Randomized Algorithms for Submodular Function Maximization with a k-System Constraint

Shuang Cui [1] **Kai Han** [*1] **Tianshuai Zhu** [1] **Jing Tang** [2] **Benwei Wu** [1] **He Huang** [3]

Abstract

Submodular optimization has numerous applications such as crowdsourcing and viral marketing. In this paper, we study the problem of non-negative submodular function maximization subject to a k-system constraint, which generalizes many other important constraints in submodular optimization such as cardinality constraint, matroid constraint, and k-extendible system constraint. The existing approaches for this problem are all based on deterministic algorithmic frameworks, and the best approximation ratio achieved by these algorithms (for a general submodular function) is $k + 2\sqrt{k+2} + 3$. We propose a randomized algorithm with an improved approximation ratio of $(1 + \sqrt{k})^2$, while achieving nearly-linear time complexity significantly lower than that of the state-of-the-art algorithm. We also show that our algorithm can be further generalized to address a stochastic case where the elements can be adaptively selected, and propose an approximation ratio of $(1 + \sqrt{k+1})^2$ for the adaptive optimization case. The empirical performance of our algorithms is extensively evaluated in several applications related to data mining and social computing, and the experimental results demonstrate the superiorities of our algorithms in terms of both utility and efficiency.

1. Introduction

Submodular optimization is an active research area in machine learning due to its wide applications such as crowdsourcing (Singla et al., 2016; Han et al., 2018a), clustering (Gomes & Krause, 2010; Han et al., 2019), viral marketing (Kempe et al., 2003; Han et al., 2018b), and data summarization (Badanidiyuru et al., 2014; Iyer & Bilmes, 2013). A lot of the existing studies in this area aim to maximize a submodular function subject to a specific constraint, and it is well known that these problems are generally NP-hard. Therefore, extensive approximation algorithms have been proposed, with the goal of achieving improved approximation ratios or lower time complexity.

Formally, given a ground set $\mathcal{N}$ with $|\mathcal{N}| = n$, a constrained submodular maximization problem can be written as:

$$\max\{f(S) : S \in \mathcal{I}\} \tag{1}$$

where $f : 2^{\mathcal{N}} \mapsto \mathbb{R}_{\geq 0}$ is a submodular function satisfying $\forall X, Y \subseteq \mathcal{N} : f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$, and $\mathcal{I} \subseteq 2^{\mathcal{N}}$ is the set of all feasible solutions. For example, if $\mathcal{I} = \{X : X \subseteq \mathcal{N} \wedge |X| \leq d\}$ for a given $d \in \mathbb{N}$, then $S \in \mathcal{I}$ represents a cardinality constraint. We also call $f(\cdot)$ "monotone" if it satisfies $\forall X \subseteq Y \subseteq \mathcal{N} : f(X) \leq f(Y)$, otherwise $f(\cdot)$ is called "non-monotone".

Although some application problems only have simple constraints like a cardinality constraint, many others have to be cast as submodular maximization problems with more complex "independence system" constraints such as matroid, k-matchoid, and k-system constraint. Among these constraints, the k-system constraint is the most general one, and a strict inclusion hierarchy of them is: cardinality $\subset$ matroid $\subset$ intersection of k matroids $\subset$ k-matchoid $\subset$ k-extendible $\subset$ k-system (Mestre, 2006). Due to the generality of k-system constraint, it can be used to model a lot of constraints in various applications, such as graph matchings, spanning trees and scheduling (Feldman et al., 2020; Mirzasoleiman et al., 2016).

It is recognized that submodular maximization with a k-system constraint is one of the most fundamental problems in submodular optimization (Calinescu et al., 2011; Feldman et al., 2017; 2020), so a lot of efforts have been devoted to it since the 1970s, and the state-of-the-art approximation ratios are $k+1$ (Fisher et al., 1978) and $k+2\sqrt{k+2}+3$ (Feldman et al., 2020) for monotone $f(\cdot)$ and non-monotone $f(\cdot)$, respectively. Feldman et al. (2020) also showed that, by weak-

---

Co-first authors: Shuang Cui and Kai Han. The work of Shuang Cui, Tianshuai Zhu and Benwei Wu is finished under the guidance of their supervisor: Kai Han. [1]School of Computer Science and Technology / SuZhou Research Institute, University of Science and Technology of China; [2]Data Science and Analytics Thrust, The Hong Kong University of Science and Technology; [3]School of Computer Science and Technology, Soochow University. Correspondence to: Kai Han <hankai@ustc.edu.cn>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

ening their approximation ratio by a factor of $(1-2\epsilon)^{-2}$, their algorithm can be implemented under time complexity of $\mathcal{O}(\frac{kn}{\epsilon}\log(\frac{n}{\epsilon}))$. Surprisingly, all the existing algorithms for this problem are intrinsically deterministic. Therefore, it is an interesting open problem whether the "power of randomization" can be leveraged to achieve better approximation ratios or better efficiency, as randomized algorithms are known to outperform the deterministic ones in many other problems.

It is noted that the utility function $f(\cdot)$ is assumed to be deterministic in Problem (1). However, in many applications such as viral marketing and sensor placement, the utility function could be stochastic and is only submodular in a probabilistic sense. To address these settings, Golovin & Krause (2011a) introduced the concept of adaptive submodular maximization, where each element $u \in \mathcal{N}$ is assumed to have a random state and the goal is to find an optimal *adaptive policy* that can select a new element based on observing the realized states of already selected elements. Based on this concept, they also investigated the adaptive submodular maximization problem under a k-system constraint and provide an approximation ratio of $k+1$ (Golovin & Krause, 2011b), but this ratio only holds when the utility function is *adaptive monotone* (a property similar to the monotonicity property under the non-adaptive case). However, it still remains as an open problem whether provable approximation ratios can be achieved for this problem when the considered utility function is more general (i.e., not necessarily adaptive monotone).

In this paper, we provide confirmative answers to all the open problems mentioned above, by presenting novel randomized algorithms for the problem of (not necessarily monotone) submodular function maximization with a k-system constraint. Our algorithms advance the state-of-the-art under both the non-adaptive setting and the adaptive setting. More specifically, our contributions include:

- Under the non-adaptive setting, we present a randomized algorithm dubbed RANDOMMULTIGREEDY that achieves an approximation ratio of $(1+\sqrt{k})^2$ under time complexity of $\mathcal{O}(nr)$, where r is the rank of the considered k-system. We also show that RANDOMMULTIGREEDY can be accelerated to achieve an approximation ratio of $(1+\epsilon)(1+\sqrt{k})^2$ under nearly-linear time complexity of $\mathcal{O}(\frac{n}{\epsilon}\log\frac{r}{\epsilon})$. Therefore, our algorithm outperforms the state-of-the-art algorithm in (Feldman et al., 2020) in terms of both approximation ratio and time complexity. Furthermore, we show that RANDOMMULTIGREEDY can also be implemented as a deterministic algorithm with better performance bounds than the existing algorithms.
- Under the adaptive setting, we provide a randomized policy dubbed ADAPTRANDOMGREEDY that achieves an approximation ratio of $(1+\sqrt{k+1})^2$ when the utility function is not necessarily adaptive monotone. To the best of our knowledge, ADAPTRANDOMGREEDY is the first adaptive algorithm to achieve a provable performance ratio under this case.
- We test the empirical performance of the proposed algoirthms in several applications including movie recommendation, image summarization and social advertising with multiple products. The extensive experimental results demonstrate that, RANDOMMULTIGREEDY achieves approximately the same performance as the best existing algorithm in terms of utility, while its performance on efficiency is much better than that of the fastest known algorithm; besides, ADAPTRANDOMGREEDY can achieve better utility than the non-adaptive algorithms by leveraging adaptivity.

