Algorithms for Optimizing the Ratio of Monotone k-Submodular Functions

Conference Paper · June 2020

CITATIONS
0

READS
30

3 authors, including:

Grigorios Loukides
King's College London
82 PUBLICATIONS **1,104** CITATIONS

SEE PROFILE

Algorithms for Optimizing the Ratio of Monotone k-Submodular Functions

Hau Chan[1] (✉), Grigorios Loukides[2], and Zhenghui Su[1]

[1] University of Nebraska-Lincoln, Nebraska, USA
hchan3@unl.edu, zsu@huskers.unl.edu
[2] King's College London, London, UK
grigorios.loukides]@kcl.ac.uk

Abstract. We study a new optimization problem that minimizes the ratio of two monotone k-submodular functions. The problem has applications in sensor placement, influence maximization, and feature selection among many others where one wishes to make a tradeoff between two objectives, measured as a ratio of two functions (e.g., solution cost vs. quality). We develop three greedy based algorithms for the problem, with approximation ratios that depend on the curvatures and/or the values of the functions. We apply our algorithms to a sensor placement problem where one aims to install k types of sensors, while minimizing the ratio between cost and uncertainty of sensor measurements, as well as to an influence maximization problem where one seeks to advertise k products to minimize the ratio between advertisement cost and expected number of influenced users. Our experimental results demonstrate the effectiveness of minimizing the respective ratios and the runtime efficiency of our algorithms. Finally, we discuss various extensions of our problems.

Keywords: k-submodular function · greedy algorithm · approximation

1 Introduction

In many applications ranging from machine learning such as feature selection and clustering [1] to social network analysis [14], we want to select k disjoint subsets of elements from a ground set that optimize a *k-submodular* function. A k-submodular function takes in k disjoint subsets as argument and has a diminishing returns property with respect to each subset when fixing the other $k-1$ subsets [6]. For example, in sensor domain with k types of sensors, a k-submodular function can model the diminishing cost of obtaining an extra sensor of a type when fixing the numbers of sensors of other types. Most recently, the problem of k-submodular function maximization has been studied in [6,7,14,18].

In this work, we study a new optimization problem which aims to find k disjoint subsets of a ground set that minimize the ratio of two non-negative and monotone k-submodular functions. We call this the RS-k minimization problem. The problem can be used to model a situation where one needs to make a trade-off between two different objectives (e.g., solution cost and quality). For the exposition of our problem, we provide some preliminary definitions.

Let $V = \{u_1, ..., u_n\}$ be a non-empty set of n elements, $k \geq 1$ be an integer, and $[k] = \{1, \ldots, k\}$. Let also $(k+1)^V = \{(X_1, ..., X_k) \mid X_i \subseteq V,\ \forall i \in [k],\ \text{and } X_i \cap X_j = \emptyset\ , \forall i \neq j \in [k]\}$ be the set of k (pairwise) disjoint subsets of V. A function $f : (k+1)^V \to \mathbb{R}$ is *k-submodular* [6] if and only if for any $\mathbf{x}, \mathbf{y} \in (k+1)^V$, it holds that:

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \sqcap \mathbf{y}) + f(\mathbf{x} \sqcup \mathbf{y}), \tag{1}$$

where

$$\mathbf{x} \sqcap \mathbf{y} = (X_1 \cap Y_1, ..., X_k \cap Y_k) \qquad \text{and}$$
$$\mathbf{x} \sqcup \mathbf{y} = ((X_1 \cup Y_1) \setminus (\bigcup_{i \in [k] \setminus \{1\}} X_i \cup Y_i), ..., (X_k \cup Y_k) \setminus (\bigcup_{i \in [k] \setminus \{k\}} X_i \cup Y_i)).$$

When $k = 1$, this definition coincides with the standard definition of a submodular function. Given any $\mathbf{x} = (X_1, \ldots, X_k) \in (k+1)^V$ and $\mathbf{y} = (Y_1, \ldots, Y_k) \in (k+1)^V$, we write $\mathbf{x} \preceq \mathbf{y}$ if and only if $X_i \subseteq Y_i$, $\forall i \in [k]$. The function $f : (k+1)^V \to \mathbb{R}$ is *monotone* if and only if $f(\mathbf{x}) \leq f(\mathbf{y})$ for any $\mathbf{x} \preceq \mathbf{y}$.

Our RS-k minimization problem aims to find a subset of $(k+1)^V$ that minimizes the ratio of non-negative and monotone k-submodular functions f and g: $\min_{\mathbf{0} \neq \mathbf{x} \in (k+1)^V} \frac{f(\mathbf{x})}{g(\mathbf{x})}$. The maximization version can be defined analogously.

Applications. We outline some specific applications of our problem below:

1. Sensor placement: Consider a set of locations V and k different types of sensors. Each sensor can be installed in a single location and has a different purpose (e.g., monitoring humidity or temperature). Installing a sensor of type $i \in [k]$ incurs a certain cost that diminishes when we install more sensors of that type. The cost diminishes, for example, because one can purchase sensors of the same type in bulks or reuse equipment for installing the sensors of that type for multiple sensor installations [8]. We want to select a vector $\mathbf{x} = (X_1, \ldots, X_k)$ containing k subsets of locations, each corresponding to a different type of sensors, so that the sensors have measurements of low uncertainty and also have small cost. The uncertainty of the sensors in the selected locations is captured by the entropy function $H(\mathbf{x})$ (large $H(\mathbf{x})$ implies low uncertainty) and their cost is captured by the cost function $C(\mathbf{x})$; $H(\mathbf{x})$ and $C(\mathbf{x})$ are monotone k-submodular [8, 14]. The problem is to select $\mathbf{x}$ that minimizes the ratio $\frac{C(\mathbf{x})}{H(\mathbf{x})}$.

2. Influence maximization: Consider a set of users (*seeds*) V who receive incentives (e.g., free products) from a company, to influence other users to use k products through word-of-mouth effects. The expected number of influenced users $I(\mathbf{x})$ and the cost of the products $C(\mathbf{x})$ for a vector $\mathbf{x}$ of seeds are monotone k-submodular functions [13, 14]. We want to select a vector $\mathbf{x} = (X_1, \ldots, X_k)$ that contains k subsets of seeds, so that each subset is given a different product to advertise and $\mathbf{x}$ maximizes the ratio $\frac{I(\mathbf{x})}{C(\mathbf{x})}$, or equivalently minimizes $\frac{C(\mathbf{x})}{I(\mathbf{x})}$.

3. Coupled feature selection: Consider a set of features V which leads to an accurate pattern classifier. We want to predict k variables Z_1, ..., Z_k using features

$Y_1, \ldots, Y_{|V|}$. Due to communication constraints [16], each feature can be used to predict only one Z_i. When $Y_1, \ldots Y_{|V|}$ are pairwise independent given Z, the monotone k-submodular function $F(\mathbf{x}) = H(X_1, \ldots, X_k) - \sum_{i \in [k]} \sum_{j \in X_i} H(Y_j | Z_i)$, where H is the entropy function and X_i is a group of features, captures the joint quality of feature groups [16], and the monotone k-submodular function $C(\mathbf{x})$ captures their cost. The problem is to select $\mathbf{x}$ that minimizes the ratio $\frac{C(\mathbf{x})}{F(\mathbf{x})}$.

The RS-1 minimization problem has been studied in [1, 15, 17] and its applications include maximizing the F-measure in information retrieval, as well as maximizing normalized cuts and ratio cuts [1]. However, the algorithms in these works cannot be directly applied to our problem.

Contributions. Our contributions can be summarized as follows:

1. We define RS-k minimization problem, a new optimization problem that seeks to minimize the ratio of two monotone k-submodular functions and finds applications in influence maximization, sensor placement, and feature selection.

2. We introduce three greedy based approximation algorithms for the problem: k-GreedRatio, k-StochasticGreedRatio, and Sandwich Approximation Ratio (SAR). The first two algorithms have an approximation ratio that depends on the curvature of f and the size of the optimal solution. These algorithms generalize the result of [15] to k-submodular functions and improve the result of [1] for the RS-1 minimization problem. k-Stochastic-GreedRatio differs from k-GreedRatio in that it is more efficient, as it uses sampling, and in that it achieves the guarantee of k-GreedRatio with a probability of at least $1 - \delta$, for any $\delta > 0$. SAR has an approximation ratio that depends on the values of f, and it is based on the sandwich approximation strategy [10] for non-submodular maximization, which we extend to a ratio of k-submodular functions.

