Generalized Budgeted Submodular Set Function Maximization

Francesco Cellinese
Gran Sasso Science Institute, L'Aquila, Italy
francesco.cellinese@gssi.it

Gianlorenzo D'Angelo
Gran Sasso Science Institute, L'Aquila, Italy
gianlorenzo.dangelo@gssi.it

Gianpiero Monaco
University of L'Aquila, L'Aquila, Italy
gianpiero.monaco@univaq.it

Yllka Velaj
University of Chieti-Pescara, Pescara, Italy
yllka.velaj@unich.it

Abstract

In this paper we consider a generalization of the well-known budgeted maximum coverage problem. We are given a ground set of elements and a set of bins. The goal is to find a subset of elements along with an associated set of bins, such that the overall cost is at most a given budget, and the profit is maximized. Each bin has its own cost and the cost of each element depends on its associated bin. The profit is measured by a monotone submodular function over the elements.

We first present an algorithm that guarantees an approximation factor of $\frac{1}{2}\left(1-\frac{1}{e^{\alpha}}\right)$, where $\alpha \leq 1$ is the approximation factor of an algorithm for a sub-problem. We give two polynomial-time algorithms to solve this sub-problem. The first one gives us $\alpha = 1-\epsilon$ if the costs satisfies a specific condition, which is fulfilled in several relevant cases, including the unitary costs case and the problem of maximizing a monotone submodular function under a knapsack constraint. The second one guarantees $\alpha = 1-\frac{1}{e}-\epsilon$ for the general case. The gap between our approximation guarantees and the known inapproximability bounds is $\frac{1}{2}$.

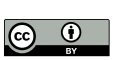
We extend our algorithm to a bi-criterion approximation algorithm in which we are allowed to spend an extra budget up to a factor $\beta \geq 1$ to guarantee a $\frac{1}{2}\left(1-\frac{1}{e^{\alpha\beta}}\right)$-approximation. If we set $\beta = \frac{1}{\alpha}\ln\left(\frac{1}{2\epsilon}\right)$, the algorithm achieves an approximation factor of $\frac{1}{2}-\epsilon$, for any arbitrarily small $\epsilon > 0$.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis, Theory of computation → Packing and covering problems

Keywords and phrases Submodular set function, Approximation algorithms, Budgeted Maximum Coverage

Digital Object Identifier 10.4230/LIPIcs.MFCS.2018.31

Related Version A full version of the paper is available at `http://arxiv.org/abs/1808.03085`.



43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018).
Editors: Igor Potapov, Paul Spirakis, and James Worrell; Article No. 31; pp. 31:1–31:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The Maximum Coverage (MC) is a fundamental combinatorial optimization problem which has several applications in job scheduling, facility locations and resource allocations [14, Ch. 3], as well as in influence maximization [17]. In the classical definition we are given a ground set X, a collection S of subsets of X with unit cost, and a budget k. The goal is selecting a subset $S' \subseteq S$, such that $|S'| \leq k$, and the number of elements of X covered by S' is maximized. A natural greedy algorithm starts with an empty solution and iteratively adds a set with maximum number of uncovered elements until k sets are selected. This algorithm has an approximation of $1 - \frac{1}{e}$ [20] and such result is tight given the inapproximability result due to Feige [11]. An interesting special case of the problem where this inapproximability result does not hold is when the size of the sets in S is small. In the maximum h-coverage, h denotes the maximum size of each set in S. This problem is APX-hard for any $h \geq 3$ [16] (notice that when $h = 2$ it is the maximum matching problem), while a simple polynomial local search heuristic has an approximation ratio very close to $\frac{2}{h}$ [4]. A polynomial time algorithm with approximation factor of $\frac{5}{6}$ is possible for the case when $h = 3$ [5]. In the Budgeted Maximum Coverage (BMC) problem, which is an extension of the maximum coverage, the cost of the sets in S are arbitrary, and thus a solution is feasible if the overall cost of the selected subset $S' \subseteq S$ is at most k. In [18], the authors present a polynomial time (greedy) algorithm with approximation factor of $1 - \frac{1}{e}$. In the Generalized Maximum Coverage (GMC) problem every set $s \in S$ has a cost $c(s)$, and every element $x \in X$ has a different weight and cost that depend on which set covers it. In [7], a polynomial time (greedy) algorithm with approximation factor of $1 - \frac{1}{e} - \epsilon$, for any $\epsilon > 0$, has been shown.