For the fluency of description, the proofs of all our lemmas/theorems are deferred to the supplementary file.

2. Related Work

There are extensive studies on submodular maximization such as (Chekuri & Quanrud, 2019), (Balkanski et al., 2019), (Lee et al., 2010), (Han et al., 2021) and (Kuhnle, 2019). For example, Kuhnle (2019) addressed a simple cardinality constraint using a nice "interlaced greedy" algorithm, where two candidate solutions are considered in a compulsory round-robin way; but it is unclear whether this algorithm can handle more complex constraints. In the sequel, we only review the studies most closely related to our work.

Non-Adaptive Algorithms: We first review the existing algorithms for non-monotone submodular maximization subject to a k-system constraint under the non-adaptive setting. The seminal work of (Gupta et al., 2010) proposed a REPEATEDGREEDY algorithm described as follows. At first, a series of candidate solutions $S_1, S_2, \cdots, S_\ell$ are sequentially found, where the elements of S_j are greedily selected from $\mathcal{N} \setminus (\cup_{1\le i\le j-1} S_i)$ for all $j \in [\ell]$. After that, an Unconstrained Submodular Maximization (USM) algorithm (e.g., (Buchbinder et al., 2015)) is called to find $S'_j \subseteq S_j$ for all $j \in [\ell]$. Finally, the set in $\{S_j, S'_j : j \in [\ell]\}$ with the maximum utility is returned. Note that the USM algorithm is only used as a "black-box" oracle and can be any deterministic/randomized algorithm, so this algorithmic framework is intrinsically deterministic. Gupta et al. (2010) showed that, by setting $\ell = k+1$, REPEATEDGREEDY can achieve an approximation ratio of $3k+6+3k^{-1}$ under $\mathcal{O}(nrk)$ time complexity. However, through a more careful analysis, Mirzasoleiman et al. (2016) proved that REPEATEDGREEDY actually has an approximation ratio of $2k+3+k^{-1}$. Subsequently, Feldman et al. (2017) further revealed that REPEATEDGREEDY can achieve an approximation ratio of $k+2\sqrt{k}+3+\frac{6}{\sqrt{k}}$ under $\mathcal{O}(nr\sqrt{k})$ time

Table 1. Approximation for submodular function maximization with a k-system constraint

Algorithms	Source	Ratio	Time Complexity	Adaptive?
REPEATEDGREEDY	(Gupta et al., 2010)	$3k + 6 + 3k^{-1}$	$\mathcal{O}(nrk)$	×
REPEATEDGREEDY	(Mirzasoleiman et al., 2016)	$2k + 3 + k^{-1}$	$\mathcal{O}(nrk)$	×
TWINGREEDYFAST	(Han et al., 2020)	$2k + 2 + \epsilon$	$\mathcal{O}(\frac{n}{\epsilon}\log(\frac{r}{\epsilon}))$	×
REPEATEDGREEDY	(Feldman et al., 2017)	$k + 2\sqrt{k} + 3 + \frac{6}{\sqrt{k}}$	$\mathcal{O}(nr\sqrt{k})$	×
FASTSGS	(Feldman et al., 2020)	$(1-2\epsilon)^{-2}(k + 2\sqrt{k+2} + 3)$	$\mathcal{O}(\frac{kn}{\epsilon}\log(\frac{n}{\epsilon}))$	×
RANDOMMULTIGREEDY	this work	$(1+\epsilon)(k + 2\sqrt{k} + 1)$	$\mathcal{O}(\frac{n}{\epsilon}\log(\frac{r}{\epsilon}))$	×
ADAPTRANDOMGREEDY	this work	$k + 2\sqrt{k+1} + 2$	$\mathcal{O}(nr)$	√

complexity by setting $\ell = \lceil\sqrt{k}\rceil$.

Recently, Han et al. (2020) proposed a different "simultaneous greedy search" framework, where two disjoint candidate solutions S_1 and S_2 are maintained simultaneously, and the algorithm always greedily selects a pair (e, S_i) such that adding e into S_i brings the maximum marginal gain. By incorporating a "thresholding" method akin to that in (Badanidiyuru & Vondrák, 2014), Han et al. (2020) proved that their algorithm achieves $(2k + 2 + \epsilon)$-approximation under $\mathcal{O}(\frac{n}{\epsilon}\log\frac{r}{\epsilon})$ time complexity. Feldman et al. (2020) also proposed an elegant algorithm where $\lfloor 2 + \sqrt{k+2}\rfloor$ disjoint candidate solutions are maintained. By leveraging a thresholding method similar to (Badanidiyuru & Vondrák, 2014; Han et al., 2020), Feldman et al. (2020) proved that their algorithm can achieve $(1-2\epsilon)^{-2}(k + 2\sqrt{k+2} + 3)$-approximation under $\mathcal{O}(\frac{kn}{\epsilon}\log\frac{n}{\epsilon})$ time complexity. On the hardness side, Feldman et al. (2017) proved that no algorithm making polynomially many queries to the value and independence oracles can achieve an approximation better than $k + 0.5 - \epsilon$.

For clarity, we list the performance bounds of the closely related algorithms mentioned above in Table 1 [1]. Finally, it is noted that some related studies also considered the k-system constraint together with multiple knapsack constraints or under the streaming setting (e.g., (Haba et al., 2020; Mirzasoleiman et al., 2018; Badanidiyuru et al., 2020)).

Adaptive Algorithms: We then provide a brief review on the related studies on adaptive submodular maximization. Golovin & Krause (2011a) initiated the study on adaptive submodular maximization and also provided several algorithms under cardinality or knapsack constraints. They also studied the more general k-system constraint in (Golovin & Krause, 2011b) and provided a $(k+1)$-approximation. Recently, Esfandiari et al. (2021) proposed adaptive submodular maximization algorithms with fewer adaptive rounds of observation. There also exist many other studies on adaptive optimization under various settings/constraints, such as (Cuong & Xu, 2016; Mitrovic et al., 2019; Parthasarathy, 2020; Fujii & Sakaue, 2019; Badanidiyuru et al., 2016). However, all these studies assumed that the target function is monotone or adaptive monotone. For non-monotone objective functions, Amanatidis et al. (2020) and Gotovos et al. (2015) have proposed adaptive submodular maximization algorithms with provable performance ratios, but only under simple cardinality and knapsack constraints.

[1] For simplicity, we only list the performance bounds of the accelerated versions of Feldman et al. (2020)'s algorithm and RANDOMMULTIGREEDY in Table 1. Without acceleration, Feldman et al. (2020) can achieve an approximation ratio of $(1 + \sqrt{k+2})^2$ under $\mathcal{O}(knr)$ time complexity, while RANDOMMULTIGREEDY achieves an approximation ratio of $(1+\sqrt{k})^2$ under $\mathcal{O}(nr)$ time complexity.