3. We experimentally demonstrate the effectiveness and efficiency of our algorithms on the sensor selection problem and the influence maximization problem outlined above, by comparing them to three baselines. The solutions of our algorithms have several times higher quality compared to those of the baselines.

2 Related Work

The concept of k-submodular function was introduced in [6], and the problem of maximizing a k-submodular function has been considered in [6, 7, 14, 18]. For example, [7] studied unconstrained k-submodular maximization and proposed a $\frac{k}{2k-1}$-approximation algorithm for monotone functions and a $\frac{1}{2}$-approximation algorithm for nonmonotone functions. The work of [14] studied constrained k-submodular maximization, with an upper bound on the solution size, and proposed k-Greedy-TS, a $\frac{1}{2}$-approximation algorithm for monotone functions, and a randomized version of it. These algorithms cannot deal with our problem.

When $k = 1$, the RS-k minimization problem coincides with the submodular ratio minimization problem studied in [1, 15, 17]. The work of [1] initiated the study of the latter problem and proposed GreedRatio, an $\frac{1}{1-e^{\kappa_f - 1}}$-approximation algorithm, where k_f is the curvature of a submodular function f

[9]. The work of [15] provided an improved approximation of $\frac{|X^*|}{1+(|X^*|-1)(1-\hat{\kappa_f}(X^*))}$ for GreedRatio, where $|X^*|$ is the size of an optimal solution X^* and $\hat{\kappa_f}$ is an alternative curvature notion of f [9]. The work of [17] considered the submodular ratio minimization problem where both of the functions may not be submodular and showed that GreedRatio provides approximation guarantees that depend on the submodularity ratio [5] and the curvatures of the functions. These previous results do not apply to RS-k minimization with $k > 1$.

3 Preliminaries

We may use the word dimension when referring to a particular subset of $\mathbf{x} \in (k+1)^V$. We use $\mathbf{0} = (X_1 = \{\}, \ldots, X_k = \{\})$ to denote the vector of k empty subsets and $\mathbf{x} = (X_i, \mathbf{x}_{-i})$ to highlight the set X_i in dimension i and $\mathbf{x}_{-i}$ the subsets of other dimensions except i.

We define the *marginal gain* of a k-submodular function f when adding an element to a dimension $i \in [k]$ to be $\Delta_{u,i} f(\mathbf{x}) = f(X_1, \ldots, X_{i-1}, X_i \cup \{u\}, X_{i+1}, \ldots, X_k) - f(X_1, \ldots, X_k)$, where $\mathbf{x} \in (k+1)^V$ and $u \notin \bigcup_{l \in [k]} X_l$.

Thus, we can also define k-submodular functions as those that are monotone and have a diminishing returns property in each dimension [18].

Definition 1 (k-submodular function). *A function $f : (k+1)^V \to \mathbb{R}$ is* k-submodular *if and only if: (a) $\Delta_{u,i} f(\mathbf{x}) \geq \Delta_{u,i} f(\mathbf{y})$, for all $\mathbf{x}, \mathbf{y} \in (k+1)^V$ with $\mathbf{x} \preceq \mathbf{y}$, $u \notin \cup_{\ell \in [k]} Y_i$, and $i \in [k]$, and (b) $\Delta_{u,i} f(\mathbf{x}) + \Delta_{u,j} f(\mathbf{x}) \geq 0$, for any $\mathbf{x} \in (k+1)^V$, $u \notin \cup_{\ell \in [k]} X_i$, and $i, j \in [k]$ with $i \neq j$.*

Part (a) of Definition 1 is known as the *diminishing returns* property and part (b) as *pairwise monotonicity*. We define a k-modular function as follows.

Definition 2 (k-modular function). *A function f is* k-modular *if and only if $\Delta_{u,i} f(\mathbf{x}) = \Delta_{u,i} f(\mathbf{y})$, for all $\mathbf{x}, \mathbf{y} \in (k+1)^V$ with $\mathbf{x} \preceq \mathbf{y}$, $u \notin \cup_{\ell \in [k]} Y_i$, and $i \in [k]$.*

Without loss of generality, we assume that the functions f and g in RS-k minimization are normalized such that $g(\mathbf{0}) = f(\mathbf{0}) = 0$. Moreover, we assume that, for each $u \in V$, there are dimensions $i, i' \in [k]$ such that $f(\{u\}_i, \mathbf{0}_{-i}) > 0$ and $g(\{u\}_{i'}, \mathbf{0}_{-i'}) > 0$. Otherwise, we can remove each $u \in V$ such that $g(\{u\}_i, \mathbf{0}_{-i}) = 0$ for all $i \in [k]$ from the candidate solution of the problem, as adding it to any dimension will not decrease the value of the ratio. Also, if there is $u \in V$ with $f(\{u\}_i, \mathbf{0}_{-i}) = 0$ for all $i \in [k]$, we could add u to the final solution as it will not increase the ratio. If there is more than one such element, we need to determine which dimensions to add the elements into, so that g is maximized, which boils down to solving a k-submodular function maximization problem. Applying an existing α-approximation algorithm [6, 7, 14, 18] to that problem would yield an additional approximation multiplicative factor of α to the ratio of the final solution of our problem. Finally, we assume that the values of f and g are given by value oracles.

4 The k-GreedRatio Algorithm

We present our first algorithm for the RS-k minimization problem, called k-GreedRatio. The algorithm iteratively adds an element that achieves the best marginal gain to the functions of the ratio f/g and terminates by returning the subsets (created from each iteration) that have the smallest ratio. We show that k-GreedRatio has a bounded approximation ratio that depends on the curvature of the function f. We also show that the algorithm yields an optimal solution when f and g are k-modular.

Algorithm: k-GreedRatio
Input: V, $f : (k+1)^V \to \mathbb{R}_{\geq 0}$, $g : (k+1)^V \to \mathbb{R}_{\geq 0}$
Output: Solution $\mathbf{x} \in (k+1)^V$

$j \leftarrow 0$; $\mathbf{x}_j \leftarrow \mathbf{0}$; $R \leftarrow V$; $S \leftarrow \{\}$
while $R \neq \{\}$ **do**
  $(u, i) \in \arg\min_{u \in R, i \in [k]} \frac{\Delta_{u,i} f(\mathbf{x}_j)}{\Delta_{u,i} g(\mathbf{x}_j)}$
  $\mathbf{x}_{j+1} \leftarrow (X_1, \ldots, X_i \cup \{u\}, \ldots X_k)$
  $R \leftarrow \{u \in R \mid u \notin X_i, \forall i \in [k], \text{ and } \exists i \in [k] : \Delta_{u,i} g(\mathbf{x}_{j+1}) > 0\}$
  $S \leftarrow S \cup \{\mathbf{x}_{j+1}\}$
  $j \leftarrow j + 1$
$\mathbf{x} \leftarrow \arg\min_{\mathbf{x}_j \in S} \frac{f(\mathbf{x}_j)}{g(\mathbf{x}_j)}$
return $\mathbf{x}$

4.1 The Ratio of k-Submodular Functions

Following [4,9], we define the curvature of a k-submodular function of dimension $i \in [k]$ for any $\mathbf{x} \in (k+1)^V$ as $\kappa_{f,i}(\mathbf{x}_{-i}) = 1 - \min_{u \in V \setminus \bigcup_{j \neq i} X_j} \frac{\Delta_{u,i} f(V \setminus \{u\}, \mathbf{x}_{-i})}{f(\{u\}, \mathbf{x}_{-i})}$, and $\kappa_{f,i}(X_i, \mathbf{x}_{-i}) = 1 - \min_{u \in X_i \setminus \bigcup_{j \neq i} X_j} \frac{\Delta_{u,i} f(X_i \setminus \{u\}, \mathbf{x}_{-i})}{f(\{u\}, \mathbf{x}_{-i})}$. We extend a relaxed version [9] of the above definition as $\hat{\kappa}_{f,i}(X_i, \mathbf{x}_{-i}) = 1 - \frac{\sum_{u \in X_i} \Delta_{u,i} f(X_i \setminus \{u\}, \mathbf{x}_{-i})}{\sum_{u \in X_i} f(\{u\}, \mathbf{x}_{-i})}$.

