{\ell} f(OPT \cup S_i^{(j)}) - \sum_{j=1}^{\ell} \left[\varepsilon \Delta_f + \rho \sum_{r=1}^{m} c_r(\mathcal{L}_{T+1}^{(j)}) \right] - \rho \sum_{t=1}^{T} \sum_{j=1}^{\ell} \sum_{r=1}^{m} c_r(\mathcal{L}_t^{(j)}) \\
&= \sum_{j=1}^{\ell} f(OPT \cup S_i^{(j)}) - \varepsilon \ell \Delta_f - \rho \sum_{t=1}^{T+1} \sum_{j=1}^{\ell} \sum_{r=1}^{m} c_r(\mathcal{L}_t^{(j)})
\end{aligned}$$

To complete the proof of the corollary, we need to show that $\sum_{t=1}^{T+1} \sum_{j=1}^{\ell} \sum_{r=1}^{m} c_r(\mathcal{L}_t^{(j)}) \leq \ell m$. To this end, observe that for each solution $1 \leq j \leq \ell$, the sets $\mathcal{L}_1^{(j)}, \dots \mathcal{L}_{T+1}^{(j)}$ are disjoint subsets of OPT. Also observe that OPT is a feasible solution so that it satisfies all knapsack constraints, $c_r(OPT) \leq 1$ for all $1 \leq r \leq m$. Using these facts and the modularity of the knapsack functions, we have that

$$\begin{aligned}
\sum_{t=1}^{T+1}\sum_{j=1}^{\ell}\sum_{r=1}^{m} c_r(\mathcal{L}_t^{(j)}) &= \sum_{j=1}^{\ell}\sum_{r=1}^{m}\sum_{t=1}^{T+1} c_r(\mathcal{L}_t^{(j)}) && \text{(rearranging terms)} \\
&= \sum_{j=1}^{\ell}\sum_{r=1}^{m} c_r\left(\cup_{t=1}^{T+1}\mathcal{L}_t^{(j)}\right) && \text{(disjointedness, modularity)} \\
&\leq \sum_{j=1}^{\ell}\sum_{r=1}^{m} c_r\left(OPT\right) && (\cup_{t=1}^{T+1}\mathcal{L}_t^{(j)} \subseteq OPT) \\
&\leq \sum_{j=1}^{\ell}\sum_{r=1}^{m} 1 && (\text{feasibility of } OPT) \\
&= \ell m \ .
\end{aligned}$$

■

Proof *(Proof of Proposition 23)*: The analysis proceeds with two cases, depending on whether $E = 1$ or $E = 0$.

First, suppose that $E = 1$, which is to say that Line 7 evaluates to `false` at some point during the execution of the algorithm. This happens when, at some iteration i there exists a solution $S_i^{(j)}$ and a high density element u such that adding the element to this set is feasible in the independence system, but the knapsack constraint is violated. More precisely, the set $A \triangleq S_i^{(j)} + u$ is independent (i.e., $A \in \mathcal{I}$) but $c_r(S_i^{(j)} + u) > 1$ for some knapsack function $1 \leq r \leq m$. Although A itself is not feasible, we claim that $f(A) > \rho$. To this end, let us order the elements of A according to the order in which they were added to $S_i^{(j)}$, with u appearing last, i.e., $A = \{u_1, u_2, \dots u_k\}$ with $u_k = u$. For $1 \leq i \leq k$, define the sets $A_i = \{u_1, u_2, \dots u_i\}$ and $A_0 = \varnothing$. Then, we obtain the lower bound

$$f(A) = \sum_{i=1}^{k} f(u_i \mid A_{i-1}) \geq \sum_{i=1}^{k} \rho \cdot \sum_{r=1}^{m} c_r(u_i) = \rho \sum_{r=1}^{m} c_r(A) > \rho \cdot 1 = \rho \ ,$$

where the first inequality follows from the fact that each of the elements has high density when it is added to the solution and the second inequality follows from the fact that A violates at least one of the knapsack constraints.

The next step is to show that between $S_i^{(j)}$ and $\{u\}$, at least one of these has value larger than $\rho/2$. In particular, observe that

$$\max\left\{f(S_i^{(j)}), f(\{u\})\right\} \geq \frac{1}{2}\left(f(S_i^{(j)}) + f(\{u\})\right) \geq \frac{1}{2}\left(f(S_i^{(j)} + u) + f(\varnothing)\right) \geq \frac{1}{2} f(A) > \frac{\rho}{2} \ ,$$

where the first inequality bounds the maximum by the average, the second inequalities follows by submodularity, the third inequality follows by non-negativity, and the final inequality follows from the bound above.