In all the above problems the profit of a solution is given by the sum of the weights of the covered elements. An important and studied extension is adopting a nonnegative, nondecreasing, submodular function f, which assigns a profit to each subset of elements. In the Submodular set Function subject to a Knapsack Constraint maximization (SFKC) problem we have a cost $c(x)$ for any element $x \in X$, and the goal is selecting a set $X' \subseteq X$ of elements that maximizes $f(X')$, where f is a monotone submodular function subject to the constraint that the sum of the costs of the selected elements is at most k. This problem admits a polynomial time algorithm that is $\left(1 - \frac{1}{e}\right)$-approximation [23]. Since the MC problem is a special case of SFKC problem, such result is tight. A more general setting was considered in [15], where the authors consider the following problem called Submodular Cost Submodular Knapsack (SCSK): given a set of elements $V = \{1, 2, \ldots, n\}$, two monotone non-decreasing submodular functions g and f ($f, g : 2^V \to \mathbb{R}$), and a budget b, the goal is finding a set of elements $X \subseteq V$ that maximizes the value $g(X)$ under the constraint that $f(X) \leq b$. They show that the problem cannot be approximated within any constant bound. Moreover, they give a $1/n$ approximation algorithm and mainly focus on bi-criterion approximation.

In this paper we consider the Generalized Budgeted submodular set function Maximization problem (GBSM) that is not captured by any of the above settings. We are given a ground set of elements X, a set of bins S, and a budget k. The goal is to find a subset of elements along with an associated set of bins such that the overall costs of both is at most a given budget and the profit is maximized. Each bin has its own cost, while the cost of each element depends on its associated bin. Finally, the profit is measured by a monotone submodular function over the elements.

We emphasize that the problem considered here is not a special case of the GMC problem, since we consider any monotone submodular functions for the profits. Moreover, it is possible to show that our cost function is not submodular and hence our problem is not a special

case of SCSK. Finally, our setting extends the SFKC problem, given that, the cost of an element is not fixed like in SFKC, but instead depends on the bin used for covering it.

In addition to its theoretical appeal, our setting is motivated by the adaptive seeding problem, which is an algorithmic challenge motivated by influence maximization in social networks [1, 22]. In its non-stochastic version, the problem is to select amongst certain accessible nodes in a network, and then select amongst neighbors of those nodes, in order to maximize a global objective function. In particular, given a set X and its neighbors $N(X)$ there is a monotone submodular function defined on $N(X)$, and the goal is to select $t \leq k$ elements in X connected to a set of size at most $k - t$ for which the submodular function has the largest value. Our setting is an extension of it since we consider more general costs.

1.1 Our results

In Section 3 we present an algorithm that guarantees an approximation factor of $\frac{1}{2}\left(1 - \frac{1}{e^{\alpha}}\right)$ for GBSM. Here, α is the approximation factor of an algorithm used to select a subset of elements whose ratio between marginal increment in the objective function and marginal cost is maximum. We give two polynomial-time algorithms to solve this sub-problem. In particular, in Section 4 we propose an algorithm that gives us $\alpha = 1 - \epsilon$ if the costs satisfy a specific condition. This latter is fulfilled in several relevant cases including the unitary costs case and the problem of maximizing a monotone submodular function under a knapsack constraint. In Section 5 we propose an algorithm that guarantees $\alpha = 1 - \frac{1}{e} - \epsilon$ for the general case.

The gap between our approximation guarantees and the known inapproximability bounds, i.e. the $1-\frac{1}{e}$ hardness for the MC problem [11] and the $1-\frac{1}{e^{1-\frac{1}{e}}}$ hardness for the non-stochastic adaptive seeding problem with knapsack constraint [21], is $\frac{1}{2}$, unless $P = NP$.

In Section 6, we extend our algorithm to a bi-criterion approximation algorithm in which we are allowed to spend an extra budget up to a factor β. An algorithm gives a $[\rho, \beta]$ bi-criterion approximation for GBSM if it is guaranteed to obtain a solution (S', X') such that $f(X') \geq \rho f(X^*)$ and $c(S', X') \leq \beta k$, where X^* is the optimal solution. We denote by β the extra-budget we are allowed to use in order to obtain a better approximation factor. Our algorithm guarantees a $\left[\frac{1}{2}\left(1 - \frac{1}{e^{\alpha\beta}}\right), \beta\right]$-approximation. If we set $\beta = \frac{1}{\alpha}\ln\left(\frac{1}{2\epsilon}\right)$, the algorithm achieves an approximation factor of $\frac{1}{2} - \epsilon$, for any arbitrarily small $\epsilon > 0$.

Due to space constraints, some of the proofs have been omitted and will appear in a full version of the paper.

1.2 Related work

Maximum coverage and submodular set function maximization are important problems. In the literature, besides the above mentioned ones, there are many other papers dealing with related issues. For instance, in the maximum coverage with group budgeted constraints, the set S is partitioned into groups, and the goal is to pick k sets from S to maximize the cardinality of their union with the restriction that at most one set can be picked from each group. In [6], the authors propose a $\frac{1}{2}$-approximation algorithms for this problem, and smaller constant approximation algorithm for the cost version. In the ground-set-cost budgeted maximum coverage problem, given a budget and a hypergraph, where each vertex has a non-negative cost and a non-negative profit, we want to select a set of hyperedges such that the total cost of the covered vertices is at most the budget and the total profit of all covered vertices is maximized. This problem is strictly harder than budgeted max coverage. The difference of our problem to the budgeted maximum coverage problem is that

the costs are associated with the covered vertices instead of the selected hyperedges. In [24], the authors obtain a $\frac{1}{2}\left(1-\frac{1}{\sqrt{e}}\right)$-approximation algorithm for graphs (which means having sets of size 2) and an FPTAS if the incidence graph of the hypergraph is a forest (i.e. the hypergraph is Berge-acyclic).