Note that, for a given $\mathbf{x} \in (k+1)^V$, $\hat{\kappa}_{f,i}(X_i, \mathbf{x}_{-i}) \leq \kappa_{f,i}(X_i, \mathbf{x}_{-i}) \leq \kappa_{f,i}(\mathbf{x}_{-i})$ when f is monotone submodular in each dimension [9].

Let $\hat{\kappa}^{\max}_{f,i}(X_i) = \max_{(X_i, \overline{\mathbf{x}}_i) \in (k+1)^V} \hat{\kappa}_{f,i}(X_i, \overline{\mathbf{x}}_{-i})$. The following lemma (whose proof easily follows from Lemma 3.1 of [9]) provides an upper bound on the sum of the individual elements of a given set of elements for a dimension.

Lemma 1. *Given any non-negative and monotone k-submodular function f, $\boldsymbol{x} \in (k+1)^V$ and $i \in [k]$, $\sum_{u \in X_i} f(\{u\}_i, \boldsymbol{x}_{-i}) \leq \frac{|X_i|}{1 + (|X_i|-1)(1 - \hat{\kappa}^{\max}_{f,i}(X_i))} f(X_i, \boldsymbol{x}_{-i})$.*

Notice that the inequality in Lemma 1 depends only on X_i. Thus, it holds for any $\overline{\mathbf{x}} \in (k+1)^V$ as long as $\overline{X}_i = X_i$. We now begin to prove the approximation guarantee of k-GreedRatio.

Let $\mathbf{x}^* = (X_1^*, ..., X_k^*) \in \arg\min_{\mathbf{0} \neq \mathbf{x} \in (k+1)^V} \frac{f(\mathbf{x})}{g(\mathbf{x})}$ be an optimal solution of the RS-k minimization problem. Let $S(\mathbf{x}^*) = \{\mathbf{x} \in (k+1)^V \mid |X_i| = \mathbb{1}[|X_i^*| >$

$0] \; \forall i \in [k]\}$ be the subsets of $(k+1)^V$ in which each dimension contains at most one element (with respect to $\mathbf{x}^*$) where $\mathbb{1}[\cdot]$ is an indicator function. Given $S(\mathbf{x}^*)$, we let $\mathbf{x}' = (X'_1, ..., X'_k) \in \arg\min_{\mathbf{0} \neq \mathbf{x} \in S(\mathbf{x}^*)} \frac{f(\mathbf{x})}{g(\mathbf{x})}$. We first compare $\mathbf{x}'$ with $\mathbf{x}^*$ using a proof idea from [15].

Theorem 1. *Given two non-negative and monotone k-submodular functions f and g, we have $\frac{f(\boldsymbol{x}')}{g(\boldsymbol{x}')} \leq \alpha \frac{f(\boldsymbol{x}^*)}{g(\boldsymbol{x}^*)}$, where $\alpha = \prod_{i \in [k] s.t. |X_i^*|>0} \frac{|X_i^*|}{1+(|X_i^*|-1)(1-\hat{\kappa}_{f,i}^{\max}(X_i^*))}$.*

Proof. We have that

$$g(\mathbf{x}^*) = g(X_1^*, .., X_k^*) \leq \sum_{u_i \in X_i^*} g(\{u_i\}_i, \mathbf{x}_{-i}^*) \leq \sum_{u_1 \in X_1^*, ..., u_k \in X_k^*} g(\{u_1\}_1, ..., \{u_k\}_k)$$

$$\leq \sum_{u_1 \in X_1^*, ..., u_k \in X_k^*} f(\{u_1\}_1, ..., \{u_k\}_k) \frac{g(\mathbf{x}')}{f(\mathbf{x}')}$$

$$\leq \prod_{i \in [k] s.t. |X_i^*|>0} \frac{|X_i^*|}{1+(|X_i^*|-1)(1-\hat{\kappa}_{f,i}^{\max}(X_i^*))} \frac{g(\mathbf{x}')}{f(\mathbf{x}')} f(\mathbf{x}^*),$$

where the first inequality is from applying Definition 1(a) to dimension i, the second inequality is from applying Definition 1(a) to other dimensions successively, the third inequality is from noting that $\frac{f(\mathbf{x}')}{g(\mathbf{x}')} \leq \frac{f(\mathbf{x})}{g(\mathbf{x})} \iff g(\mathbf{x}) \leq f(\mathbf{x}) \frac{g(\mathbf{x}')}{f(\mathbf{x}')}$ for any $\mathbf{x} \in S(\mathbf{x}^*)$ and summing up the corresponding terms, and the fourth inequality is from applying Lemma 1 repeatedly from $i = 1$ to k. □

Notice that α in Theorem 1 could be hard to compute[3]. However, the bound is tight. For instance, if f is k-modular, then $\mathbf{x}'$ yields an optimal solution.

Theorem 1 shows that we can approximate the optimal solution, using the optimal solution $\mathbf{x}'$ where each dimension contains at most one element. However, computing any $\mathbf{x}'$ cannot be done efficiently for large k. Our next theorem shows that k-GreedRatio solution can be used to approximate any $\mathbf{x}'$, which, in turn can be used to to approximate any $\mathbf{x}^*$. To begin, we let $\overline{x} \in \arg\min_{x \in V} \min_{i \in [k]} \frac{f(\{x\}_i, \mathbf{0}_{-i})}{g(\{x\}_i, \mathbf{0}_{-i})}$ and let $\bar{i}$ be the corresponding dimension.

Theorem 2. *Given two non-negative and monotone k-submodular functions f and g, we have $\frac{f(\{\overline{x}\}_{\bar{i}}, \boldsymbol{0}_{-\bar{i}})}{g(\{\overline{x}\}_{\bar{i}}, \boldsymbol{0}_{-\bar{i}})} \leq k \frac{f(\boldsymbol{x}'')}{g(\boldsymbol{x}'')}$, for any $\boldsymbol{x}'' \in S = \{\boldsymbol{x} \in (k+1)^V \mid |X_i| \leq 1 \; \forall i \in [k]\}$.*

Proof. We have that

$$g(\mathbf{x}'') = g(\{x_1''\}, ..., \{x_k''\}) \leq g(\{x_1''\}, \mathbf{0}_{-1}) + ... + g(\{x_k''\}, \mathbf{0}_{-k})$$

$$\leq [f(\{x_1''\}, \mathbf{0}_{-1}) + ... + f(\{x_k''\}, \mathbf{0}_{-k})] \frac{g(\{\overline{x}\}_{\bar{i}}, \mathbf{0}_{-\bar{i}})}{f(\{\overline{x}\}_{\bar{i}}, \mathbf{0}_{-\bar{i}})} \leq k f(\mathbf{x}'') \frac{g(\{\overline{x}\}_{\bar{i}}, \mathbf{0}_{-\bar{i}})}{f(\{\overline{x}\}_{\bar{i}}, \mathbf{0}_{-\bar{i}})},$$

[3] It is possible to obtain a computable bound by redefining the curvature related parameters with respect to $\mathbf{x}_{-i} = \mathbf{0}_{-i}$. The proof in Theorem 1 follows similarly up until the third inequality. However, the achieved approximation essentially depends on the product of the sizes of the sets (without all k but one denominator term).

where the first inequality is from applying the definition of k-submodularity according to Inequality 1 in Section 1 repeatedly, the second inequality is from noting that $\frac{f(\{\overline{x}\}_{\overline{i}},\mathbf{0}_{-\overline{i}})}{g(\{\overline{x}\}_{\overline{i}},\mathbf{0}_{-\overline{i}})} \leq \frac{f(\{u\}_j,\mathbf{0}_{-j})}{g(\{u\}_j,\mathbf{0}_{-j})} \iff g(\{u\}_j,\mathbf{0}_{-j}) \leq f(\{u\}_j,\mathbf{0}_{-j})\frac{g(\{\overline{x}\}_{\overline{i}},\mathbf{0}_{-\overline{i}})}{f(\{\overline{x}\}_{\overline{i}},\mathbf{0}_{-\overline{i}})}$ for any $u \in V$ and $j \in [k]$ and summing up the corresponding terms, and the third inequality is due to monotonicity. □

Combining Theorems 1 and 2, we have the following result.