Recall now that the algorithm returns the set S among the sets $S_T^{(1)}, \dots S_T^{(\ell)}$, and $\{e\} = \arg\max_{u \in \mathcal{N}} f(u)$ which maximizes the objective value. One can note that the final solutions have larger objective values than the solutions at iteration i (i.e., $f(S_T^{(j)}) \geq f(S_i^{(j)})$) because only elements with positive marginal gains are added to the solutions by the algorithm. We also note that by construction of e, we have that $f(\{e\}) \geq f(\{u\})$ because u is a feasible element. Together, these facts imply that

$$f(S) \geq \max\left\{S_T^{(1)}, \dots S_T^{(\ell)}, \{e\}\right\} \geq \max\left\{f(S_T^{(j)}), f(e)\right\} \geq \max\left\{f(S_i^{(j)}), f(u)\right\} > \frac{\rho}{2} \enspace ,$$

which completes our proof for the case of $E = 1$.

Next, we turn our attention to the case of $E = 0$. Recall that the algorithm returns the set S among the sets $S_T^{(1)}, \dots S_T^{(\ell)}$, and $\{e\} = \arg\max_{u \in \mathcal{N}} f(u)$ which maximizes the objective value. Therefore, to lower bound $f(S)$, it suffices to only consider the maximum over the sets $S_T^{(1)}, \dots S_T^{(\ell)}$. Applying an averaging argument to the guarantee of Corollary 54 yields

$$f(S) \geq \max\left\{S_T^{(1)}, \dots S_T^{(\ell)}\right\} \geq \frac{1}{\ell}\sum_{j=1}^{\ell} f(S_T^{(j)}) \geq \frac{(1-\varepsilon)}{(p+1)}\left(\frac{1}{\ell}\sum_{j=1}^{\ell} f(OPT \cup S_T^{(j)}) - \varepsilon\Delta_f - \rho m\right) \enspace . \tag{21}$$

Consider now a random set $\bar{S}$ chosen uniformly at random from the ℓ constructed solutions $S_T^{(1)}, S_T^{(2)}, \dots, S_T^{(\ell)}$. Since these solutions are disjoint by construction, an element can belong to $\bar{S}$ with probability at most ℓ^{-1}. Hence, by applying Lemma 7 to the submodular function $g(S) = f(OPT \cup S)$, we get

$$\frac{1}{\ell} \cdot \sum_{j=1}^{\ell} f(OPT \cup S_T^{(\ell)}) = \mathbb{E}[f(OPT \cup \bar{S})] = \mathbb{E}[g(\bar{S})] \geq (1 - \ell^{-1}) \cdot g(\varnothing) = (1 - \ell^{-1}) \cdot f(OPT) \enspace .$$

Plugging this inequality into (21), and using the fact that $\Delta_f \leq OPT$, we obtain the lower bound

$$\begin{aligned} f(S) &\geq \frac{(1-\varepsilon)}{(p+1)}\Big((1-\ell^{-1}) \cdot f(OPT) - \varepsilon\Delta_f - \rho m\Big) \\ &\geq \frac{(1-\varepsilon)}{(p+1)}\Big((1-\ell^{-1}) \cdot f(OPT) - \varepsilon f(OPT) - \rho m\Big) \\ &= \frac{(1-\varepsilon)}{(p+1)}\Big((1-\ell^{-1}-\varepsilon) \cdot f(OPT) - \rho m\Big) \enspace . \end{aligned}$$

Suppose further that f is monotone. In this case, relating $f(OPT \cup S)$ to $f(OPT)$ is more straightforward and does not require a loss of approximation. In particular, applying monotonicity directly to (21), we get

$$f(S) \geq \frac{(1-\varepsilon)}{(p+1)}\left(\frac{1}{\ell}\sum_{j=1}^{\ell} f(OPT \cup S_T^{(j)}) - \varepsilon\Delta_f - \rho m\right) \qquad \text{(Inequality (21))}$$

$$
\begin{aligned}
&\geq \frac{(1-\varepsilon)}{(p+1)}\left(\frac{1}{\ell}\sum_{j=1}^{\ell} f(OPT) - \varepsilon\Delta_f - \rho m\right) && \text{(monotonicity)}\\
&= \frac{(1-\varepsilon)}{(p+1)}\Big(f(OPT) - \varepsilon\Delta_f - \rho m\Big)\\
&\geq \frac{(1-\varepsilon)}{(p+1)}\Big(f(OPT) - \varepsilon f(OPT) - \rho m\Big) && (\Delta_f \leq f(OPT))\\
&= \frac{(1-\varepsilon)}{(p+1)}\Big((1-\varepsilon)f(OPT) - \rho m\Big)\ . && \blacksquare
\end{aligned}
$$

Appendix B. Proof of Proposition 40 (ModifiedRepeatedGreedy)

In this section, we present a proof of Proposition 40 which provides approximation guarantees for ModifiedRepeatedGreedy when the density parameter ρ is fixed. We begin by restating the proposition.