Maximizing submodular set function is another important research topic. The general version of the problem is: given a set of elements and a monotone submodular function, the goal is to find the subset of elements that gives the maximum value, subjected to some constraints. The case when the subset of elements must be an independent set of the matroid over the set of elements has been considered in [3], where the authors show an optimal randomized $\left(1-\frac{1}{e}\right)$-approximation algorithm. A simpler algorithm has been proposed in [13]. The case of multiple k matroid constraints has been considered in [19], where the authors give a $\frac{1}{k+\epsilon}$-approximation. An improved result appeared in [26]. Finally, unconstrained (resp. constrained) general non-monotone submodular maximization, have been considered in [2, 12] (resp. [25]).

Another related topic is the adaptive seeding problem in which the aim is to select amongst a set X of nodes of a network, called the *core*, and then adaptively selecting amongst the neighbors $N(X)$ of those nodes as they become accessible in order to maximize a submodular function of the selected nodes in $N(X)$ [1, 22]. An approximation algorithm with ratio $\left(1-\frac{1}{e}\right)^2$ has been proposed in [1]. In the adaptive seeding with knapsack constraints problem, nodes in X and in $N(X)$ are associated with a cost and the aim is to maximize the objective function while respecting a budget constraint. In this case, an $\left(1-\frac{1}{e}\right)\left(1-\frac{1}{e^{1-\frac{1}{e}}}\right)$-approximation algorithm is known [21]. In the non-stochastic version of these problems, all the nodes in $N(X)$ become accessible with probability one. Even in this case it is not possible to approximate an optimal solution within a factor greater than $\left(1-\frac{1}{e^{1-\frac{1}{e}}}\right)$, unless $P=NP$. A similar problem in which the core is made of the whole network and the network can be augmented by adding edges according to a given cost function has been shown to admit a 0.0878-approximation algorithm [9]. Finally, in [8, 10] the authors consider the problem where the core is made of a give set of nodes and the network can be augmented by adding edges incident only to the nodes in the core. In the unit-cost version of the problem where the cost of adding any edge is constant and equal to 1 the problem is NP-hard to be approximated within a constant factor greater than $1-(2e)^{-1}$. Then they provide a greedy approximation algorithm that guarantees an approximation factor of $1-\frac{1}{e}-\epsilon$, where ϵ is any positive real number. Then, they study the more general problem where the cost of edges is in $[0,1]$ and propose an algorithm that achieves an approximation guarantee of $1-\frac{1}{e}$ combining greedy and enumeration technique.

2 Preliminaries

We are given a set X of n elements and a set S of m bins. Let us denote the cost of a bin $s\in S$ by $c(s)\in\mathbb{R}_{\geq 0}$. For each bin $s\in S$ and element $x\in X$, we denote by $c(s,x)$ the cost of associating x to s. Given a budget $k\in\mathbb{R}_{\geq 0}$, and a monotone submodular function $f:2^X\to\mathbb{R}_{\geq 0}$[1], our goal is to find a subset X' of X and a subset $S'\neq\emptyset$ of S such that $c(S',X')=\sum_{s\in S'}c(s)+\sum_{x\in X'}\min_{s\in S'}c(s,x)\leq k$, and $f(X')$ is maximum. We call this problem the Generalized Budgeted submodular set function Maximization problem (GBSM).

[1] For a ground set X, a function $f:2^X\to\mathbb{R}_{\geq 0}$ is submodular if for any pair of sets $S\subseteq T\subseteq X$ and for any element $x\in X\setminus T$, $f(S\cup\{x\})-f(S)\geq f(T\cup\{x\})-f(T)$.

Our problem generalizes several well-known problems. Indeed, by setting $c(s,x) = \infty$, we do not allow the association of element x to bin s, while by setting $c(s,x) = 0$ we allow to assign element x to bin s with no additional cost. Moreover, we relax the constraints related to the association of elements to bins by setting $c(s) = 0$ for each $s \in S$, and $c(s_1,x) = c(s_2,x)$, for each $s_1, s_2 \in S$ and $x \in X$. By suitably combining these conditions we can capture the following problems: budgeted maximum coverage problem [18]; non-stochastic adaptive seeding problem [1] (also with knapsack constraints [21]); monotone submodular set function subject to a knapsack constraint maximization [23].