Theorem 3. *Given two non-negative and monotone k-submodular functions f and g, k-GreedRatio finds a solution that is at most $k\alpha$ times of the optimal solution, where $\alpha = \prod_{i\in[k]s.t.|X_i^*|>0} \frac{|X_i^*|}{1+(|X_i^*|-1)(1-\hat{\kappa}_{f,i}^{\max}(X_i^*))}$, in $O(|V|^2k)$ time, assuming it is given (value) oracle access to f and g.*

Proof. Let $\mathbf{x}$ be the output of k-GreedRatio. We have that $\frac{f(\mathbf{x})}{g(\mathbf{x})} \leq \frac{f(\{\overline{x}\}_{\overline{i}},\mathbf{0}_{-\overline{i}})}{g(\{\overline{x}\}_{\overline{i}},\mathbf{0}_{-\overline{i}})} \leq k\frac{f(\mathbf{x}')}{g(\mathbf{x}')} \leq k\alpha\frac{f(\mathbf{x}^*)}{g(\mathbf{x}^*)}$ where the first inequality holds because $\overline{x}_i$ is the first element selected by the algorithm, the second inequality is due to Theorem 2 (which holds for any $\mathbf{x}'' \in S$), and the third inequality is due to Theorem 1 and $\alpha = \prod_{i\in[k]s.t.|X_i^*|>0} \frac{|X_i^*|}{1+(|X_i^*|-1)(1-\hat{\kappa}_{f,i}^{\max}(X_i^*))}$. k-GreedRatio needs $O(|V|^2k)$ time, as step 3 needs $O(|V|k)$ time and the loop in step 2 is executed $O(|V|)$ times. □

Our result generalizes the result of [15] to k-submodular functions and improves the result of [1] for the RS-1 minimization problem.

4.2 The Ratio of k-Modular Functions

Theorem 4. *Given two non-negative and monotone k-modular functions f and g, k-GreedRatio finds an optimal solution $\mathbf{x} \in \arg\min_{\mathbf{x}'\in(k+1)^V} \frac{f(\mathbf{x}')}{g(\mathbf{x}')}$. There is an $O(|V|k+|V|\log|V|)$-time implementation of k-GreedRatio, assuming it is given (value) oracle access to f and g.*

Proof. The proof follows a similar argument to [1]. From the definition of k-modular function, f and g satisfy $\Delta_{u,i}f(\mathbf{x}) = f(\{u\}_i,\mathbf{0}_{-i})$ and $\Delta_{u,i}g(\mathbf{x}) = g(\{u\}_i,\mathbf{0}_{-i})$ for all $\mathbf{x} \in (k+1)^V$, $u \notin \cup_{\ell\in[k]}X_\ell$, and $i \in [k]$.

As a result, we can provide an (efficient) alternative implementation of k-GreedRatio by computing $Q(u) = \min_{i\in[k]} \frac{f(\{u\}_i,\mathbf{0}_{-i})}{g(\{u\}_i,\mathbf{0}_{-i})}$ for each $u \in V$ and sorting $u's$ in increasing order of $Q(u)$ (breaking ties arbitrarily).

Without loss of generality, let $Q(u_1) \leq Q(u_2) ... \leq Q(u_n)$ be such an ordering and let i_1, i_2, .., i_n be the corresponding dimensions so that $\frac{f(\{u_1\}_{i_1},\mathbf{0}_{-i_1})}{g(\{u_1\}_{i_1},\mathbf{0}_{-i_1})} \leq \ldots \leq \frac{f(\{u_n\}_{i_n},\mathbf{0}_{-i_n})}{g(\{u_n\}_{i_n},\mathbf{0}_{-i_n})}$. It is not hard to see that k-GreedRatio picks the first i elements according to the ordering (each in the i iteration).

Let $\mathbf{x}^* \in \arg\min_{\mathbf{x}\in(k+1)^V} \frac{f(\mathbf{x})}{g(\mathbf{x})}$ and $r^* = \frac{f(\mathbf{x}^*)}{g(\mathbf{x}^*)}$. There must be some $u_j \in V$ such that $Q(u_j) \leq \tau^*$, otherwise r^* cannot be obtained from the elements of $\mathbf{x}^*$.

Consider the set $\mathbf{0} \neq \mathbf{x}^{\tau^*} = (X_1, ..., X_k)$ where $X_l = \{u_j \in V \mid Q(u_j) \leq \tau^* \text{ and } i_j = l\}$ for each $l \in [k]$. First note that $\mathbf{x}^{\tau^*}$ is among the solutions $\{\mathbf{x}_i\}_{i=1}^n$

obtained by k-GreedRatio. Second, we have that $\tau^* \leq \frac{f(\mathbf{x}^{\tau^*})}{g(\mathbf{x}^{\tau^*})} \leq \tau^*$ (i.e., each of the ratio is bounded by τ^*). Thus, $\frac{f(\mathbf{x}^{\tau^*})}{g(\mathbf{x}^{\tau^*})} = \tau^*$.

The above implementation takes $O(|V|k)$ and $O(|V| \log |V|)$ time to compute ratios for each element/dimension and to sort the $|V|$ elements, respectively. □

5 *k*-StochasticGreedRatio

We introduce a more efficient randomized version of k-GreedRatio that uses a smaller number of function evaluations at each iteration of the algorithm. The algorithm is linear in $|V|$ and it uses sampling in a similar way as the algorithm of [12] for submodular maximization. That is, it selects elements to add into $\mathbf{x}$ based on a sufficiently large random sample of V instead of V.

Algorithm: k-StochasticGreedRatio
Input: V, $f : (k+1)^V \to \mathbb{R}_{\geq 0}$, $g : (k+1)^V \to \mathbb{R}_{\geq 0}$, $\delta > 0$
Output: Solution $\mathbf{x} \in (k+1)^V$

1 $j \leftarrow 0$; $\mathbf{x}_j \leftarrow \mathbf{0}$; $R \leftarrow V$; $S \leftarrow \{\}$
2 **while** $R \neq \{\}$ **do**
3 Q$\leftarrow$ a random subset of size $\min\{\left\lceil \log \frac{|V|}{\delta} \right\rceil, |V|\}$ uniformly sampled with replacement from $V \setminus S$
4 $(u, i) \in \underset{u \in Q, i \in [k]}{\arg\min} \frac{\Delta_{u,i} f(\mathbf{x}_j)}{\Delta_{u,i} g(\mathbf{x}_j)}$
5 the next steps are the same as steps 4 to 9 of k-GreedRatio

---

Theorem 5. *With probability at least $1 - \delta$, k-StochasticGreedRatio outputs a solution that is: (a) at most $k\alpha$ times of the optimal solution when f and g are non-negative and monotone k-submodular, or (b) optimal when f and g are non-negative and monotone k-modular where α is the ratio in Theorem 3.*

Proof. (a) Let Q^1 be Q of the first iteration and consider the first element selected by k-GreedRatio. If $|Q^1| = |V|$, then the probability that the first element is *not* contained in Q^1 is 0. Otherwise, this probability is $\left(1 - \frac{1}{|V|}\right)^{|Q^1|} \leq \mathrm{e}^{-\log \frac{|V|}{\delta}} = \frac{\delta}{|V|}$. We have that with probability at least $1-\delta$ k-StochasticGreedRatio selects the first element. The claims follows from this and Theorem 3.