Proposition 40 *If $(\mathcal{N}, \mathcal{I})$ is a k-system, then the solution S returned by* ModifiedRepeatedGreedy *satisfies the following approximation guarantees.*

$$
f(S) \geq \begin{cases} \frac{1}{2}\rho & \text{if } E = 1\ ,\\ \left(\frac{1-\varepsilon}{k+1+\alpha(\ell-1)/2}\right)\Big((1-1/\ell-\varepsilon)f(OPT) - \rho m\Big) & \text{if } E = 0\ . \end{cases} \tag{13}
$$

Moreover, when f is monotone, these approximation guarantees improve to

$$
f(S) \geq \begin{cases} \frac{1}{2}\rho & \text{if } E = 1\ ,\\ \left(\frac{(1-\varepsilon)}{k+1+\alpha(\ell-1)/2}\right)\Big((1-\varepsilon)f(OPT) - \rho m\Big) & \text{if } E = 0\ . \end{cases} \tag{14}
$$

The main technical aspect is to prove an approximation guarantee for ModifiedGreedy when $(\mathcal{N}, \mathcal{I})$ is a k-system. Roughly speaking, this will be similar to the analysis of the vanilla greedy algorithm for k-systems (Lemma 3.2 of Gupta et al. (2010)), but we will need to account for the marginal gain thresholding and the knapsack density technique.

In order to analyze ModifiedGreedy, we now introduce the following lemma, which is a structural result about k-systems. This lemma is implicit in the proof of Lemma 3.2 of Gupta et al. (2010), but we choose to state it separately since our use of it is slightly more involved. We remark that a nearly identical construction appears in Section 3.3.

Lemma 55 *Consider in an arbitrary $X \in \mathcal{I}$ and let T be the number of iterations of* ModifiedGreedy. *There exists sets $C_1, C_2, \dots C_{T+1}$ with the following properties:*

- *The sets $C_1, C_2, \dots C_{T+1}$ form a disjoint partition of X.*
- *For every integer $1 \leq t \leq T$, $|C_t| \leq k$.*
- *For every integer $1 \leq t \leq T+1$, $C_t \subseteq \{u \mid S_{t-1} + u \in \mathcal{I}\}$.*

Proof We construct the sets $C_1, C_2, \dots C_{T+1}$ recursively, with knowledge of the algorithm's execution path. We begin by defining the last set,

$$C_{T+1} = \{u \in X \setminus S_T \mid S_T + u \in \mathcal{I}\} \enspace .$$

We construct the remaining sets recursively. For an integer $1 \leq t \leq T$, define the set B_t to be the elements in X not contained in $C_{t+1} \cup \dots \cup C_{T+1}$ which are feasible to add to solution S_{t-1}, i.e.,

$$B_t = \{u \in (X \setminus S_{t-1}) \setminus (\cup_{s=t+1}^{T+1} C_s) \mid S_{t-1} + u \in \mathcal{I}\} \enspace .$$

We define C_t to be an arbitrary subset of B_t of size $\max(|B_t|, k)$. At this point, the second and third properties in the lemma follow by construction of the sets $C_1, C_2, \dots C_{T+1}$. In the remainder of the proof, we show that the sets $C_1, C_2, \dots, C_{T+1}$ satisfy the first property; that is, they form a disjoint partition of X.

By construction, it is clear that the sets $C_1, C_2, \dots, C_{T+1}$ are disjoint and that $\cup_{t=1}^{T+1} C_t \subseteq X$. Thus, we seek to show that $X \subseteq \cup_{t=1}^{T+1} C_t$. To do this, we prove the stronger guarantee that for each integer $1 \leq t \leq T+1$,

$$|X \setminus (\cup_{s=t}^{T+1} C_s)| \leq k \cdot |S_{t-1}| \enspace .$$

Note that $X \subseteq \cup_{t=1}^{T+1} C_t$ follows as $S_0 = \varnothing$. We prove this inequality by induction, starting at $t = T+1$ as the base case and working backwards. By definition of C_{T+1}, no element of $X \setminus (C_{T+1} \cup S_T)$ can be added to S_T without violating independence, and thus, S_T is a base of $(X \setminus C_{T+1}) \cup S_T$. In contrast, $X \setminus C_{T+1}$ is an independent subset of $(X \setminus C_{T+1}) \cup S_T$ because it is a subset of the independent set X. Thus, since $(\mathcal{N}, \mathcal{I})$ is a k-system,