Let us consider a partial solution (S', X'). Given a set $T \subseteq X \setminus X'$, we denote by $c_{\min}(T)$ the minimum cost of associating the elements in T with a single bin in S, considering that the cost of bins in S' has been already paid, formally:

$$c_{\min}(T) = \min_{s \in S} \left\{ c_{S'}(s) + \sum_{x \in T} c(s,x) \right\},$$

where $c_{S'}(s) = c(s)$ if $s \notin S'$, and $c_{S'}(s) = 0$ if $s \in S'$. We call $c_{\min}(T)$ the *marginal cost* of T with respect to the partial solution (S', X'). We define $s_{\min}(T)$ as the bin $s \in S$ needed to cover T with cost $c_{\min}(T)$. Moreover, we denote by $\bar{c}(T)$ the cost of associating the elements in T to $s_{\min}(T)$, $\bar{c}(T) = c_{\min}(T) - c_{S'}(s_{\min}(T))$.

The *marginal increment* of $T \subseteq X$ with respect to the partial solution (S', X') is defined as $f(X' \cup T) - f(X')$. To simplify the notation, we use $g(T) = f(X' \cup T) - f(X')$ to denote the marginal increment.

In the algorithm in the next section, we will look for subsets of X that maximize the ratio between the marginal increment and the marginal cost with respect to some partial solution. In the following we define a family of subsets of X containing a set that approximates such maximal ratio. Given a partial solution (S', X'), we denote by $\mathcal{F}$ the family of subsets T of X that can be associated to bins in $S' \cup \{s\}$, for some single bin $s \in S$, with a cost such that $c(S' \cup \{s_{\min}(T)\}, T) \leq k$, formally $\mathcal{F} = \left\{ T \in 2^{X \setminus X'} \mid c(S' \cup \{s_{\min}(T)\}, T) \leq k \right\}$. A sub-family of $\mathcal{F}$ is an *α-list* with respect to (S', X') if it contains a subset T whose ratio between marginal increment and marginal cost is at least α times the optimal such ratio amongst all the subsets $\mathcal{F}$. Formally, $L \subseteq \mathcal{F}$ is an α-list with respect to (S', X') if

$$\max \left\{ \frac{g(T)}{c_{\min}(T)} \mid T \in L, c_{\min}(T) > 0 \right\} \geq \alpha \cdot \max \left\{ \frac{g(T)}{c_{\min}(T)} \mid T \in \mathcal{F}, c_{\min}(T) > 0 \right\}.$$

Note that the sets that maximize the above formula are not necessarily singletons due to the bin opening cost. Moreover, the algorithm given in the next section build partial solutions (S', X') in such a way that $c_{\min}(T) > 0$, for each $T \in \mathcal{F}$.

3 Greedy Algorithm

In this section we give an algorithm that guarantees a $\frac{1}{2}\left(1 - \frac{1}{e^{\alpha}}\right)$-approximation to the GBSM problem, if we assume that we can compute, in polynomial time, an α-list of polynomial size. In the next sections we will give two algorithms to compute such lists for bounded values of α.

The pseudo-code is reported in Algorithm 1. In the first step (line 3) we add all zero-cost bins to the solution. Then, the algorithm finds two candidate solutions. The first one is found at lines 4–11 with a greedy strategy as follows. The algorithm iteratively constructs a partial solution (S', X') by adding a subset $\hat{T}$ to X' and a bin $s_{\min}(\hat{T})$ to S'. In particular, at

Algorithm 1: General Algorithm.

Input : S, X
Output : S', X'

1 $S' := \emptyset$;
2 $X' := \emptyset$;
3 **foreach** $s \in S$ s.t. $c(s) = 0$ **do** $S' := S' \cup \{s\}$;
4 **repeat**
5 **foreach** $x \in X \setminus X'$ s.t. $c(s', x) = 0$ and $s' \in S'$ **do** $X' := X' \cup \{x\}$;
6 Build an α-list L w.r.t. (S', X');
7 $\hat{T} := \arg\max_{T \in L} \frac{f(X' \cup T) - f(X')}{c_{\min}(T)}$;
8 **if** $c(S' \cup \{s_{\min}(\hat{T})\}, X' \cup \hat{T}) \leq k$ **then**
9 $S' := S' \cup \{s_{\min}(\hat{T})\}$;
10 $X' := X' \cup \hat{T}$;
11 **until** $c(S' \cup \{s_{\min}(\hat{T})\}, X' \cup \hat{T}) > k$ or $X' = X$;
12 **if** $f(\hat{T}) \geq f(X')$ **then**
13 $S' := S' \cup \{s_{\min}(\hat{T})\}$;
14 $X' := \hat{T}$;
15 **return** (S', X');

each iteration, it first adds all the elements that can be associated to S' with cost 0 (line 5). Then, it selects a subset $\hat{T}$ that maximizes the ratio between the marginal increment and the marginal cost amongst the elements of an α-list L. Here, we assume that we have an algorithm to compute an α-list L w.r.t. (S', X') (see line 6). In the next sections, we will show how to compute L in polynomial time for some bounded α. The algorithm stops when adding the element with the maximum ratio would exceed the budget k or when $X' = X$. Without loss of generality, we can assume that at each iteration, the sets in the α-list L do not contain any element in X', since such elements do not increase the value of the marginal increment and possibly increase the marginal cost. This implies that at each iteration of the greedy procedure at least a new element in X is added to X' and then the number of iterations is $O(n)$.