(b) It follows from the fact that k-StochasticGreedRatio selects the first element with probability at least $1 - \delta$ and Theorem 4. □

Lemma 2. *The time complexity of k-StochasticGreedRatio is $O(k|V| \log \frac{|V|}{\delta})$ for $\delta \geq \frac{|V|}{\mathrm{e}^{|V|}}$ and $O(k|V|^2)$ otherwise, where* e *is the base of the natural logarithm.*

Proof. Step 4 needs $O(k \min\{\lceil \log \frac{|V|}{\delta} \rceil, |V|\}) = O(k \min\{\log \frac{|V|}{\delta}, |V|\})$ time and it is executed $O(|V|)$ times, once per iteration of the loop in step 2. If the sample size $\min\{\lceil \log \frac{|V|}{\delta} \rceil, |V|\} = \lceil \log \frac{|V|}{\delta} \rceil$, or equivalently if $\delta \geq \frac{|V|}{\mathrm{e}^{|V|}}$, then the algorithm takes $O(k|V| \log \frac{|V|}{\delta})$ time. Otherwise, it takes $O(k|V|^2)$ time. □

6 Sandwich Approximation Ratio (SAR)

We present SAR, a greedy based algorithm that provides an approximation guarantee based on the value of f, by extending the idea of [10] from non-submodular function maximization to RS-k minimization problems. SAR uses an upper bound k-submodular function and a lower bound k-submodular function of the ratio function f/g, applies k-Greedy-TS [14] with size constraint $|V|$ using each bound function as well as f/g, and returns the solution that maximizes the ratio among the solutions constructed by k-Greedy-TS[4].

SAR uses the functions $\frac{g(x)}{2c}$ and $\frac{g(x)}{c'}$, where $2c \geq c^* = \max_{\mathbf{x}\in(k+1)^V} f(\mathbf{x})$ and $c' = \min_{x\in V} \min_{i\in[k]} f(\{x\}_i, \mathbf{0}_{-i})$. It is easy to see that these functions bound $h(\mathbf{x}) = \frac{g(\mathbf{x})}{f(\mathbf{x})}$ from below and above, respectively. While c^* cannot be computed exactly, we have $2c \geq c^*$, where c is the solution of the k-Greedy-TS $\frac{1}{2}$-approximation algorithm for maximizing a monotone k-submodular function [14], when applied with function f and solution size threshold $|V|$.

Algorithm: Sandwich Approximation Ratio (SAR)
Input: V, $f : (k+1)^V \to \mathbb{R}_{\geq 0}$, $g : (k+1)^V \to \mathbb{R}_{\geq 0}$, c, c'
Output: Solution $\mathbf{x}_{SAR} \in (k+1)^V$

$(\mathbf{x}_\ell^1, \ldots, \mathbf{x}_\ell^{|V|}) \leftarrow k$-Greedy-TS with $g/2c$ and threshold $|V|$
$(\mathbf{x}_h^1, \ldots, \mathbf{x}_h^{|V|}) \leftarrow k$-Greedy-TS with $h = g/f$ and threshold $|V|$
$(\mathbf{x}_u^1, \ldots, \mathbf{x}_u^{|V|}) \leftarrow k$-Greedy-TS with g/c' and threshold $|V|$
return $\mathbf{x}_{SAR} \leftarrow \arg\max_{\mathbf{x}\in\{\mathbf{x}_\ell^1,\ldots,\mathbf{x}_\ell^{|V|},\mathbf{x}_h^1,\ldots,\mathbf{x}_h^{|V|},\mathbf{x}_u^1,\ldots,\mathbf{x}_u^{|V|}\}} \frac{g(\mathbf{x})}{f(\mathbf{x})}$

Algorithm: k-Greedy-TS
Input: $f : (k+1)^V \to \mathbb{R}_{\geq 0}$, solution size threshold B
Output: Vector of solutions $\mathbf{x}_f^1, \ldots, \mathbf{x}_f^B$ [5]

$\mathbf{x}_f^0 \leftarrow \mathbf{0}$
for $j = 1$ *to* B **do**
  $(u, i) \in \arg\max_{u\in V\setminus R, i\in[k]} \Delta_{u,i} f(\mathbf{x}_f)$
  $\mathbf{x}_f^j \leftarrow \mathbf{x}_f^{j-1}$
  $\mathbf{x}_f^j(u) \leftarrow i$
  $R \leftarrow R \cup \{u\}$
return $(\mathbf{x}_f^1, \ldots, \mathbf{x}_f^B)$

To provide SAR's approximation guarantee, we define $\ell(\mathbf{x}) = \frac{g(\mathbf{x})}{2c}$, $h(\mathbf{x}) = \frac{g(\mathbf{x})}{f(\mathbf{x})}$, and $u(\mathbf{x}) = \frac{g(\mathbf{x})}{c'}$ for all $\mathbf{x} \in (k+1)^V$. Let $s(\mathbf{x}) = \sum_{i\in[k]} |X_i|$ be the size of any $\mathbf{x} \in (k+1)^V$. Let $\mathbf{x}_h^* \in \arg\max_{\mathbf{x}\in(k+1)^V} \frac{g(\mathbf{x})}{f(\mathbf{x})}$ be the optimal solution and $s = s(\mathbf{x}_h^*)$ be the size of the optimal solution. We let $\mathbf{x}_\ell^{j*} \in \arg\max_{\mathbf{x}\in(k+1)^V : s(\mathbf{x})=j} \ell(\mathbf{x})$, $\mathbf{x}_h^{j*} \in \arg\max_{\mathbf{x}\in(k+1)^V : s(\mathbf{x})=j} \frac{g(\mathbf{x})}{f(\mathbf{x})}$, and $\mathbf{x}_u^{j*} \in \arg\max_{\mathbf{x}\in(k+1)^V : s(\mathbf{x})=j} u(\mathbf{x})$.

[4] SAR can be easily modified to use other algorithms for monotone k-submodular maximization instead of k-Greedy-TS, such as the algorithm of [7].
[5] We modify k-Greedy-TS to return every partial solution $\mathbf{x}_f^j$, instead of only $\mathbf{x}_f^B$.

Theorem 6. *Given two non-negative and monotone k-submodular functions f and g, SAR finds a solution at most $2\Big/\max(\frac{c'}{f(\mathbf{x}_u^s)}, \frac{f(\mathbf{x}_h^{s*})}{2c})$ times the optimal solution in $O(|V|^2k)$ time, assuming it is given (value) oracle access to f and g.*

Proof. We first show that, for each $j \in [|V|]$, $\max_{\mathbf{x}^j \in \{\mathbf{x}_\ell^j, \mathbf{x}_h^j, \mathbf{x}_u^j\}} \frac{g(\mathbf{x}^j)}{f(\mathbf{x}^j)}$ from SAR approximates $\frac{g(\mathbf{x}^{j*_h})}{f(\mathbf{x}^{j*_h})}$. Since $s \in [|V|]$ and $\mathbf{x}_{SAR} \in \arg\max_{\mathbf{x}^j \in \{\mathbf{x}_\ell^j, \mathbf{x}_h^j, \mathbf{x}_u^j\}_{j \in [|V|]}} \frac{g(\mathbf{x}^j)}{f(\mathbf{x}^j)}$, our claimed approximation follows immediately. For a fixed $j \in [|V|]$, we have

$$h(\mathbf{x}_u^j) = \frac{h(\mathbf{x}_u^j)}{u(\mathbf{x}_u^j)} u(\mathbf{x}_u^j) \geq \frac{h(\mathbf{x}_u^j)}{u(\mathbf{x}_u^j)} \frac{1}{2} u(\mathbf{x}_u^{j*}) \geq \frac{h(\mathbf{x}_u^j)}{u(\mathbf{x}_u^j)} \frac{1}{2} u(\mathbf{x}_h^{j*}) \geq \frac{h(\mathbf{x}_u^j)}{u(\mathbf{x}_u^j)} \frac{1}{2} h(\mathbf{x}_h^{j*}),$$

where the first inequality is due to the use of k-GREEDY-TS in [14] for a fixed size j, the second inequality follows from the definition of $\mathbf{x}_u^{j*}$, and the third inequality is from the fact that u upper-bounds h.

We also have $h(\mathbf{x}_\ell^j) \geq \ell(\mathbf{x}_\ell^j) \geq \frac{1}{2}\ell(\mathbf{x}_\ell^{j*}) \geq \frac{1}{2}\ell(\mathbf{x}_h^{j*}) \geq \frac{\ell(\mathbf{x}_h^{j*})}{h(\mathbf{x}_h^{j*})} \frac{1}{2} h(\mathbf{x}_h^{j*})$, where the first inequality is due to the use of k-GREEDY-TS in [14] for a fixed size j, the second inequality follows from the definition of $\mathbf{x}_\ell^{j*}$, and the third inequality is from the fact that ℓ lower-bounds h.