$$|X \setminus C_{T+1}| \leq k \cdot |S_T| \enspace ,$$

which establishes the claim for $t = T+1$. Assume that the claim holds for all integers $t+1, t+2, \dots, T+1$, and let us prove it for t. There are two cases to consider. First, suppose that $|C_t| = k$. In this case,

$$\begin{aligned} |X \setminus \cup_{s=t}^{T+1} C_s| &= |X \setminus \cup_{s=t+1}^{T+1} C_s| - |C_t| \\ &= |X \setminus \cup_{s=t+1}^{T+1} C_s| - k \\ &\leq k \cdot |S_t| - k \\ &= k \cdot |S_{t-1}| \enspace , \end{aligned}$$

where the inequality follows by induction hypothesis and the first equality holds because C_t is disjoint from all $C_{t+1}, \dots, C_{T+1}$ and $C_t \subseteq X$. The second case is that $|C_t| < k$. In this case, $C_t = B_t$ and so no element of $X \setminus (\cup_{s=t}^{T+1} C_s \cup S_{t-1})$ can be added to S_{t-1} without violating independence, and thus S_{t-1} is a base of $(X \setminus \cup_{s=t}^{T+1} C_s) \cup S_{t-1}$. This allows us to prove the claim in the same way as we did for the base case. In particular, observe that $X \setminus \cup_{s=t}^{T+1} C_s$ is an independent subset of $(X \setminus \cup_{s=t}^{T+1} C_s) \cup S_{t-1}$ because it is also a subset of the independent set X. Thus, because $(\mathcal{N}, \mathcal{I})$ is a k-system,

$$|X \setminus \cup_{s=t}^{T+1} C_s| \leq k \cdot |S_{t-1}| \enspace ,$$

which completes the proof by induction. ■

Now we are ready to prove the approximation guarantee of ModifiedGreedy.

Lemma 56 *Suppose that $\mathcal{I}$ is a k-system and that S is the set returned by* Modified-Greedy*. Then,*

$$f(S) \geq \begin{cases} \left(\frac{1-\varepsilon}{k+1}\right) \cdot [f(OPT \cup S) - \varepsilon \cdot \Delta_f - \rho m] & \text{if } E = 0 \\ \rho/2 & \text{if } E = 1 \end{cases} .$$

Proof Let T denote the number of iterations in ModifiedGreedy so that the sequence of solutions it produces is $S_0, S_1, \ldots, S_T$, where $S_T = S$ is the solution that is returned.

In the first case, suppose that $E = 1$, which is to say that Line 6 evaluates to `false` at some point during the execution of the algorithm. This happens when, at some iteration t there exists a solution S_t and a high density element u such that adding the element to this set is feasible in the independence system, but the knapsack constraint is violated. More precisely, the set $A \triangleq S_t + u$ is independent (i.e., $A \in \mathcal{I}$) but $c_r(A) > 1$ for some knapsack function $1 \leq r \leq m$. Although A itself is not feasible, we claim that $f(A) > \rho$. To this end, let us order the elements of A according to the order in which they were added to S_t, with u appearing last, i.e., $A = \{u_1, u_2, \ldots, u_k\}$ with $u_k = u$. For $1 \leq t \leq k$, define the sets $A_t = \{u_1, u_2, \ldots, u_t\}$ and $A_0 = \varnothing$. Then, we obtain the lower bound

$$f(A) = \sum_{t=1}^{k} f(u_t \mid A_{t-1}) \geq \sum_{t=1}^{k} \rho \cdot \sum_{r=1}^{m} c_r(u_t) = \rho \sum_{r=1}^{m} c_r(A) > \rho \cdot 1 = \rho ,$$

where the first inequality follows from the fact that each of the elements has high density when it is added to the solution and the second inequality follows from the fact that A violates at least one of the knapsack constraints.

The next step is to show that between S_t and $\{u\}$, at least one of these has value larger than $\rho/2$. In particular, observe that

$$\max\{f(S_t), f(\{u\})\} \geq \frac{1}{2}(f(S_t)) + f(\{u\})) \geq \frac{1}{2}(f(S_t + u) + f(\varnothing)) \geq \frac{1}{2} f(A) > \frac{\rho}{2} ,$$

where the first inequality bounds the maximum by the average, the second inequalities follows by submodularity, the third inequality follows by non-negativity, and the final inequality follows from the bound above.