3. Preliminaries and Notations

It is well known that all the structures including matroid, k-matchoid, k-extendible system and k-set system are set systems obeying the "down-closed" property captured by the concept of an *independence system*:

Definition 1 (independence system)**.** *Given a finite ground set $\mathcal{N}$ and a collection of sets $\mathcal{I} \subseteq 2^{\mathcal{N}}$, the pair $(\mathcal{N}, \mathcal{I})$ is called an independence system if it satisfies: (1) $\emptyset \in \mathcal{I}$; (2) if $X \subseteq Y \subseteq \mathcal{N}$ and $Y \in \mathcal{I}$, then $X \in \mathcal{I}$.*

Given an independence system $(\mathcal{N}, \mathcal{I})$ and any two sets $X \subseteq Y \subseteq \mathcal{N}$, X is called a *base* of Y if $X \in \mathcal{I}$ and $X \cup \{u\} \notin \mathcal{I}$ for all $u \in Y \setminus X$. We also use r to denote the *rank* of $(\mathcal{N}, \mathcal{I})$, i.e., $r = \max\{|X| : X \in \mathcal{I}\}$. A k-system is a special independence system defined as:

Definition 2 (k-system)**.** *An independence system $(\mathcal{N}, \mathcal{I})$ is called a k-system ($k \geq 1$) if $|X_1| \leq k|X_2|$ holds for any two bases X_1 and X_2 of any set $Y \subseteq \mathcal{N}$.*

Non-adaptive setting: Under the non-adaptive setting, our problem is to identify an optimal solution O to Problem (1) given a k-system $(\mathcal{N}, \mathcal{I})$ and a (not necessarily monotone) submodular function $f(\cdot)$. For convenience, we use $f(X \mid Y)$ as a shorthand for $f(X \cup Y) - f(Y)$ for all $X, Y \subseteq \mathcal{N}$. It is well known that any non-negative submodular function $f(\cdot)$ satisfies the "diminishing returns" property: $\forall X \subseteq Y \subseteq \mathcal{N}, x \in \mathcal{N} \setminus Y : f(x \mid Y) \leq f(x \mid X)$. Following the existing studies, we assume that the values of $f(S)$ and $\mathbf{1}_{\mathcal{I}}(S)$ can be got by calling *oracle queries*, and use the number of oracle queries to measure time complexity.

Adaptive setting: Under the adaptive setting, each element $u \in \mathcal{N}$ is associated with an initially unknown state $\Phi(u) \in Z$, where Z is the set of all possible states. A *realization* is any function $\phi\colon \mathcal{N} \mapsto Z$ mapping every element $u \in \mathcal{N}$ to a state $z \in Z$. Therefore, Φ is the true realization and we follow (Golovin & Krause, 2011a) to assume that $\Pr[\Phi = \phi]$ is known for any possible realization ϕ. In adaptive optimization problems, an *adaptive policy* π is allowed to sequentially select elements in $\mathcal{N}$, and the true state $\Phi(u)$ of any $u \in \mathcal{N}$ can only be observed after u is selected. In such a case, the utility of π depends on not only the selected elements but also their states, so we re-define the utility function as $f\colon 2^{\mathcal{N}} \times Z^{\mathcal{N}} \mapsto \mathbb{R}_{\geq 0}$. Let $\mathcal{N}(\pi, \phi)$ denote the set of elements selected by π under any realization ϕ, the expected utility of policy π is defined as

$$f_{\mathrm{avg}}(\pi) := \mathbb{E}[f(\mathcal{N}(\pi, \Phi), \Phi)],$$

where the expectation is taken over both the randomness of Φ and the internal randomness (if any) of π.

Given any $M \subseteq \mathcal{N}$, a mapping $\psi\colon M \mapsto Z$ is called a *partial realization*, and $\mathrm{dom}(\psi) = M$ is called the *domain* of ψ. Therefore, a partial realization is ψ is also a realization when $\mathrm{dom}(\psi) = \mathcal{N}$. Intuitively, a partial realization can be used to record the already selected elements and the observed states of them during the execution of an adaptive policy. We also abuse the notations a little by regarding ψ as the set $\{(u, \psi(u))\colon u \in \mathrm{dom}(\psi)\}$. Given two partial realizations ψ and ψ', we say ψ is a *subrealization* of ψ' (denoted by $\psi' \sim \psi$) if $\psi \subseteq \psi'$. With these definitions, we follow Golovin & Krause (2011a) to define the concept of adaptive sumodularity:

Definition 3. *Given a partial realization ψ and an element u, the expected marginal gain of u conditioned on ψ is defined as $\Delta(u \mid \psi) = \mathbb{E}[f(\mathrm{dom}(\psi) \cup \{u\}, \Phi) - f(\mathrm{dom}(\psi), \Phi) \mid \Phi \sim \psi]$. A function $f\colon 2^{\mathcal{N}} \times Z^{\mathcal{N}} \mapsto \mathbb{R}_{\geq 0}$ is called adaptive submodular if it satisfies $\forall \psi \subseteq \psi', u \in \mathcal{N} \setminus \mathrm{dom}(\psi')\colon \Delta(u \mid \psi) \geq \Delta(u \mid \psi')$.*

The utility function $f(\cdot)$ is also called *adaptive monotone* if $\Delta(u \mid \psi) \geq 0$ for any $u \in \mathcal{N}$ and any partial realization ψ satisfying $\Pr[\Phi \sim \psi] > 0$. However, in this paper we consider the case that $f(\cdot)$ is not necessarily adaptive monotone. In such a case, all the the current studies (e.g., (Amanatidis et al., 2020; Gotovos et al., 2015)) assume that $f(\cdot)$ is also *pointwise submodular*, whose definition is given below:

Definition 4. *A function $f\colon 2^{\mathcal{N}} \times Z^{\mathcal{N}} \mapsto \mathbb{R}_{\geq 0}$ is pointwise submodular if $f(\cdot, \phi)$ is submodular for any realization ϕ satisfying $\Pr[\Phi = \phi] > 0$.*

Given a k-system $(\mathcal{N}, \mathcal{I})$ and an adaptive and pointwise submodular function $f\colon 2^{\mathcal{N}} \times Z^{\mathcal{N}} \mapsto \mathbb{R}_{\geq 0}$, our problem is to identify an optimal policy π_{opt} to the following adaptive optimization problem:

$$\max\{f_{\mathrm{avg}}(\pi)\colon \mathcal{N}(\pi, \phi) \in \mathcal{I} \text{ for all realization } \phi\}. \quad (2)$$

Algorithm 1 RANDOMMULTIGREEDY(ℓ, p)

Initialize: $\forall i \in [\ell]:\ S_i \leftarrow \emptyset;\ \ t \leftarrow 1$

repeat
  for $i = 1$ **to** ℓ **do**
    $A_i \leftarrow \{u \in \mathcal{N} : S_i \cup \{u\} \in \mathcal{I}\}$
    $v_i \leftarrow \arg\max_{u \in A_i} f(u \mid S_i)$
  end for
  if $\cup_{i \in [\ell]} A_i \neq \emptyset$ **then**
    $i_t \leftarrow \arg\max_{i \in [\ell]: A_i \neq \emptyset} f(v_i \mid S_i)$; $u_t \leftarrow v_{i_t}$
    if $f(u_t \mid S_{i_t}) > 0$ **then**
      with probability p **do** $S_{i_t} \leftarrow S_{i_t} \cup \{u_t\}$
      $\mathcal{N} \leftarrow \mathcal{N} \setminus \{u_t\}; t \leftarrow t + 1$
    else
      break;
    end if
  end if
until $\bigcup_{i \in [\ell]} A_i = \emptyset \vee \mathcal{N} = \emptyset$
$S^* \leftarrow \arg\max_{S \in \{S_1, S_2, \cdots, S_\ell\}} f(S); T \leftarrow t - 1$
Output: S^*, T

4. Non-Adaptive Algorithm

In this section, we propose an algorithm dubbed RANDOMMULTIGREEDY, as shown by Algorithm 1. RANDOMMULTIGREEDY iterates for T steps to construct ℓ candidate solutions $S_1, S_2, \cdots, S_\ell$. At each step t, it greedily finds a pair $(u_t, i_t) \in \mathcal{N} \times [\ell]$ such that $S_{i_t} \cup \{u_t\} \in \mathcal{I}$ and $f(u_t \mid S_{i_t})$ is maximized. If $f(u_t \mid S_{i_t}) > 0$, then Algorithm 1 adds u_t into S_{i_t} with probability p and discard u_t with probability $1 - p$. After that, u_t is removed from $\mathcal{N}$. The iterations stop immediately when the pair (u_t, i_t) cannot be found or $f(u_t \mid S_{i_t}) \leq 0$.