Let (S_G, X_G) be the first candidate solution computed at the end of the greedy procedure. The second candidate solution (lines 12–14) is computed by using the set $\hat{T}$ that is discarded in the last iteration of the greedy procedure because adding $\{s_{\min}(\hat{T})\}$ and $\hat{T}$ to (S_G, X_G) would exceed the budget. Indeed, the second candidate solution is $(S_G \cup \{s_{\min}(\hat{T})\}, \hat{T})$. Note that this solution is feasible because $\hat{T}$ is contained in the α-list L computed in the last iteration of the greedy algorithm. Therefore, by definition of α-list, $c(S_G \cup \{s_{\min}(\hat{T})\}, \hat{T}) \leq k$.

The algorithm returns one of the two candidate solutions that maximizes the objective function.

The computational complexity of Algorithm 1 is $O(n \cdot (|L_{\max}| + cl))$, where $L_{\max}$ is the largest α-list computed and cl is the computational complexity of the algorithm at line 6. In the next sections we will show that our algorithms construct the α-lists in such a way that both $|L_{\max}|$ and cl are polynomially bounded in the input size.

In what follows we analyze the approximation ratio of Algorithm 1. The proof generalizes known arguments for monotone submodular maximization, see e.g. [7, 18, 23].

We give some additional definitions that will be used in the proof. We denote an optimal solution by (S^*, X^*). Let us consider the iterations executed by the greedy algorithm. Let $l + 1$ be the index of the iteration in which an element in the α-list is not added to X'

because it violates the budget constraint[2]. For $i = 1, 2, \ldots, l$, we define X'_i and S'_i as the sets X' and S' at the end of the i-th iteration of the algorithm, respectively. Moreover, let $X'_{l+1} = X'_l \cup \{\hat{T}\}$ and $S'_{l+1} = S'_l \cup \{s_{\min}(\hat{T})\}$, where $\hat{T}$ is the element selected at line 7 of iteration $l+1$ (see Algorithm 1). Let c_i be the value of $c_{\min}(\hat{T})$ as computed at iteration i of the greedy algorithm. The next lemma will be used in the proof of Theorem 2.

▶ **Lemma 1.** *After each iteration $i = 1, 2, \ldots, l+1$,*

$$f(X'_i) \geq \left(1 - \prod_{j=1}^{i}\left(1 - \alpha\frac{c_j}{k}\right)\right) f(X^*).$$

Armed with Lemma 1, we can prove Theorem 2.

▶ **Theorem 2.** *Algorithm 1 guarantees an approximation factor of $\frac{1}{2}\left(1 - \frac{1}{e^\alpha}\right)$ for GBSM.*

Proof. We observe that since (S'_{l+1}, X'_{l+1}) violates the budget, then $c(S'_{l+1}, X'_{l+1}) > k$. Moreover, for a sequence of numbers $a_1, a_2, \ldots, a_n$ such that $\sum_{\ell=1}^{n} a_\ell = A$, the function $\left[1 - \prod_{i=1}^{n}\left(1 - \frac{a_i \cdot \alpha}{A}\right)\right]$ achieves its minimum when $a_i = \frac{A}{n}$ and that $\left[1 - \prod_{i=1}^{n}\left(1 - \frac{a_i \cdot \alpha}{A}\right)\right] \geq 1 - \left(1 - \frac{\alpha}{n}\right)^n \geq 1 - e^{-\alpha}$. Therefore, by applying Lemma 1 for $i = l+1$ and observing that $\sum_{\ell=1}^{l+1} c_\ell = c(S'_{l+1}, X'_{l+1})$, we obtain:

$$f(X'_{l+1}) \geq \left[1 - \prod_{\ell=1}^{l+1}\left(1 - \frac{c_\ell \cdot \alpha}{k}\right)\right] f(X^*) \tag{1}$$

$$> \left[1 - \prod_{\ell=1}^{l+1}\left(1 - \frac{c_\ell \cdot \alpha}{c(S'_{l+1}, X'_{l+1})}\right)\right] f(X^*) \tag{2}$$

$$\geq \left[1 - \left(1 - \frac{\alpha}{(l+1)}\right)^{l+1}\right] f(X^*) \geq \left(1 - \frac{1}{e^\alpha}\right) f(X^*). \tag{3}$$

Since, by submodularity, $f(X'_{l+1}) \leq f(X'_l) + f(\hat{T})$, where $\hat{T}$ is the set selected at iteration $l+1$, we get

$$f(X'_l) + f(\hat{T}) \geq \left(1 - \frac{1}{e^\alpha}\right) f(X^*).$$

Hence, $\max\{f(X'_l), f(\hat{T})\} \geq \frac{1}{2}\left(1 - \frac{1}{e^\alpha}\right) f(X^*)$. The theorem follows by observing that $\hat{T}$ is the set selected as the second candidate solution at lines 12–14 of Algorithm 1. ◀