Recall now that the algorithm returns the set S which has the larger objective value among S_T and $\{u^*\}$. One can note that the final solution has larger objective value than the solution at iteration t (i.e., $f(S_T) \geq f(S_t)$) because only elements with positive marginal gains are added to the solutions by the algorithm. We also note that by construction of u^*, we have that $f(\{u^*\}) \geq f(\{u\})$ because u is a feasible element. Together, these facts imply that

$$f(S) \geq \max\{f(S_T), f(u^*)\} \geq \max\{f(S_t), f(u)\} > \frac{\rho}{2} ,$$

which completes our proof for the case of $E = 1$.

In the second case, suppose that $E = 0$ so that the algorithm never considers an element which might violate the knapsack constraints. We seek to upper bound the marginal gain of adding OPT to the returned solution S_T. To this end, we begin by splitting the elements of OPT into two sets: those elements with high density with respect to S_T and those with low density. More precisely, define the set of low density elements to be

$$\mathcal{L} = \left\{ u \in OPT \mid f(u \mid S_T) < \rho \cdot \sum_{r=1}^{m} c_r(u) \right\} \enspace ,$$

and define the set of high density elements to be the remaining elements of OPT,

$$\mathcal{H} = OPT \setminus \mathcal{L} = \left\{ u \in OPT \mid f(u \mid S_T) \geq \rho \cdot \sum_{r=1}^{m} c_r(u) \right\} \enspace .$$

By submodularity of f, we may now bound the marginal gain of adding OPT to S_T in terms of adding the high and low density elements separately as

$$f(OPT \cup S_T) - f(S_T) \leq f(\mathcal{H} \mid S_T) + f(\mathcal{L} \mid S_T) \enspace . \tag{22}$$

We now upper bound the marginal gain of adding the low density elements $\mathcal{L}$ to the solution S_T. Observe that

$$\begin{aligned}
f(\mathcal{L} \mid S_T) &\leq \sum_{u \in \mathcal{L}} f(u \mid S_T) && \text{(by submodularity of } f\text{)} \\
&\leq \sum_{u \in \mathcal{L}} \rho \cdot \sum_{r=1}^{m} c_r(u) && \text{(definition of } \mathcal{L}\text{)} \\
&= \rho \cdot \sum_{r=1}^{m} c_r(\mathcal{L}) && \text{(by modularity)} \\
&\leq \rho \cdot \sum_{r=1}^{m} c_r(OPT) && (\mathcal{L} \subset OPT) \\
&\leq \rho m \enspace , && (23)
\end{aligned}$$

where the last line follows because OPT is feasible and so it satisfies the cardinality constraints $c_r(OPT) \leq 1$ for all $1 \leq r \leq m$.

We now seek to upper bound the marginal gain of adding the high density elements $\mathcal{H}$ to the solution S_T. However, this direction is more involved and it is simpler to work backwards by lower bounding the objective value of the returned solution in terms of the high density elements of OPT. Taking $X = \mathcal{H}$, define a partition of its elements into sets $C_1, \ldots, C_T, C_{T+1}$ as in the statement of Lemma 55. By non-negativity of f and a telescoping sum, we have

$$\begin{aligned}
k \cdot f(S_T) &\geq k \cdot (f(S_T) - f(S_0)) && \text{(non-negativity of } f\text{)} \\
&= k \sum_{t=1}^{T} [f(S_t) - f(S_{t-1})] && \text{(telescoping sum)} \\
&= \sum_{t=1}^{T} k \cdot f(u_t \mid S_{t-1}) && \text{(distributing)} \\
&\geq \sum_{t=1}^{T} |C_t| \cdot f(u_t \mid S_{t-1}) && \text{(by Lemma 55, } |C_t| \leq k\text{)}
\end{aligned}$$

Note that at each iteration, the chosen element u_t is a feasible high density element which has a marginal gain within a $(1-\varepsilon)$ multiplicative factor of the largest marginal gain among all such elements. We may now use the greedy selection of the element u_t and submodularity of f to establish the following lower bound:

$$\begin{aligned}
\sum_{t=1}^{T} |C_t| \cdot f(u_t \mid S_{t-1}) &\geq \sum_{t=1}^{T} |C_t| \cdot (1-\varepsilon) \max_{u \in C_t} f(u \mid S_{t-1}) && \text{(approx. greedy selection)} \\
&\geq (1-\varepsilon) \sum_{t=1}^{T} |C_t| \cdot \frac{1}{|C_t|} \sum_{u \in C_t} f(u \mid S_{t-1}) && \text{(max} \geq \text{average)} \\
&\geq (1-\varepsilon) \sum_{t=1}^{T} f(C_t \mid S_{t-1}) && \text{(submodularity of } f\text{)} \\
&\geq (1-\varepsilon) \sum_{t=1}^{T} f(C_t \mid S_T) && \text{(submodularity of } f\text{)} \\
&= (1-\varepsilon) \left[\sum_{t=1}^{T+1} f(C_t \mid S_T) - f(C_{T+1} \mid S_T) \right] && \text{(adding and subtracting term)} \\
&\geq (1-\varepsilon) \left[f(\cup_{t=1}^{T+1} C_t \mid S_T) - f(C_{T+1} \mid S_T) \right] && \text{(subadditivity of } f\text{)} \\
&= (1-\varepsilon) \left[f(\mathcal{H} \mid S_T) - f(C_{T+1} \mid S_T) \right] && \text{(Lemma 55)} \ ,
\end{aligned}$$

where subadditivity of f follows from submodularity and non-negativity.