From combining $h(\mathbf{x}_u^j) \geq \frac{h(\mathbf{x}_u^j)}{u(\mathbf{x}_u^j)} \frac{1}{2} h(\mathbf{x}_h^{j*})$ and $h(\mathbf{x}_\ell^j) \geq \frac{\ell(\mathbf{x}_h^{j*})}{h(\mathbf{x}_h^{j*})} \frac{1}{2} h(\mathbf{x}_h^{j*})$, we have

$$\max_{\mathbf{x}^j \in \{\mathbf{x}_\ell^j, \mathbf{x}_h^j, \mathbf{x}_u^j\}} h(\mathbf{x}^j) \geq \max\left(\frac{h(\mathbf{x}_u^j)}{u(\mathbf{x}_u^j)}, \frac{\ell(\mathbf{x}_h^{j*})}{h(\mathbf{x}_h^{j*})}\right) \frac{1}{2} h(\mathbf{x}_h^{j*}).$$

From the above for each j and step 4 of SAR, we have

$$h(\mathbf{x}_{\text{SAR}}) \geq \max_{j \in [|V|]} \left\{ \max\left(\frac{h(\mathbf{x}_u^j)}{u(\mathbf{x}_u^j)}, \frac{\ell(\mathbf{x}_h^{j*})}{h(\mathbf{x}_h^{j*})}\right) \frac{1}{2} h(\mathbf{x}_h^{j*}) \right\} \geq \left(\frac{h(\mathbf{x}_u^s)}{u(\mathbf{x}_u^s)}, \frac{\ell(\mathbf{x}_h^{s*})}{h(\mathbf{x}_h^{s*})}\right) \frac{1}{2} h(\mathbf{x}_h^{s*}).$$

It follows that: $\frac{f(\mathbf{x}_{\text{SAR}})}{g(\mathbf{x}_{\text{SAR}})} \leq 2\Big/ \max\left(\frac{c'}{f(\mathbf{x}_u^s)}, \frac{f(\mathbf{x}_h^{s*})}{2c}\right) \arg\min_{\mathbf{x} \in (k+1)^V} \frac{f(\mathbf{x})}{g(\mathbf{x})}$. The time complexity of SAR follows from executing k-GREEDY-TS three times. □

7 Experimental Results

We experimentally evaluate the effectiveness and efficiency of our algorithms for cost-effective variants [13] of two problems on two publicly available datasets; a sensor placement problem where sensors have k different types [14], and an influence maximization problem under the k-topic independent cascade model [14]. We compare our algorithms to three baseline algorithms, alike those in [14], as explained below. We implemented all algorithms in C++ and executed them on an Intel Xeon @ 2.60GHz with 128GB RAM. Our source code and the datasets we used are available at: `https://bitbucket.org/grigorios_loukides/ksub`.

7.1 Sensor placement

Entropy and cost functions. We first define the entropy function for the problem, following [14]. Let the set of random variables $\Omega = \{X_1, \ldots, X_n\}$ and $H(\mathcal{S}) = -\sum_{\mathbf{s} \in \text{dom } \mathcal{S}} Pr[\mathbf{s}] \cdot \log Pr[\mathbf{s}]$ be the entropy of a subset $\mathcal{S} \subseteq \Omega$ and dom $\mathcal{S}$ is the domain of $\mathcal{S}$. The conditional entropy of Ω after having observed $\mathcal{S}$ is $H(\Omega \mid \mathcal{S}) = H(\Omega) - H(\mathcal{S})$. Thus, an $\mathcal{S}$ with large entropy $H(\mathcal{S})$ has small uncertainty and is preferred. In our sensor placement problem, we want to select locations at which we will install sensors of k types, one sensor per selected location. Let $\Omega = \{X_i^u\}_{i \in [k], u \in V}$ be the set of random variables for each sensor type $i \in [k]$ and each location $u \in V$. Each X_i^u is the random variable representing the observation collected from a sensor of type i that is installed at location u. Thus, $X_i = \{X_i^u\} \subseteq \Omega$ is the set representing the observations for all locations at which a sensor of type $i \in [k]$ is installed. Then, the entropy of a vector $\mathbf{x} = (X_1, \ldots, X_k) \in (k+1)^V$ is given by $H(\mathbf{x}) = H(\cup_{i \in [k]} X_i)$.

Let c_i be the cost of installing any sensor of type $i \in [k]$. We selected each c_i uniformly at random from $[1, 10]$, unless stated otherwise, and computed the cost of a vector $\mathbf{x} = (X_1, \ldots, X_k) \in (k+1)^V$ using the cost function $C(\mathbf{x}) = \sum_{i \in [k]} c_i \cdot |X_i|^\beta$, where $\beta \in (0, 1]$ is a user-specified parameter, similarly to [8]. This function models that the cost of installing an extra sensor of any type i diminishes when more sensors of that type have been installed (i.e., when $|X_i|$ is larger) and that the total cost of installing sensors is the sum of the costs of installing all sensors of each type. The function $|X_i|^\beta$, for $\beta \in (0, 1]$, is monotone submodular, as a composition of a monotone concave function and a monotone modular function [3]. Thus, $C(\mathbf{x})$ is monotone k-submodular, as a composition of a monotone concave and a monotone k-modular function (the proof easily follows from Theorem 5.4 in [3]). The RS-k minimization problem is to minimize $\frac{C(\mathbf{x})}{H(\mathbf{x})}$. We solve the equivalent problem of maximizing $\frac{H(\mathbf{x})}{C(\mathbf{x})}$.

Setup. We evaluate our algorithms on the Intel Lab dataset (`http://db.csail.mit.edu/labdata/labdata.html`) which is preprocessed as in [14]. The dataset is a log of approximately 2.3 million values that are collected from 54 sensors installed in 54 locations in the Intel Berkeley research lab. There are three types of sensors. Sensors of type 1, 2, and 3 collect temperature, humidity, and light values, respectively. Our k-submodular functions take as argument a vector: (1) $\mathbf{x} = (X_1)$ of sensors of type 1, when $k = 1$; (2) $\mathbf{x} = (X_1, X_2)$ of sensors of type 1 and of type 2, when $k = 2$, or (3) $\mathbf{x} = (X_1, X_2, X_3)$ of sensors of type 1 and of type 2 and of type 3, when $k = 3$.

We compared our algorithms to two baselines: 1. Single(i), which allocates only sensors of type i to locations, and 2. Random, which allocates sensors of any type randomly to locations. The baselines are similar to those in [14]; the only difference is that Single(i) is based on $\frac{H}{C}$. That is, Single(i) adds into the dimension i of vector $\mathbf{x}$ the location that incurs the maximum gain with respect to $\frac{H}{C}$. We tested all different i's and report results for Single(1), which performed the best. Following [14], we used the lazy evaluation technique [11] in k-GreedRatio and SAR, for efficiency. For these algorithms, we maintain

an upper bound on the gain of adding each u in X_i, for $i \in [k]$, with respect to $\frac{H}{C}$ and apply the technique directly. For k-StochasticGreedRatio, we used $\delta = 10^{-1}$ (unless stated otherwise) and report the average over 10 runs.

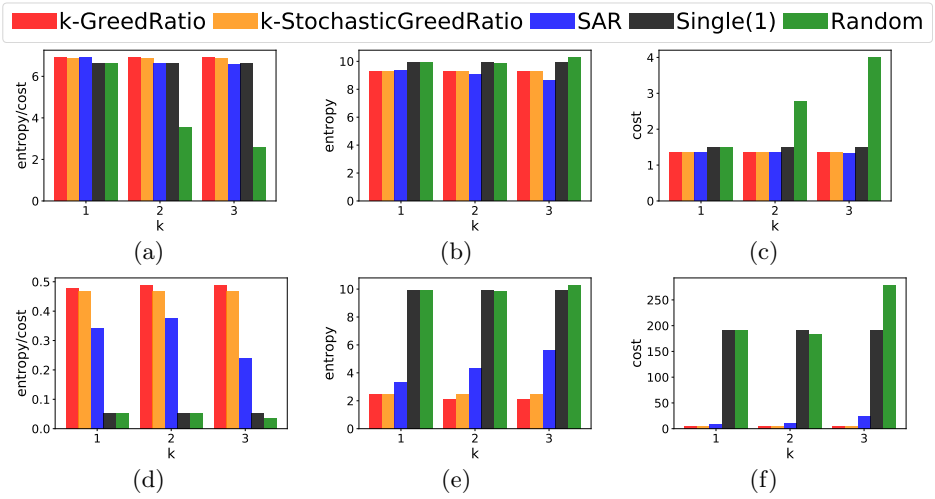


Fig. 1: (a) Entropy to cost ratio $\frac{H}{C}$ for varying $k \in [1, 3]$ and $\beta = 0.1$. (b) Entropy H for varying $k \in [1, 3]$ and $\beta = 0.1$. (c) Cost C for varying $k \in [1, 3]$ and $\beta = 0.1$. (d) Entropy to cost ratio $\frac{H}{C}$ for varying $k \in [1, 3]$ and $\beta = 0.9$. (b) Entropy H for varying $k \in [1, 3]$ and $\beta = 0.9$. (c) Cost C for varying $k \in [1, 3]$ and $\beta = 0.9$.