For convenience, we introduce the following notations. Let $U = \{u_1, \cdots, u_T\}$ denote the set of all elements that have been considered to be added into $\cup_{i \in [\ell]} S_i$. For any $u \in \mathcal{N}$, let $S_i^<(u)$ denote the set of elements already in S_i at the moment that u is considered by the algorithm, and let $S_i^<(u) = S_i$ if u is never considered by the algorithm.

Although the design of RANDOMMULTIGREEDY is quite simple, its performance analysis is highly non-trivial due to the complex relationships between the elements in $S_1, \cdots, S_\ell$ and the randomness of the algorithm. To address these challenges, we first classify the elements in O as follows:

Definition 5. *Let D_j denote the set of elements in U that have been considered to be added into S_j but are discarded* *due to Line 9. For any $i, j \in [\ell]$ satisfying $i \neq j$, we define:*

$$O_j^{i+} = \left\{u \in O \cap S_j : S_i^{<}(u) \cup \{u\} \in \mathcal{I}\right\};$$
$$O_j^{i-} = \left\{u \in O \cap S_j : S_i^{<}(u) \cup \{u\} \notin \mathcal{I}\right\};$$
$$\widehat{O}_j^{i+} = \left\{u \in O \cap D_j : S_i^{<}(u) \cup \{u\} \in \mathcal{I}\right\};$$
$$\widehat{O}_j^{i-} = \left\{u \in O \cap D_j : S_i^{<}(u) \cup \{u\} \notin \mathcal{I}\right\};$$
$$O_i^{-} = \{u \in O \setminus U : S_i \cup \{u\} \notin \mathcal{I} \land f(u \mid S_i) > 0\};$$

Note that both O_j^{i+} and O_j^{i-} are disjoint subsets of $O \cap S_j$. Intuitively, each element $u \in O_j^{i+}$ (resp. $u \in O_j^{i-}$) can (resp. cannot) be added into S_i without violating the feasibility of $\mathcal{I}$ at the moment that u is added into S_j. The sets $\widehat{O}_j^{i+}$, $\widehat{O}_j^{i-}$ are also defined similarly for the elements in $O \cap D_j$. Based on Definition 5, it can be seen that, when Algoirthm 1 terminates, all the elements in O_i^-, O_j^{i-} and $\widehat{O}_j^{i-}$ ($\forall j \neq i$) cannot be added into S_i due to the violation of $\mathcal{I}$. Note that these elements together with the elements in $O \cap S_i$ all belong to O. So we can map them to the elements in S_i using a method similar to that in (Calinescu et al., 2011; Han et al., 2020) based on the definition of k-system, as shown by Lemma 1:

Lemma 1. *For each $i \in [\ell]$, let $Q_i = \cup_{j \in [\ell]\setminus\{i\}}(O_j^{i-} \cup \widehat{O}_j^{i-}) \cup (O \cap S_i) \cup O_i^-$. There exists a mapping $\sigma_i : Q_i \mapsto S_i$ satisfying: (1) The element $\sigma_i(u)$ can be added into $S_i^{<}(\sigma_i(u))$ without violating the feasibility of $\mathcal{I}$ for all $u \in Q_i$; (2) The number of elements in Q_i mapped to the same element in S_i by $\sigma_i(\cdot)$ is no more than k; and (3) we have $\forall u \in O \cap S_i : \sigma_i(u) = u$.*

The purpose for creating the mapping in Lemma 1 is to bound the value of $f(u \mid S_i)$ for all $u \in Q_i$. For example, given any element $u \in O_j^{i-}$, u can be mapped to an element $v = \sigma_i(u)$ satisfying $S_i^{<}(v) \cup \{u\} \in \mathcal{I}$, which implies that the value of $f(u \mid S_i)$ is no more than $f(v \mid S_i^{<}(v))$, because otherwise u should have been added into S_i instead of v according to the greedy rule of Algorithm 1. Based on this intuition, a more careful analysis reveals that:

Lemma 2. *For any $u_t \in U$ where $t \in [T]$, we define $\delta(u_t) = \sum_{j=1}^{\ell} \mathbf{1}\{i_t = j\} \cdot f(u \mid S_j^{<}(u))$. Given any $i, j \in [\ell]$ satisfying $i \neq j$, we have*

$$\forall u \in O_j^{i+} \cup \widehat{O}_j^{i+} : f(u \mid S_i) \leq \delta(u); \tag{3}$$
$$\forall u \in Q_i : f(u \mid S_i) \leq \delta(\pi_i(u)); \tag{4}$$

where Q_i is defined in Lemma 1.

Using Lemma 2, we can prove Lemma 3, which provides an upper bound of $\sum_{i \in [\ell]} f(O \mid S_i)$:

Lemma 3. *For any $u \in \mathcal{N}$, define $X_u = 1$ if $u \in (U \cap O) \setminus \cup_{i=1}^{\ell} S_i$, otherwise define $X_u = 0$. Given any integer $\ell \geq 2$, we have*

$$\sum_{i \in [\ell]} f(O \mid S_i) \leq \ell(k + \ell - 2) f(S^*) + \ell \sum_{u \in \mathcal{N}} X_u \cdot \delta(u) \tag{5}$$

The proof idea of Lemma 3 is roughly explained as follows. As the elements in $O \setminus S_i$ can be classified using the sets defined in Definition 5, we can leverage Lemma 1 and Lemma 2 to bound $f(u \mid S_i)$ for all $u \in O \setminus S_i$ using the marginal gains of the elements in $\cup_{i=1}^{\ell} S_i$. These marginal gains are further grouped in a subtle way such that their summation can be bounded by the RHS of Eqn. (5).

Note that Eqn. (5) holds for *every* random output of Algorithm 1. So the inequality still holds after taking expectation. Furthermore, we introduce Lemma 4 to bound the expectations of the LHS and RHS of Eqn. (5). The proof of Lemma 4 leverages the property that each element in $\mathcal{N}$ is only accepted with probability of at most p.