Our final goal now is to bound the value $f(C_{T+1} \mid S_T)$, which is the marginal gain of all the elements of $\mathcal{H}$ that were not added to the final solution S_T, but could maintain feasibility in $\mathcal{I}$ if added. Consider an element $e \in C_{T+1}$. Because $u \notin S_T$ and $E = 0$, it must be the case that the marginal gain of this element to the final solution is bounded by $f(u \mid S_T) < \max(\tau, \rho \cdot \sum_{r=1}^{m} c_r(e))$. However, this element e is in $\mathcal{H}$ so it has high density with respect to the solution S_T. Thus, it must be the case that $f(u \mid S_T) < \tau$. By the termination condition, we have that $\tau < (\varepsilon/n) \cdot \Delta_f$, which implies a bound on the marginal gain $f(u \mid S_T) < (\varepsilon/n) \cdot \Delta_f$. This upper bound on the marginal gain, together with submodularity of f and the (trivial) cardinality bound $|C_{T+1}| \leq n$, yields

$$f(C_{T+1} \mid S_T) \leq \sum_{u \in C_{T+1}} f(u \mid S_T) \leq \sum_{u \in C_{T+1}} (\varepsilon/n) \cdot \Delta_f \leq \varepsilon \cdot \left(\frac{|C_{T+1}|}{n} \right) \Delta_f \leq \varepsilon \cdot \Delta_f \ .$$

Using these inequalities together yields an upper bound on the marginal gain of adding the high density elements to the returned solution,

$$f(\mathcal{H} \mid S_T) \leq \frac{k}{1-\varepsilon} \cdot f(S_T) + \varepsilon \cdot \Delta_f \ . \tag{24}$$

Thus, we may now bound the marginal gain of adding OPT to the final solution S_T by combining the above upper bounds on adding the high and low density elements. More precisely, substituting inequalities (23) and (24) into inequality (22) yields

$$f(OPT \cup S_T) - f(S_T) \leq f(\mathcal{H} \mid S_T) + f(\mathcal{L} \mid S_T) \leq \frac{k}{1-\epsilon} \cdot f(S_T) + \varepsilon \Delta_f + \rho m$$

Rearranging this inequality and using the inequality $1 \leq (1-\varepsilon)^{-1}$, we obtain

$$f(S_T) \geq \frac{1-\varepsilon}{k+1} \Big(f(OPT \cup S_T) - \varepsilon \Delta_f - \rho m \Big) \ . \qquad \blacksquare$$

We are now ready to prove the approximation guarantees of ModifiedRepeatedGreedy as stated in Proposition 40. The approximation analysis of ModifiedRepeatedGreedy is similar to the approximation analysis of RepeatedGreedy in the main paper. The main difference is that we apply Lemma 56 when considering the ModifiedGreedy subroutine rather than applying Lemma 3.2 of Gupta et al. (2010), which holds only for the vanilla greedy algorithm (which is slower than ModifiedGreedy and does not handle knapsack constraints).

Proof *(Proof of Proposition 40)*: Observe that, for every $1 \leq i \leq \ell$, we have

$$OPT \setminus \mathcal{N}_i = OPT \cap (\mathcal{N} \setminus \mathcal{N}_i) = OPT \cap \left(\cup_{j=1}^{i-1} S_i \right) = \cup_{j=1}^{i-1} (OPT \cap S_j) \tag{25}$$

where the first equality holds because $OPT \subseteq \mathcal{N}$, and the second equality follows from the removal of S_i from the ground set in each iteration of ModifiedRepeatedGreedy. Using the previous lemmata and this observation, we can obtain a lower bound on the objective value of the returned solution S in terms of the average value of $f(S_i \cup OPT)$ as