4 Computing an α-list for a particular case

In this section, we give a polynomial time algorithm to find a $(1-\epsilon)$-list with respect to a partial solution (S', X') for the particular case in which, for a given parameter $\epsilon \in (0, 1)$, the following condition holds:

$$\sum_{x \in T} c(s, x) \geq \frac{1}{\epsilon} c(s), \tag{4}$$

[2] We can assume that this iteration exists, as otherwise the algorithm is able to select $X' = X$, which is the optimum.

for each $s \in S$ and for each $T \subseteq X$ such that $|T| = \frac{1}{\epsilon}$. We observe that this condition is fulfilled for any $\epsilon \in (0,1)$ in the case in which $c(s) = 1$ and $c(s,x) \geq 1$, for each $s \in S$ and for each $x \in X$, which generalizes the non-stochastic adaptive seeding problem [1]. Indeed, in this case $\sum_{x \in T} c(s,x) \geq |T| = \frac{1}{\epsilon} c(s)$, for each $s \in S$ and for each $T \subseteq X$, such that $|T| = \frac{1}{\epsilon}$.

We give a simple algorithm that returns a $(1-\epsilon)$-list with respect to a partial solution (S', X'). The algorithm works as follows: build a list which contains all the subsets T of $X \setminus X'$ such that $|T| \leq \frac{1}{\epsilon}$ and $c(S' \cup \{s_{\min}(\hat{T})\}, \hat{T}) \leq k$.

Plugging this algorithm into line 6 of Algorithm 1, we can guarantee an approximation factor of $\frac{1}{2}\left(1 - \frac{1}{e}\right) - \epsilon'$, where $\epsilon' = \frac{1}{2e}\left(e^{\epsilon} - 1\right)$ for GBSM.

We observe that the case in this section contains the problem of maximizing a submodular set function under a knapsack constraint as a special case. Indeed, it is enough to set $c(s) = 0$, for each $s \in S$, and $c(s_1, x) = c(s_2, x)$, for each $s_1, s_2 \in S$ and $x \in X$. Note that in this case Condition 4 is satisfied for any $\epsilon \in (0,1)$. A special case of submodular set function maximization is the maximum coverage problem, and since this latter is NP-hard to be approximated within a factor greater than $\left(1 - \frac{1}{e}\right)$ [11], then the gap between the approximation factor of our algorithm and the best achievable one in polynomial time is $\frac{1}{2}$, unless $P = NP$.

It is easy to see that the computational complexity required by the algorithm in this section is $O(n^{\frac{1}{\epsilon}})$ and that $|L_{\max}| = O(n^{\frac{1}{\epsilon}})$.

In what follows, we assume that any set T^* that maximizes the ratio between marginal increment and marginal cost has size greater than $\frac{1}{\epsilon}$, as otherwise the α-list returned by our algorithm would contain such set. The following two technical lemmata will be used in the analysis of the algorithm.

▶ **Lemma 3.** *Given a monotone submodular set function $f : 2^X \to \mathbb{R}_{\geq 0}$, then, for any $X' \subseteq X$, the function $g(T) = f(X' \cup T) - f(X')$ is monotone and submodular.*

▶ **Lemma 4.** *Let us consider a monotone submodular set function $f : 2^X \to \mathbb{R}_{\geq 0}$ and a cost function $c : 2^X \to \mathbb{R}_{\geq 0}$ such that $c(T) = \sum_{x \in T} c(\{x\})$, for each $T \subseteq X$. For each set $T \subseteq X$, if T_y denotes the subset of T such that $\frac{f(T_y)}{c(T_y)}$ is maximum and $|T_y| = y$, then $\frac{f(T)}{c(T)} \leq \frac{f(T_y)}{c(T_y)}$, for any $y \leq |T|$.*

The next theorem shows the approximation ratio of the algorithm. The main idea is to consider the subset $\hat{T}$ that maximizes the ratio between the marginal increment and marginal cost in L and to derive a series of inequalities to lead us state that this value is greater than the ratio given by the optimal subset T^* times the factor $(1-\epsilon)$. We first compare the ratio computed for $\hat{T}$ with that for $T^*_{\frac{1}{\epsilon}}$ that is a subset of cardinality $\frac{1}{\epsilon}$ of maximal ratio, then, by rewriting the marginal cost formula according to its definition and by exploiting Lemmata 3 and 4, and Condition (4) we compare this ratio to that given by the subset T^* and this last inequality concludes the theorem.