Lemma 4. *For any $p \in (0, 1]$, we have*

$$\mathbb{E}\Big[\sum\nolimits_{i \in [\ell]} f(O \cup S_i)\Big] \geq (\ell - p) f(O) \tag{6}$$
$$\mathbb{E}\Big[\sum_{u \in \mathcal{N}} X_u \cdot \delta(u)\Big] \leq \frac{1-p}{p} \mathbb{E}\Big[\sum_{i \in [\ell]} f(S_i)\Big] \tag{7}$$

By combining Lemma 3, Lemma 4 and the fact that $\forall i \in [\ell] : f(S_i) \leq f(S^*)$, we can immediately get the approximation ratio of Algorithm 1 as follows:

Theorem 1. *For any $\ell \geq 2$ and $p \in (0, 1]$, the* RandomMultiGreedy(ℓ, p) *algorithm outputs a solution S^* satisfying*

$$f(O) \leq \frac{\ell(k + \frac{\ell}{p} - 1)}{\ell - p} \mathbb{E}[f(S^*)] \tag{8}$$

Discussion of Theorem 1: From Theorem 1, it can be seen that the approximation ratio of Algorithm 1 can be optimized by choosing proper values of ℓ and p. Indeed, the ratio can be minimized to $(1 + \sqrt{k})^2$ by setting $\ell = 2, p = \frac{2}{1+\sqrt{k}}$. Besides, if we set $\ell = \lceil \sqrt{k} \rceil + 1, p = 1$, then the approximation ratio turns into $k + \sqrt{k} + \lceil \sqrt{k} \rceil + 1$. Clearly, setting $\ell = 2$ implies faster running time as only two candidate solutions are maintained, while setting $p = 1$ implies a deterministic algorithm.

4.1. Acceleration

It can be seen that RandomMultiGreedy has time complexity of $\mathcal{O}(\ell n r)$. This time complexity can be further reduced by implementing Lines 2-5 using a "lazy evaluation" method inspired by (Minoux, 1978; Ene & Nguyen, 2019). More specifically, for each solution set S_i, we maintain an ordered list A_i which is initialized to $\mathcal{N}$. Each element $u \in A_i$ has a weight $w_i(u) = f(u \mid S_i)$ and the elements in A_i are always sorted according to the non-increasing order of their weights. When S_i changes, we pop out the top element u from A_i and discard u if $S_i \cup \{u\} \notin \mathcal{I}$. If $S_i \cup \{u\} \in \mathcal{I}$ and $f(u \mid S_i)$ has not been computed, then we update the weight of u and set $v_i = u$ if the new weight of u is at least $(1+\epsilon)^{-1}$ fraction of its old weight (otherwise u is re-inserted into A_i and we pop out the next element). During this process, any element in A_i is removed from A_i immediately when its weight has been updated for more than $\mathcal{O}(\frac{1}{\epsilon}\log\frac{\ell r}{\epsilon})$ times. Using this method, we can guarantee that $f(v_i \mid S_i)$ is at least $\frac{1}{1+\epsilon}$ fraction of the marginal gain of the best element in A_i that can be added into S_i, and the total number of incurred value and independence oracles is no more than $\mathcal{O}(\frac{n}{\epsilon}\log\frac{\ell r}{\epsilon})$ for each $S_i : i \in [\ell]$. Combining these results with Theorem 1, we can get:

Theorem 2. *For the problem of submodular maximization subject to a k-system constraint, there exist: (1) a randomized algorithm with an approximation ratio of $(1+\epsilon)(1+\sqrt{k})^2$ under $\mathcal{O}(\frac{n}{\epsilon}\log\frac{r}{\epsilon})$ time complexity, and (2) a deterministic algorithm with an approximation ratio of $(1+\epsilon)(k+\sqrt{k}+\lceil\sqrt{k}\rceil+1)$ under $\mathcal{O}(\frac{\sqrt{k}n}{\epsilon}\log\frac{\sqrt{k}r}{\epsilon})$ time complexity.*

Remark: From Theorem 2, it can be seen that Algorithm 1 actually can be regarded as a "universal algorithm" that achieves the best-known performance bounds under different settings, as explained in the following. First, if $f(\cdot)$ is non-monotone, then Algorithm 1 outperforms the state-of-the-art algorithm of (Feldman et al., 2020) in terms of both approximation ratio and time efficiency, no matter Algorithm 1 is implemented as a randomized algorithm or as a deterministic algorithm; moreover, when $(\mathcal{N}, \mathcal{I})$ is a matroid (i.e., $k = 1$), Algorithm 1 achieves an approximation ratio of $4+\epsilon$ under $O(\frac{n}{\epsilon}\log\frac{r}{\epsilon})$ time complexity, matching the performance bounds of the fastest algorithm in (Han et al., 2020) for a matroid constraint. Second, if the considered submodular function $f(\cdot)$ is monotone, it can be easily seen that the standard greedy algorithm proposed in (Fisher et al., 1978) equals to RANDOMMULTIGREEDY$(1,1)$, so Algorithm 1 also achieves the best-known approximation ratio of $k+1$ under this case.

5. Adaptive Optimization

The framework of Algorithm 1 can be naturally extended to address the adaptive case (i.e., Problem (2)), as shown by Algorithm 2. For convenience, we use $\pi_{\mathcal{A}}$ to denote the adaptive policy adopted by Algorithm 2. Algorithm 2 runs in iterations and identifies an element u^* in each iteration which maximizes the expected marginal gain $\Delta(u^* \mid \psi)$ without violating the feasibility $\mathcal{I}$, where ψ is the partial realization observed by $\pi_{\mathcal{A}}$ at the moment that u^* is identified.

Algorithm 2 ADAPTRANDOMGREEDY(p)

Initialize: $S \leftarrow \emptyset$ and $\psi \leftarrow \emptyset$
1: **while** $\mathcal{N} \neq \emptyset$ **do**
2: $\quad A \leftarrow \{u \in \mathcal{N} : S \cup \{u\} \in \mathcal{I}\}$
3: $\quad u^* \leftarrow \arg\max_{u \in A} \Delta(u \mid \psi)$
4: $\quad$ **if** $A = \emptyset \vee \Delta(u^* \mid \psi) \leq 0$ **then**
5: $\quad\quad$ **break**
6: $\quad$ **end if**
7: $\quad$ **with** probability p **do**
8: $\quad\quad$ observe $z = \Phi(u^*)$;
9: $\quad\quad S \leftarrow S \cup \{u^*\}$;
10: $\quad\quad \psi \leftarrow \psi \cup \{(u^*, z)\}$
11: $\quad \mathcal{N} \leftarrow \mathcal{N} \setminus \{u^*\}$
12: **end while**
13: **return** S

After that, $\pi_{\mathcal{A}}$ observes the state of u^* and adds u^* into the solution set S with probability p, and discard u^* with probability $1-p$. The algorithm stops when no more elements can be added into S without violating the feasibility of $\mathcal{I}$ or when $\Delta(u^* \mid \psi)$ is non-positive.

Although the framework of Algorithm 2 looks similar to RANDOMMULTIGREEDY, its performance analysis is very different, as there does not exist a fixed optimal solution set under the adaptive setting, and we have to compare the average performance of $\pi_{\mathcal{A}}$ with that of an optimal policy π_{opt}. To address this problem, we first build a relationship between $\pi_{\mathcal{A}}$ and π_{opt} as follows:

Lemma 5. *Given any two adaptive policy π_1 and π_2, let $\pi_1 @ \pi_2$ denote a new policy that first execute π_1 and then execute π_2 without any knowledge about π_1. So we have*

$$f_{\mathrm{avg}}(\pi_{\mathcal{A}} @ \pi_{\mathrm{opt}}) = f_{\mathrm{avg}}(\pi_{\mathrm{opt}} @ \pi_{\mathcal{A}}) \geq (1-p) \cdot f_{\mathrm{avg}}(\pi_{\mathrm{opt}})$$

Lemma 5 implies that we may get an approximation ratio by further bounding $f_{\mathrm{avg}}(\pi_{\mathcal{A}} @ \pi_{\mathrm{opt}})$ using $f_{\mathrm{avg}}(\pi_{\mathcal{A}})$. Given any $u \in \mathcal{N}$ and any realization ϕ, let $\psi_u(\phi)$ denote the partial realization observed by $\pi_{\mathcal{A}}$ right before u is considered by Lines 7-10 of Algorithm 2; if u is never considered, then let $\psi_u(\phi)$ denote the observed partial realization at the end of $\pi_{\mathcal{A}}$. Based on this definition, we can get:

Lemma 6. *The value of $f_{\mathrm{avg}}(\pi_{\mathcal{A}} @ \pi_{\mathrm{opt}}) - f_{\mathrm{avg}}(\pi_{\mathcal{A}})$ is no more than $\mathbb{E}_{\pi_{\mathcal{A}},\Phi}\Big[\sum_{u \in \mathcal{N}(\pi_{\mathrm{opt}},\Phi) \setminus \mathcal{N}(\pi_{\mathcal{A}},\Phi)} \Delta(u \mid \psi_u(\Phi))\Big]$, where the expectation is taken with respect to both the randomness of Φ and the randomness of $\pi_{\mathcal{A}}$.*

Next, we try to establish some quantitative relationships between $f_{\mathrm{avg}}(\pi_{\mathcal{A}})$ and the upper bound found in Lemma 6. Given any realization ϕ, Note that $\mathcal{N}(\pi_{\mathrm{opt}}, \phi) \setminus \mathcal{N}(\pi_{\mathcal{A}}, \phi)$ denotes the set of elements that are selected by π_{opt} but not $\pi_{\mathcal{A}}$ under the realization ϕ. The elements in this set can be partitioned into three disjoint sets $O_1(\phi)$, $O_2(\phi)$ and $O_3(\phi)$, where $O_2(\phi)$ denotes the set of elements that have been considered by $\pi_{\mathcal{A}}$ in Lines 7-10 but discarded (due to the probability p); $O_3(\phi)$ denotes the set of elements satisfying $\Delta(u \mid \psi_u(\phi)) \leq 0$ for all $u \in O_3(\phi)$; and the rest elements are all in $O_1(\phi)$. It can be seen that each element u in $O_1(\phi)$ must satisfy $\mathrm{dom}(\psi_u(\phi)) \cup \{u\} \notin \mathcal{I}$. Therefore, by using a similar method as that under the non-adaptive case, we can map the elements in $O_1(\phi)$ to the elements selected by $\pi_{\mathcal{A}}$ under realization ϕ, and hence prove:

Lemma 7. *We have*

$$\mathbb{E}_{\pi_{\mathcal{A}},\Phi}\Big[\sum\nolimits_{u\in O_1(\Phi)} \Delta(u \mid \psi_u(\Phi))\Big] \leq k \cdot f_{\mathrm{avg}}(\pi_{\mathcal{A}})$$

Now we try to bound the "utility loss" caused by $O_2(\phi)$. Note that although these elements are discarded (with probability $1-p$), they got a chance to be selected by $\pi_{\mathcal{A}}$ with probability p. So the ratio of the total expected (conditional) marginal gain of these elements to $f_{\mathrm{avg}}(\pi_{\mathcal{A}})$ should be no more than $(1-p)/p$, which is proved by the following lemma:

Lemma 8. *We have*

$$\mathbb{E}_{\pi_{\mathcal{A}},\Phi}\Bigg[\sum_{u\in O_2(\Phi)} \Delta(u \mid \psi_u(\Phi))\Bigg] \leq \frac{1-p}{p} \cdot f_{\mathrm{avg}}(\pi_{\mathcal{A}})$$

Combining all the above lemmas, we can get the approximation ratio of AdaptRandomGreedy as follows:

Theorem 3. AdaptRandomGreedy *achieves an approximation ratio of $\frac{pk+1}{p(1-p)}$ (i.e., $f_{\mathrm{avg}}(\pi_{\mathcal{A}}) \geq \frac{p(1-p)}{pk+1} \cdot f_{\mathrm{avg}}(\pi_{\mathrm{opt}})$) under time complexity of $\mathcal{O}(nr)$. The ratio is minimized to $(1+\sqrt{k+1})^2$ when $p = (1+\sqrt{k+1})^{-1}$.*

Remark: When the objective function $f(\cdot)$ is monotone, it can be easily seen that AdaptRandomGreedy(1) can achieve an approximation ratio of $(k+1)$–the same ratio as that in (Golovin & Krause, 2011b). Therefore, AdaptRandomGreedy(p) can also be considered as a "universal algorithm" for both non-monotone and monotone submodular maximization.

6. Performance Evaluation

In this section, we compare our algorithms with the state-of-the-art algorithms for submodular maximization subject to a k-system constraint, using the metrics of both utility and the number of oracle queries to the objective function. We implemented five algorithms in the experiments: (1) the accelerated version of our RandomMultiGreedy algorithm (as described in Sec. 4.1), abbreviated as "RAMG"; (2) the RepeatedGreedy algorithm presented in (Feldman et al., 2017), abbreviated as "REPG"; (3) the TwinGreedyFast algorithm proposed in (Han et al., 2020), abbreviated as "TGF"; (4) the FastSGS algorithm proposed in (Feldman et al., 2020), abbreviated as "FSGS"; and (5) our AdaptRandomGreedy algorithm, abbreviated as "ARG". Note that the three baseline algorithms REPG, TGF and FSGS achieve the best-known performance bounds among the related studies, as illustrated in Table 1. In all experiments, we adopt the optimal settings of each implemented algorithm such that their theoretical approximation ratio is minimized (e.g., setting $\ell = 2, p = \frac{2}{1+\sqrt{k}}$ for RAMG), and we set $\epsilon = 0.1$ whenever ϵ is an input parameter for the considered algorithms. The implemented algorithms are tested in three applications, as elaborated in the following.

6.1. Movie Recommendation

This application is also considered in (Mirzasoleiman et al., 2016; Feldman et al., 2017; Haba et al., 2020), where there are a set $\mathcal{N}$ of movies and each movie is labeled by several genres chosen from a predefined set G. The goal is to select a subset S of movies from $\mathcal{N}$ to maximize the utility

$$f(S) = \sum_{u\in\mathcal{N}}\sum_{v\in S} M_{u,v} - \sum_{u\in S}\sum_{v\in S} M_{u,v}, \quad (9)$$

under the constraint that the number of movies in S labeled by genre g is no more than m_g for all $g \in G$ and $|S| \leq m$, where $m_g : g \in G$ and m are all predefined integers. Intuitively, by using $M_{u,v}$ to denote the "similarity" between movie u and movie v, the first and second factors in Eqn. (9) encourage the "coverage" and "diversity" of the movie set S, respectively. It is indicated in (Mirzasoleiman et al., 2016; Feldman et al., 2017) that the function $f(\cdot)$ is submodular and the problem constraint is essentially a k-system constraint with $k = |G|$. In our experiments, we use the MovieLens dataset (Haba et al., 2020) containing 1793 movies, where each movie u is associated with a 25-dimensional feature vector t_u calculated from user ratings. We set $M_{u,v} = e^{-\lambda \mathrm{dist}(t_u,t_v)}$ where $\mathrm{dist}(t_u, t_v)$ denotes the Euclidean distance between t_u and t_v and λ is set to 0.2. There are three genres "Adventure", "Animation" and "Fantasy" in MovieLens, and we set $m_g = 10$ for all genres.