$$\begin{aligned}
\frac{1}{\ell} \sum_{i=1}^{\ell} f(S_i \cup OPT) &\leq \frac{1}{\ell} \sum_{i=1}^{\ell} f(S_i \cup (OPT \cap \mathcal{N}_i)) + \frac{1}{\ell} \sum_{i=1}^{\ell} f(OPT \setminus \mathcal{N}_i) && \text{(Lemma 30)} \\
&= \frac{1}{\ell} \sum_{i=1}^{\ell} f(S_i \cup (OPT \cap \mathcal{N}_i)) + \frac{1}{\ell} \sum_{i=1}^{\ell} f\left(\cup_{j=1}^{i-1} (OPT \cap S_j) \right) && \text{(Equality (25))} \\
&\leq \frac{1}{\ell} \sum_{i=1}^{\ell} f(S_i \cup (OPT \cap \mathcal{N}_i)) + \frac{1}{\ell} \sum_{i=1}^{\ell} \sum_{j=1}^{i-1} f(OPT \cap S_j) && \text{(submodularity)}
\end{aligned}$$

$$
\begin{aligned}
&\le \frac{1}{\ell}\sum_{i=1}^{\ell}\left[\frac{k+1}{1-\varepsilon}f(S_i)+\varepsilon\Delta_f+\rho m\right]+\frac{\alpha}{\ell}\sum_{i=1}^{\ell}\sum_{j=1}^{i-1}f(S'_j) && \text{(Lemmas 56 and 29)}\\
&\le \frac{1}{\ell}\sum_{i=1}^{\ell}\left[\frac{k+1}{1-\varepsilon}f(S)+\varepsilon\Delta_f+\rho m\right]+\frac{\alpha}{\ell}\sum_{i=1}^{\ell}\sum_{j=1}^{i-1}f(S) && \text{(definition of } S\text{)}\\
&= \frac{k+1}{1-\varepsilon}f(S)+\varepsilon\Delta_f+\rho m+\frac{\alpha(\ell-1)}{2}f(S)\\
&\le (1-\varepsilon)^{-1}\left(k+1+\alpha(\ell-1)/2\right)f(S)+\varepsilon f(OPT)+\rho m\ .
\end{aligned}
$$

Rearranging this inequality yields the following lower bound on the value of the returned solution:

$$
f(S)\ge\left(\frac{1-\varepsilon}{k+1+\alpha(\ell-1)/2}\right)\left[\frac{1}{\ell}\sum_{i=1}^{\ell}f(S_i\cup OPT)-\varepsilon f(OPT)-\rho m\right]\ . \tag{26}
$$

In order to remove the dependence of the right hand side on the solutions S_i, we again use Lemma 7 [Lemma 2.2 of Buchbinder et al. (2014)]. In particular, consider a set $\bar{S}$ chosen uniformly at random from the ℓ constructed solutions $S_1, S_2, \dots S_\ell$. Because the solutions are disjoint by construction, an element can belong to $\bar{S}$ with probability at most ℓ^{-1}. Hence, applying Lemma 7 to the submodular function $g(S) = f(OPT \cup S)$, we get

$$
\frac{1}{\ell}\sum_{i=1}^{\ell}f(S_i\cup OPT)=\mathbb{E}[f(OPT\cup\bar{S})]=\mathbb{E}[g(\bar{S})]\ge(1-\ell^{-1})\cdot g(\varnothing)=(1-\ell^{-1})\cdot f(OPT)\ . \tag{27}
$$

Substituting (27) into the lower bound of (26) yields the desired result.

When f is monotone submodular, we may obtain an improved approximation ratio by applying monotonicity directly to the lower bound (26). In particular, applying monotonicity yields

$$
\frac{1}{\ell}\sum_{i=1}^{\ell}f(S_i\cup OPT)\ge\frac{1}{\ell}\sum_{i=1}^{\ell}f(OPT)=f(OPT)\ ,
$$

which yields the desired approximation in the monotone setting. ■

References

Ayya Alieva, Aiden Aceves, Jialin Song, Stephen Mayo, Yisong Yue, and Yuxin Chen. Learning to make decisions via submodular regularization. In *International Conference on Learning Representations*, 2020.

Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *SODA*, pages 1497–1514, 2014.

Andrew An Bian, Joachim M Buhmann, Andreas Krause, and Sebastian Tschiatschek. Guarantees for greedy maximization of non-submodular functions with applications. In *International conference on machine learning*. PMLR, 2017.

Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, USA, 1998. ISBN 0521563925.

N. Buchbinder, M. Feldman, N.S. Joseph, and R. Schwartz. A tight linear time (1/2)-approximatoin for unconstrained submodular maximization. *SIAM Journal on Computing*, 44:1384–1402, 2015.

Niv Buchbinder and Moran Feldman. Submodular functions maximization problems. In Teofilo F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics, Second Edition, Volume 1: Methologies and Traditional Applications*, pages 753–788. Chapman and Hall/CRC, 2018a.