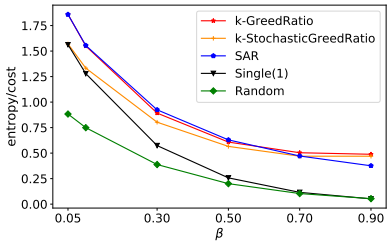


Fig. 2: Entropy to cost ratio for varying $\beta \in [0.05, 0.9]$ and $k = 2$.

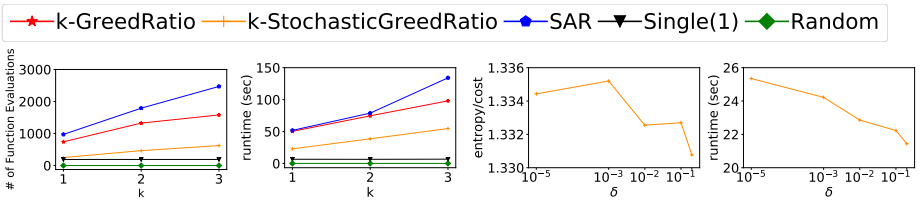
Results. Fig. 1 shows that our algorithms outperform the baselines with respect to entropy to cost ratio. In these experiments, we used $c_1 = c_2 = c_3 = 1$. Specifically, our algorithms outperform the best baseline, Single(1), by 5.2, 5.1, and 3.6 times on average over the results of Fig. 1a and 1d. Our algorithms perform best when C is close to being k-modular (i.e., in Figs. 1d, 1e and 1f where $\beta = 0.9$). In this case, they outperform Single(1) by at least 6.2 times. The cost function $C(\mathbf{x})$ increases as β increases and thus it affects the entropy to cost ratio $H(\mathbf{x})/C(\mathbf{x})$ more substantially when $\beta = 0.9$. Yet, our algorithms again selected sensors with smaller costs than the baselines (see Fig. 1f), achieving a solution with much higher ratio (see Fig. 1d). The good performance of k-GreedRatio and k-StochasticGreedRatio when $\beta = 0.9$ is because $C(\mathbf{x})$ is "close" to k-modular (it is k-modular with $\beta = 1$) and these algorithms offer a better approximation guarantee for a k-modular function (see Theorem 3).

Fig. 2 shows that our algorithms outperform the baselines with respect to entropy to cost ratio for different values of $\beta \in [0.05, 0.9]$. Specifically, k-GreedRatio, k-StochasticGreedRatio, and SAR outperform the best base-

line, Single(1), by 3.4, 3.1, and 3 times on average, respectively. Also, note that k-GreedRatio and k-StochasticGreedRatio perform very well for $\beta > 0.5$, as they again were able to select sensors with smaller costs.

Fig. 3a shows the number of function evaluations ($\frac{H}{C}$, H and C for SAR and $\frac{H}{C}$ for all other algorithms) when k varies in $[1, 3]$. The number of function evaluations is a proxy for efficiency and shows the benefit of lazy evaluation [14]. As can be seen, the number of function evaluations is the largest for SAR, since it evaluates both $h = \frac{H}{C}$ and $g = C$, while it is zero for Random, since it does not evaluate any function to select sensors. Single(1) performs a small number of evaluations of $\frac{H}{C}$, since it adds all sensors into a fixed dimension. k-StochasticGreedRatio performs fewer evaluations than k-GreedRatio due to the use of sampling. Fig. 3b shows the runtime of all algorithms for the same experiment as that of Fig. 3a. Observe that all algorithms take more time as k increases and that our algorithms are slower than the baselines, since they perform more function evaluations. However, the runtime of our algorithms increases sublinearly with k. k-StochasticGreedRatio is the fastest, while SAR is the slowest among our algorithms.

Figs. 3c and 3d show the impact of parameter δ on the entropy to cost ratio and on runtime of k-StochasticGreedRatio, respectively. As can be seen, when δ increases, the algorithm finds a slightly worse solution but runs faster. This is because a smaller δ leads to a smaller sample size. In fact, the sample size was 30% for $\delta = 10^{-5}$ and 10.9% for $\delta = 0.2$.



(a) (b) (c) (d)

Fig. 3: (a) Number of evaluations of $\frac{H}{C}$, H, or C, for varying $k \in [1, 3]$ and $\beta = 0.9$. (b) Runtime (sec) for varying $k \in [1, 3]$ and $\beta = 0.9$. (c) Entropy to cost ratio $\frac{H}{C}$ for varying $\delta \in [10^{-5}, 0.2]$ used in k-StochasticGreedRatio. (d) Runtime (sec) for varying $\delta \in [10^{-5}, 0.2]$ used in k-StochasticGreedRatio.

7.2 Influence maximization

Influence and cost functions. In the k-topic independent cascade model [14], k different topics spread through a social network independently. At the initial time $t = 0$, there is a vector $\mathbf{x} = (X_1, \ldots, X_k)$ of influenced users called *seeds*. Each seed u in X_i, $i \in [k]$, is influenced about topic i and has a single chance to influence its out-neighbor v, if v is not already influenced. The node v is influenced at $t = 1$ by u on topic i with probability $p^i_{u,v}$. When v becomes influenced, it stays influenced and has a single chance to influence each of its out-neighbors that is not already influenced. The process proceeds similarly until no new nodes are influenced. The expected number of influenced users is $I(\mathbf{x}) = \mathrm{E}[|\cup_{i\in[k]} A_i(X_i)|]$, where $A_i(X_i)$ is a random variable representing the

set of users influenced about topic i through X_i. The influence function I is shown to be k-submodular [14]. The selection of a node u as seed in X_i incurs a cost $C(u,i)$, which we selected uniformly at random from $[2000, 20000]$. The cost of $\mathbf{x}$ is $C(\mathbf{x}) = \left(\sum_{u \in \cup_{i\in[k]} X_i} C(u,i)\right)^{\beta}$, where $\beta \in (0,1]$. $C(\mathbf{x})$ is monotone k-submodular, as a composition of a monotone concave and a monotone k-modular function (the proof easily follows from Theorem 5.4 in [3]). The RS-k minimization problem is to minimize $\frac{C(\mathbf{x})}{I(\mathbf{x})}$. We solve the equivalent problem of maximizing $\frac{I(\mathbf{x})}{C(\mathbf{x})}$.