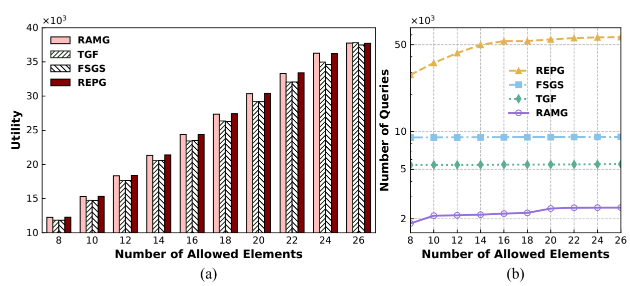


Figure 1. Movie Recommendation

In Fig. 1(a)-(b), we scale the the total number of movies allowed to be selected (i.e., m) to compare the performance of the implemented algorithms. It can be seen from Fig. 1(a) that RAMG and REPG achieve almost the same utility, while both of them outperform TGF and FSGS. Moreover, Fig. 1(b) shows that RAMG incurs much fewer oracle queries than all the baseline algorithms, and TGF is more efficient than FSGS. This can be explained by the reason that, FSGS maintains more candidate solutions than TGF, while the acceleration method adopted by RAMG is more efficient than the "thresholding" method adopted by TGF in practice.

6.2. Image Summarization

This application is also considered in (Mirzasoleiman et al., 2016; Fahrbach et al., 2019), where there is a set $\mathcal{N}$ of images classified into several categories, and the goal is to select a subset S of images from $\mathcal{N}$ to maximize the utility $f(S) = \sum_{u\in\mathcal{N}} \max_{v\in S} s_{u,v} - \frac{1}{|\mathcal{N}|} \sum_{u\in S} \sum_{v\in S} s_{u,v}$ (where $s_{u,v}$ denotes the similarity between image u and image v), under the constraint the the numbers of images in S belonging to every category and the total number of images in S are all bounded. It can be verified that such a constraint is a matroid (i.e., 1-system) constraint. We perform the experiment using the CIFAR-10 dataset (Krizhevsky et al., 2009) containing ten thousands 32×32 color images. The similarity $s_{u,v}$ is computed as the cosine similarity of the 3,072-dimensional pixel vectors of images u and v. We restrict the selection of images from three categories: Airplane, Automobile and Bird, and the number of images selected from each category is bounded by 5.

In Fig. 2, we plot the experimental results by scaling the number of images allowed to be selected. It can be seen from Fig. 2(a) that RAMG and REPG achieve approximately the same utility and outperform FSGS and TGF again. Besides, TGF performs much worse than FSGS on utility in this application, as it uses a more rigorous stopping condition in its thresholding method and hence neglects many elements with small marginal gains. The results in Fig. 2(b) show that the superiority of RAMG on efficiency still maintains, while REPG outperforms FSGS significantly. This can be explained by the fact that, the marginal gains of the unselected elements diminish vastly after a new element is selected in the image summarization application, so the performance of the thresholding method adopted in FSGS deteriorates to be close to a naive greedy algorithm, which results in its worse efficiency as FSGS maintains more candidate solutions than the other algorithms. In contrast, the performance of RAMG on efficiency is more robust against the variations of underlying data distribution.

6.3. Social Advertising with Multiple Products

This application is also considered in (Mirzasoleiman et al., 2016; Fahrbach et al., 2019; Amanatidis et al., 2020). We are given a social network $G = (\mathcal{N}, E)$ where each node represents a user and each edge $(u, v) \in E$ is associated with a weight $w_{u,v}$ denoting the "strength" that u can influence v. Suppose that there are d kinds of products and an advertiser needs to select a "seed" set $H_i \subseteq \mathcal{N}$ for each $i \in [d]$, such that the total revenue can be maximized by presenting a free sample of product with type i to each node in H_i. We also follow (Mirzasoleiman et al., 2016; Amanatidis et al., 2020) to assume that the valuation of any user for a product is determined by the neighboring nodes owning the product with the same type, and the total revenue of H_i is defined as

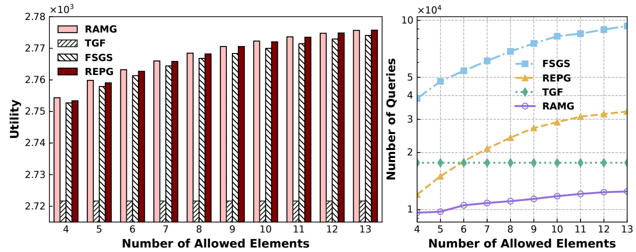


Figure 2. Image Summarization

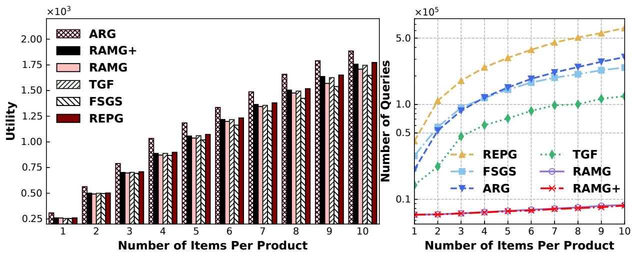
$$f_i(H_i) = \sum_{u\in\mathcal{N}\setminus H_i} \alpha_{u,i} \sqrt{\sum_{v\in H_i} w_{v,u}}, \qquad (10)$$

where $\alpha_{u,i}$ is a random number with known distributions. Suppose that each node $u \in \mathcal{N}$ can serve as a seed for at most q types of products, and the total number of free samples available for any type of product is no more than m. The goal of the advertiser is to identify the seed sets $H_1, \cdots, H_d$ to maximize the expected value of $\sum_{i\in[d]} f_i(H_i)$ under the constraints described above[2]. It is indicated in (Mirzasoleiman et al., 2016) that this problem is essentially a submodular maximization problem with a 2-system constraint.

We use the LastFM Social Network (Barbieri & Bonchi, 2014; Aslay et al., 2017) with 1372 nodes and 14708 edges, and the edge weights in the network are randomly generated from the uniform distribution $\mathcal{U}(0, 1)$. We adopt the same settings of (Amanatidis et al., 2020) to assume that, the parameter $\alpha_{u,i}$ follows a Pareto Type II distribution with $\lambda = 1, \alpha = 2$ for all node u and product i; and the parameters of u's neighboring nodes can be observed after u is selected under the adaptive setting. The values of d and q are set to 5 and 3, respectively. Following a comparison method in (Amanatidis et al., 2020), we also implement a variation of RAMG (dubbed RAMG+) where the input parameter p is randomly sampled from (0.9,1). To test the

[2]This application also has many other stochastic versions where the objective function is adaptive submodular and pointwise submodular (e.g., the scenario where only the edges' weights are stochastic). For simplicity, we only conduct experiments under the described setting similar to that in (Amanatidis et al., 2020).

performance of ARG, we randomly generate 20 realizations of the problem instance described above, and plot the average utility/number of queries of ARG on all the generated realizations.



(a) (b)

Figure 3. Social Advertising with Multiple Products

We study the performance of all algorithms in Fig. 3 by scaling the number of items of each product available for seeding (i.e., m). It can be seen from Fig. 3(a) that RAMG+, TGF and REPG achieve approximately the same utility, while the performance of RAMG and FSGS is slightly weaker. Note that RAMG+ has a weaker theoretical approximation ratio than RAMG. However, it is well known that approximation ratio is only a worst-case performance guarantee. Fig. 3(a) also reveals that ARG performs the best on utility, which is not surprising as it can take advantage on side observation. In Fig. 3(b), we compare the efficiency of all implemented algorithms and the results are qualitatively similar to those in Figs. 1-2. Note that RAMG+ performs almost the same with RAMG on efficiency, which implies that it improves the utility performance of RAMG "for free".