Niv Buchbinder and Moran Feldman. Deterministic Algorithms for Submodular Maximization Problems. *ACM Trans. Algorithms*, 14(3):32:1–32:20, 2018b.

Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular Maximization with Cardinality Constraints. In *SODA*, pages 1433–1452, 2014.

Niv Buchbinder, Moran Feldman, and Roy Schwartz. Comparing apples and oranges: Query trade-off in submodular maximization. *Math. Oper. Res.*, 42(2):308–329, 2017.

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

Yuxin Chen, Shervin Javdani, Amin Karbasi, J Bagnell, Siddhartha Srinivasa, and Andreas Krause. Submodular surrogates for value of information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015.

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

Ethan R. Elenberg, Alexandros G. Dimakis, Moran Feldman, and Amin Karbasi. Streaming weak submodularity: Interpreting neural networks on the fly. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

Alina Ene and Huy L. Nguyen. A nearly-linear time algorithm for submodular maximization with a knapsack constraint. In *ICALP*, pages 53:1–53:12, 2019a.

Alina Ene and Huy L. Nguyen. Towards nearly-linear time algorithms for submodular maximization with a matroid constraint. In *ICALP*, pages 54:1–54:14, 2019b. doi: 10.4230/LIPIcs.ICALP.2019.54. URL `https://doi.org/10.4230/LIPIcs.ICALP.2019.54`.

Uriel Feige, Vahab S. Vahab S. Mirrokni, and Jan Vondrak. Maximizing non-monotone submodular functions. In *FOCS*, 2007.

Moran Feldman, Joseph Naor, Roy Schwartz, and Justin Ward. Improved approximations for k-exchange systems - (extended abstract). In *ESA*, pages 784–798, 2011.

Moran Feldman, Christopher Harshaw, and Amin Karbasi. Greed is good: Near-optimal submodular maximization via greedy optimization. In *COLT*, pages 758–784, 2017.

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

Shayan Oveis Gharan and Jan Vondrak. Submodular maxzimiation by simulated annealing. In *SODA*, 2011.

Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Near-optimal map inference for determinantal point processes. In *Advances in Neural Information Processing Systems 25*, pages 2735–2743, 2012.

Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained Non-monotone Submodular Maximization: Offline and Secretary Algorithms. In *WINE*, pages 246–257, 2010.

Ran Haba, Ehsan Kazemi, Moran Feldman, and Amin Karbasi. Streaming Submodular Maximization under a k-Set System Constraint. In *ICML*, 2020.

Kai Han, Shuang Cui, Tianshuai Zhu, Jing Tang, Benwei Wu, and He Huang. The power of randomization: Efficient and effective algorithms for constrained submodular maximization, 2021.

Chris Harshaw, Moran Feldman, Justin Ward, and Amin Karbasi. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2019.

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

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *international conference on Knowledge discovery and data mining (KDD)*, pages 137–146, 2003.

A. Krause and C. Guestrin. Near-optimal Nonmyopic Value of Information in Graphical Models. In *Uncertainty in Artificial Intelligence (UAI)*, pages 324–331, 2005.

Alan Kuhnle. Interlaced Greedy Algorithm for Maximization of Submodular Functions in Nearly Linear Time. In *NeurIPS*, 2019.

Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM J. Discrete Math.*, 23(4):2053–2078, 2010a.

Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular Maximization over Multiple Matroids via Generalized Exchange Properties. *Math. Oper. Res.*, 35(4):795–806, 2010b.

Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics, 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 *European Symposium on Algorithms (ESA)*, pages 528–539, 2006.

Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, pages 234–243, 1978.

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

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier Than Lazy Greedy. In *AAAI Conference on Artificial Intelligence*, pages 1812–1818, 2015.

Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast Constrained Submodular Maximization: Personalized Data Summarization. In *ICML*, pages 1358–1367, 2016.

Mehraveh Salehi, Amin Karbasi, Dustin Scheinost, and R. Todd Constable. A submodular approach to create individualized parcellations of the human brain. In *Medical Image Computing and Computer Assisted Intervention, MICCAI 2017*, pages 478–485. Springer International Publishing, 2017.

Adish Singla, Ilija Bogunovic, Gabor Bartok, Amin Karbasi, and Andreas Krause. Near-optimally teaching the crowd to classify. In *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2014.

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

Ehsan Tohidi, Rouhollah Amiri, Mario Coutino, David Gesbert, Geert Leus, and Amin Karbasi. Submodularity in action: From machine learning to signal processing applications, 2020.

Jan Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM J. Comput.*, 42(1):265–304, 2013. doi: 10.1137/110832318. URL `https://doi.org/10.1137/110832318`.

Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 222–227, 1977. doi: 10.1109/SFCS.1977.24.