▶ **Theorem 5.** *If for each $T \subseteq X$ such that $|T| = \frac{1}{\epsilon}$ and for each $s \in S$ we have $\sum_{x \in T} c(s,x) \geq \frac{1}{\epsilon} c(s)$, then the list L made of all the subsets of $X \setminus X'$ of size at most $\frac{1}{\epsilon}$ and cost at most k is a $(1-\epsilon)$-list.*

Proof. We recall that $g(T) = f(X' \cup T) - f(X')$. Given a subset T of $X \setminus X'$, we denote by T_y a subset of T such that $|T_y| = y$ and $\frac{f(T_y)}{c(T_y)}$ is maximum. Let T^* be the subset of $X \setminus X'$ that maximizes the ratio between the marginal increment and the marginal cost. Let $\hat{T}$ be

the element of L that maximizes $\frac{g(\hat{T})}{c_{\min}(\hat{T})}$. Since $|\hat{T}| \leq \frac{1}{\epsilon}$, then

$$\frac{g(\hat{T})}{c_{\min}(\hat{T})} \geq \frac{g\left(T^*_{\frac{1}{\epsilon}}\right)}{c_{\min}\left(T^*_{\frac{1}{\epsilon}}\right)} = \frac{g\left(T^*_{\frac{1}{\epsilon}}\right)}{c_{S'}\left(s_{\min}\left(T^*_{\frac{1}{\epsilon}}\right)\right) + \bar{c}\left(T^*_{\frac{1}{\epsilon}}\right)}.$$

By the hypothesis of the theorem, $\bar{c}\left(T^*_{\frac{1}{\epsilon}}\right) \geq \frac{1}{\epsilon} c\left(s_{\min}\left(T^*_{\frac{1}{\epsilon}}\right)\right)$, moreover, $c\left(s_{\min}\left(T^*_{\frac{1}{\epsilon}}\right)\right) \geq c_{S'}\left(s_{\min}\left(T^*_{\frac{1}{\epsilon}}\right)\right)$ and then $c_{S'}\left(s_{\min}\left(T^*_{\frac{1}{\epsilon}}\right)\right) \leq \epsilon \bar{c}\left(T^*_{\frac{1}{\epsilon}}\right)$. Therefore,

$$\frac{g\left(T^*_{\frac{1}{\epsilon}}\right)}{c_{S'}\left(s_{\min}\left(T^*_{\frac{1}{\epsilon}}\right)\right) + \bar{c}\left(T^*_{\frac{1}{\epsilon}}\right)} \geq \frac{g\left(T^*_{\frac{1}{\epsilon}}\right)}{\epsilon\bar{c}\left(T^*_{\frac{1}{\epsilon}}\right) + \bar{c}\left(T^*_{\frac{1}{\epsilon}}\right)} = \frac{g\left(T^*_{\frac{1}{\epsilon}}\right)}{(\epsilon+1)\bar{c}\left(T^*_{\frac{1}{\epsilon}}\right)}.$$

Since f is monotone and submodular, then, by Lemma 3, also $g\left(T^*_{\frac{1}{\epsilon}}\right)$ is submodular. By Lemma 4 follows that

$$\frac{g\left(T^*_{\frac{1}{\epsilon}}\right)}{(\epsilon+1)\bar{c}\left(T^*_{\frac{1}{\epsilon}}\right)} \geq \frac{g(T^*)}{(\epsilon+1)\bar{c}(T^*)}.$$

We now focus on the denominator, and we obtain that:

$$\frac{1}{(\epsilon+1)\bar{c}(T^*)} = \frac{1+\epsilon-\epsilon}{(\epsilon+1)\bar{c}(T^*)} = \frac{1}{\bar{c}(T^*)} - \frac{\epsilon}{(\epsilon+1)\bar{c}(T^*)} \geq \frac{1}{\bar{c}(T^*) + c_{S'}(s_{\min}(T^*))} - \frac{\epsilon}{\epsilon\bar{c}(T^*) + \bar{c}(T^*)}.$$

By applying the hypothesis $\bar{c}(T^*) \geq \frac{1}{\epsilon} c(s_{\min}(T^*))$, it follows that:

$$\frac{1}{\bar{c}(T^*) + c_{S'}(s_{\min}(T^*))} - \frac{\epsilon}{\epsilon\bar{c}(T^*) + \bar{c}(T^*)} \geq \frac{1}{\bar{c}(T^*) + c_{S'}(s_{\min}(T^*))} - \frac{\epsilon}{c(s_{\min}(T^*)) + \bar{c}(T^*)} \geq \frac{1}{\bar{c}(T^*) + c_{S'}(s_{\min}(T^*))} - \frac{\epsilon}{\bar{c}(T^*) + c_{S'}(s_{\min}(T^*))} = \frac{1-\epsilon}{c_{\min}(T^*)}.$$

To conclude:

$$\frac{g(\hat{T})}{c_{\min}(\hat{T})} \geq (1-\epsilon)\frac{g(T^*)}{c_{\min}(T^*)}.$$ ◀