7. Conclusion

We have proposed the first randomized algorithms for submodular maximization with a k-system constraint, under both the non-adaptive setting and the adaptive setting. Our algorithms outperform the existing algorithms in terms both approximation ratio and time complexity, and their superiorities have also been demonstrated by extensive experimental results on several applications related to data mining and social computing.

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant No. 2018AAA0101204, the National Natural Science Foundation of China (NSFC) under Grant No. 61772491, Grant No. U20A20182 and Grant No. 61873177, and Anhui Initiative in Quantum Information Technologies under Grant No. AHY150300. Kai Han is the corresponding author.

References

Amanatidis, G., Fusco, F., Lazos, P., Leonardi, S., and Reiffenhäuser, R. Fast adaptive non-monotone submodular maximization subject to a knapsack constraint. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Aslay, C., Bonchi, F., Lakshmanan, L. V., and Lu, W. Revenue maximization in incentivized social advertising. *Proceedings of the VLDB Endowment (PVLDB)*, 10 (11):1238–1249, 2017.

Badanidiyuru, A. and Vondrák, J. Fast algorithms for maximizing submodular functions. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1497–1514, 2014.

Badanidiyuru, A., Mirzasoleiman, B., Karbasi, A., and Krause, A. Streaming submodular maximization: Massive data summarization on the fly. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 671–680, 2014.

Badanidiyuru, A., Papadimitriou, C., Rubinstein, A., Seeman, L., and Singer, Y. Locally adaptive optimization: Adaptive seeding for monotone submodular functions. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 414–429, 2016.

Badanidiyuru, A., Karbasi, A., Kazemi, E., and Vondrák, J. Submodular maximization through barrier functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Balkanski, E., Rubinstein, A., and Singer, Y. An optimal approximation for submodular maximization under a matroid constraint in the adaptive complexity model. In *ACM Symposium on Theory of Computing (STOC)*, pp. 66–77, 2019.

Barbieri, N. and Bonchi, F. Influence maximization with viral product design. In *Proceedings of the 2014 SIAM International Conference on Data Mining (SDM)*, pp. 55–63, 2014.

Buchbinder, N., Feldman, M., Naor, J., and Schwartz, R. Submodular maximization with cardinality constraints. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1433–1452, 2014.

Buchbinder, N., Feldman, M., Seffi, J., and Schwartz, R. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM Journal on Computing*, 44(5):1384–1402, 2015.

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

Chekuri, C. and Quanrud, K. Parallelizing greedy for submodular set function maximization in matroids and beyond. In *ACM Symposium on Theory of Computing (STOC)*, pp. 78–89, 2019.

Cuong, N. V. and Xu, H. Adaptive maximization of pointwise submodular functions with budget constraint. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1252–1260, 2016.

Ene, A. and Nguyen, H. L. A nearly-linear time algorithm for submodular maximization with a knapsack constraint. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 53:1–53:12, 2019.

Esfandiari, H., Amin, K., and Mirrokni, V. Adaptivity in adaptive submodularity. In *Conference on Learning Theory (COLT)*, 2021.

Fahrbach, M., Mirrokni, V., and Zadimoghaddam, M. Non-monotone submodular maximization with nearly optimal adaptivity and query complexity. In *International Conference on Machine Learning (ICML)*, pp. 1833–1842, 2019.

Feldman, M., Harshaw, C., and Karbasi, A. Greed is good: Near-optimal submodular maximization via greedy optimization. In *Conference on Learning Theory (COLT)*, pp. 758–784, 2017.

Feldman, M., Harshaw, C., and Karbasi, A. Simultaneous greedys: A swiss army knife for constrained submodular maximization. *arXiv:2009.13998*, 2020.

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

Fujii, K. and Sakaue, S. Beyond adaptive submodularity: Approximation guarantees of greedy policy with adaptive submodularity ratio. In *International Conference on Machine Learning (ICML)*, pp. 2042–2051, 2019.

Golovin, D. and Krause, A. Adaptive submodularity: theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42: 427–486, 2011a.

Golovin, D. and Krause, A. Adaptive submodular optimization under matroid constraints. *arXiv:1101.4450*, 2011b.

Gomes, R. and Krause, A. Budgeted nonparametric learning from data streams. In *International Conference on Machine Learning (ICML)*, pp. 391–398, 2010.

Gotovos, A., Karbasi, A., and Krause, A. Non-monotone adaptive submodular maximization. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1996–2003, 2015.

Gupta, A., Roth, A., Schoenebeck, G., and Talwar, K. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *International Workshop on Internet and Network Economics (WINE)*, pp. 246–257, 2010.

Haba, R., Kazemi, E., Feldman, M., and Karbasi, A. Streaming submodular maximization under a k-set system constraint. In *International Conference on Machine Learning (ICML)*, 2020.

Han, K., Huang, H., and Luo, J. Quality-aware pricing for mobile crowdsensing. *IEEE/ACM Transactions on Networking*, 26(4):1728–1741, 2018a.

Han, K., Huang, K., Xiao, X., Tang, J., Sun, A., and Tang, X. Efficient algorithms for adaptive influence maximization. *Proceedings of the VLDB Endowment*, 11(9):1029–1040, 2018b.

Han, K., Gui, F., Xiao, X., Tang, J., He, Y., Cao, Z., and Huang, H. Efficient and effective algorithms for clustering uncertain graphs. *Proceedings of the VLDB Endowment*, 12(6):667–680, 2019.

Han, K., Cao, Z., Cui, S., and Wu, B. Deterministic approximation for submodular maximization over a matroid in nearly linear time. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Han, K., Cui, S., Zhu, T., Zhang, E., Wu, B., Yin, Z., Xu, T., Tang, S., and Huang, H. Approximation algorithms for submodular data summarization with a knapsack constraint. *Proceedings of the ACM on Measurement and Analysis of Computing Systems (POMACS)*, 5(1):05:1–05:31, 2021.

Iyer, R. K. and Bilmes, J. A. Submodular optimization with submodular cover and submodular knapsack constraints. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2436–2444, 2013.

Kempe, D., Kleinberg, J., and Tardos, É. Maximizing the spread of influence through a social network. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 137–146, 2003.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kuhnle, A. Interlaced greedy algorithm for maximization of submodular functions in nearly linear time. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2371–2381, 2019.

Lee, J., Mirrokni, V. S., Nagarajan, V., and Sviridenko, M. Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM Journal on Discrete Mathematics*, 23(4):2053–2078, 2010.

Mestre, J. Greedy in approximation algorithms. In *European Symposium on Algorithms (ESA)*, pp. 528–539, 2006.

Minoux, M. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*, pp. 234–243. Springer, 1978.

Mirzasoleiman, B., Badanidiyuru, A., and Karbasi, A. Fast constrained submodular maximization: Personalized data summarization. In *International Conference on Machine Learning (ICML)*, pp. 1358–1367, 2016.

Mirzasoleiman, B., Jegelka, S., and Krause, A. Streaming non-monotone submodular maximization: Personalized video summarization on the fly. In *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1379–1386, 2018.

Mitrovic, M., Kazemi, E., Feldman, M., Krause, A., and Karbasi, A. Adaptive sequence submodularity. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5353–5364, 2019.

Parthasarathy, S. Adaptive submodular maximization under acm symposium on theory of computing (stoc)hastic item costs. In *Conference on Learning Theory (COLT)*, pp. 3133–3151, 2020.

Singla, A., Tschiatschek, S., and Krause, A. Noisy submodular maximization via adaptive sampling with applications to crowdsourced image collection summarization. In *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2037–2041, 2016.