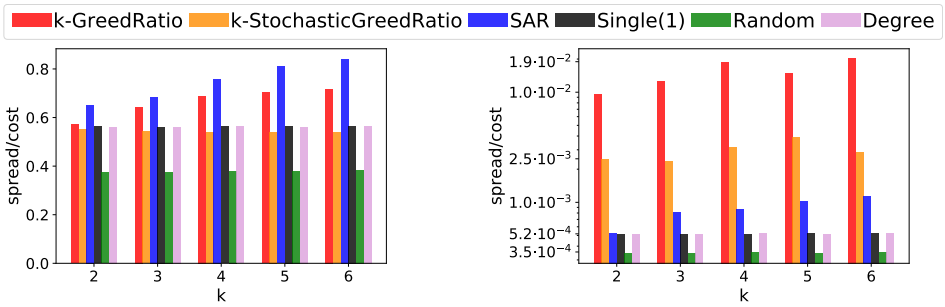
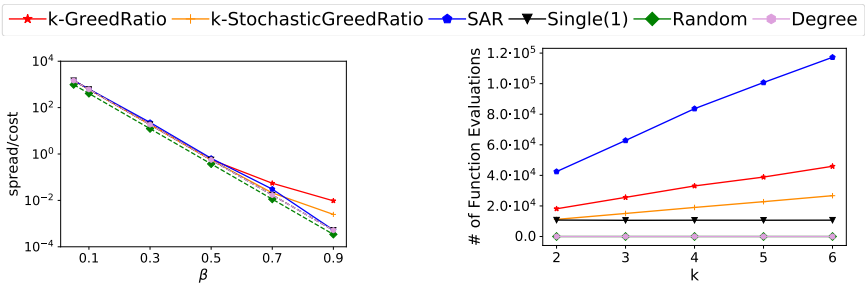


Fig. 4: Spread to cost ratio $\frac{I}{C}$ for varying: (a) $k \in [2,6]$ and $\beta = 0.5$, (b) $k \in [2,6]$ and $\beta = 0.9$

Setup. We evaluate our algorithms on a dataset of a social news website (`http://www.isi.edu/~lerman/downloads/digg2009.html`) following the setup of [14]. The dataset consists of a graph and a log of user votes for stories. Each node represents a user and each edge (u,v) represents that user u can watch the activity of v. The edge probabilities $p^i_{u,v}$ for each edge (u,v) and topic i were set using the method of [2]. We compared our algorithms to three baselines [14]: 1. Single(i); 2. Random, and 3. Degree. Single(i) is similar to that used in Section 7.1 but it is based on $\frac{I}{C}$. Following [14], we used the lazy evaluation technique [11] on k-GreedRatio, SAR, and k-StochasticGreedRatio. For the first two algorithms, we applied the technique similarly to Section 7.1. For k-StochasticGreedRatio, we maintain an upper bound on the gain of adding each u into X_i, for $i \in [1,k]$, w.r.t. $\frac{H}{C}$ and select the element in Q with the largest gain in each iteration. We tested all i's in Single(i) and report results for Single(1) that performed best. Degree sorts all nodes in decreasing order of out-degree and assigns each of them to a random topic. We simulated the influence process based on Monte Carlo simulation. For k-StochasticGreedRatio, we used $\delta = 10^{-1}$ and report the average over 10 runs.

Results. Figs. 4a and 4b show that our algorithms outperform all three baselines, by at least 15.3, 3.3, and 1.5 times on average for k-GreedRatio, k-StochasticGreedRatio, and SAR, respectively. The first two algorithms perform best when C is close to being k-modular (i.e., in Fig. 4b where $\beta = 0.9$). This is because C is k-modular when $\beta = 1$ and these algorithms offer a better approximation guarantee for a k-modular function (see Theorem 3).

Fig. 5a shows that all algorithms perform similarly for $\beta < 0.5$. This is because in these cases C has a much smaller value than I. Thus, the benefit of our algorithms in terms of selecting seeds with small costs is not significant. For $\beta \geq 0.5$, our algorithms substantially outperformed the baselines, with k-GreedRatio and k-StochasticGreedRatio performing better as β approaches 1 for the reason mentioned above.



(a) (b)

Fig. 5: (a) Spread to cost ratio for varying $\beta \in [0.05, 0.9]$ and $k = 6$. (b) Number of evaluations of $\frac{I}{C}$, I or C, for varying $k \in [2, 6]$ and $\beta = 0.7$.

Fig. 5b shows the number of evaluations of the functions $\frac{I}{C}$ and I for SAR, and of $\frac{I}{C}$ for all other algorithms, when k varies in $[2, 6]$. The number of evaluations is the largest for SAR, since SAR applies k-Greedy-TS on both $h = \frac{I}{C}$ and $g = C$, and zero for Random and Degree, since these algorithms select seeds without evaluating $\frac{I}{C}$. Single(1) performs a small number of function evaluations of $\frac{I}{C}$, since it adds all nodes into a fixed dimension (i.e., dimension 1). k-StochasticGreedRatio performs fewer function evaluations than k-GreedRatio, because it uses sampling. k-StochasticGreedRatio was also 30% faster than SAR on average but 5 times slower than Single(1).

8 Conclusion

In this paper, we studied RS-k minimization, a new optimization problem that seeks to minimize the ratio of two monotone k-submodular functions. To deal with the problem, we developed k-GreedRatio, k-StochasticGreedRatio, and Sandwich Approximation Ratio (SAR), whose approximation ratios depend on the curvatures of the k-submodular functions and the values of the functions. We also demonstrated the effectiveness and efficiency of our algorithms by applying them to sensor placement and influence maximization problems.

Extensions. One interesting question is to consider the RS-k minimization problem with size constraints, alike those for k-submodular maximization in [14]. The constrained k-minimization problem seeks to select k disjoint subsets that minimize the ratio and either all contain at most a specified number of elements, or each of them contains at most a specified number of elements. Our algorithms can be extended to tackle this constrained problem.

Another interesting question is to consider RS-k minimization when f and g are not exactly k-submodular. A recent work [17] shows that the approxima-

tion ratios of algorithm for RS-1 minimization depend on the curvatures and submodularity ratios [5] of the functions f and g, when f and g are not submodular. A similar idea can be considered for our algorithms, provided that we extend the notion of submodularity ratio to k-submodular functions.

Acknowledgments

We would like to thank the authors of [14] for providing the code of the baselines.

References

1. Bai, W., Iyer, R., Wei, K., Bilmes, J.: Algorithms for optimizing the ratio of submodular functions. In: ICML. vol. 48, pp. 2751–2759 (2016)
2. Barbieri, N., Bonchi, F., Manco, G.: Topic-aware social influence propagation models. In: ICDM. pp. 81–90 (2012)
3. Bilmes, J.A., Bai, W.: Deep submodular functions. CoRR **abs/1701.08939** (2017), `http://arxiv.org/abs/1701.08939`
4. Conforti, M., Cornuéjols, G.: Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem. Discrete Applied Mathematics **7**(3), 251 – 274 (1984)
5. Das, A., Kempe, D.: Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In: ICML. pp. 1057–1064 (2011)
6. Huber, A., Kolmogorov, V.: Towards minimizing k-submodular functions. In: Proceedings of the Second International Conference on Combinatorial Optimization. pp. 451–462 (2012)
7. Iwata, S., Tanigawa, S.i., Yoshida, Y.: Improved approximation algorithms for k-submodular function maximization. In: SODA. pp. 404–413 (2016)
8. Iyer, R., Bilmes, J.: Algorithms for approximate minimization of the difference between submodular functions, with applications. In: UAI. p. 407–417 (2012)
9. Iyer, R.K., Jegelka, S., Bilmes, J.A.: Curvature and optimal algorithms for learning and minimizing submodular functions. In: NIPS, pp. 2742–2750 (2013)
10. Lu, W., Chen, W., Lakshmanan, L.V.S.: From competition to complementarity: Comparative influence diffusion and maximization. PVLDB **9**(2), 60–71 (2015)
11. Minoux, M.: Accelerated greedy algorithms for maximizing submodular set functions. In: Optimization Techniques. pp. 234–243. Berlin, Heidelberg (1978)
12. Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrák, J., Krause, A.: Lazier than lazy greedy. In: AAAI. p. 1812–1818 (2015)
13. Nguyen, H., Zheng, R.: On budgeted influence maximization in social networks. IEEE Journal on Selected Areas in Communications **31**(6), 1084–1094 (2013)
14. Ohsaka, N., Yoshida, Y.: Monotone k-submodular function maximization with size constraints. In: NIPS. pp. 694–702 (2015)
15. Qian, C., Shi, J.C., Yu, Y., Tang, K., Zhou, Z.H.: Optimizing ratio of monotone set functions. In: IJCAI. pp. 2606–2612 (2017)
16. Soma, T.: No-regret algorithms for online k-submodular maximization. In: PMLR. vol. 89, pp. 1205–1214 (2019)
17. Wang, Y.J., Xu, D.C., Jiang, Y.J., Zhang, D.M.: Minimizing ratio of monotone non-submodular functions. Journal of the Operations Research Society of China **7**(3), 449–459 (2019)
18. Ward, J., Živný, S.: Maximizing k-submodular functions and beyond. ACM Trans. Algorithms **12**(4), 47:1–47:26 (2016)