5 Computing an α-list for the general case

In this section we give a polynomial time algorithm that builds a $\left(1-\frac{1}{e}\right)(1-\epsilon)$-list with respect to a partial solution (S', X'), for any $\epsilon \in (0,1)$. Using this algorithm as routine at line 6 of Algorithm 1, we can guarantee an approximation factor of

$$\frac{1}{2}\left(1 - \frac{1}{e^{\left(1-\frac{1}{e}\right)(1-\epsilon)}}\right)$$

for GBSM. We observe that this case generalizes the non-stochastic adaptive seeding with knapsack constraints problem, which cannot be approximated within a factor greater than

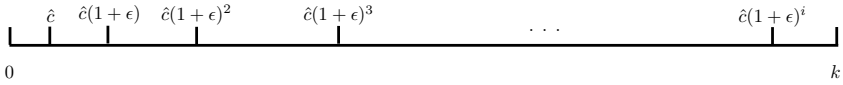


Figure 1 Growth of the budget B_i in the inner cycle of the algorithm.

$\left(1-\frac{1}{e^{1-\frac{1}{e}}}\right)$, unless $P=NP$ [21]. Then, the gap between the approximation factor of our algorithm and the best achievable one in polynomial time is $\frac{1}{2}$, unless $P=NP$.

In the algorithm of this section we make use of a procedure called `GreedyMaxCover` to maximize the value of a monotone submodular function $g: 2^X \to \mathbb{R}_{\geq 0}$, given a certain budget and costs associated to the elements of X. It is well-known that there exists a polynomial-time procedure that guarantees a $\left(1-\frac{1}{e}\right)$-approximation for this problem [23].

Let us denote by $\hat{c}$ the minimum possible positive value of functions $c(s)$ and $c(s,x)$, amongst all elements x and bins s, i.e. $\hat{c}=\min\{\min\{c(s): s\in S, c(s)>0\}, \min\{c(s,x): s\in S, x\in X, c(s,x)>0\}\}$.

The main idea is to build an α-list L which contains approximate solutions to the problem of maximizing a monotone submodular set function subject to a knapsack constraint in which the budget increases by a factor $1+\epsilon$ starting from $\hat{c}$, and the cost of the elements are given by the cost of associating them to a single bin. In particular, we consider $q=\left\lfloor \log_{1+\epsilon}\left(\frac{k}{\hat{c}}\right)\right\rfloor+1$ different budgets B_i that iteratively increase by a factor $1+\epsilon$, i.e. $B_0=\hat{c}$ and $B_i=(1+\epsilon)B_{i-1}$, for $i=1,\ldots,q$. Moreover we define $B_{q+1}=k$. For each $i=0,\ldots,q+1$ and for each bin $s\in S$, we apply procedure `GreedyMaxCover` with ground set X, budget B_i, and the cost of associating the elements to bin s as cost function. Then, we add the set returned by `GreedyMaxCover` to L. In this way we consider a budget that is at most a factor $1+\epsilon$ greater than the cost of an optimal solution and the solution returned by `GreedyMaxCover` for this budget has a value that is at most $1-\frac{1}{e}$ times smaller than that of the optimal solution.

The pseudo-code of the algorithm is reported in Algorithm 2. The outer cycle at lines 2–11 iteratively selects a bin s in S and finds a list of sets of elements assigned to bin s. The inner cycle at lines 4–8, at each iteration i, calls procedure `GreedyMaxCover` which uses g as function to maximize, $\hat{c}(1+\epsilon)^i - c_{S'}(s)$ as budget and the cost of associating the elements to bin s as cost function (to compute this costs, we only pass s as a parameter to `GreedyMaxCover`). The budget is increased by a factor $(1+\epsilon)$ until $\hat{c}(1+\epsilon)^i \geq k$. Finally the algorithm runs `GreedyMaxCover` with the full budget k. See Figure 1 for an illustration.

We call q the value of i at the end of the last iteration in the inner cycle of the algorithm. Let T_j be the set in L that maximizes the ratio between $g(T_j)$ and its assigned budget, that is:

$$T_j = \arg\max\left\{\frac{g(T_i(s))}{B_i} : s\in S, i=0,1,\ldots,q+1\right\}. \tag{5}$$

In order to bound the approximation ratio, we consider $\bar{X}^*$ as the set with the optimal ratio $\frac{g(\bar{X}^*)}{c_{\min}(\bar{X}^*)}$ amongst any possible subset of items. Let B_l be the smallest value of B_i, for $i\in\{0,1,\ldots,q+1\}$, that is greater than or equal to the cost of an optimal solution, that is the smallest B_l such that $B_l \geq c_{\min}(\bar{X}^*)$. See figure 2 for an illustration. We call T_l^* the set in L that has the highest ratio $\frac{g(T_l)}{B_l}$ amongst those computed by `GreedyMaxCover` with budget B_l, i.e. $T_l^* = \max\left\{\frac{g(T_l(s))}{B_l} : s\in S\right\}$. We also denote the set that maximizes $g(X_l^*)$ with budget B_l by X_l^*.

The idea of the approximation analysis is that an optimal solution $\bar{X}^*$ has a value of g that is at most $g(X_l^*)$ and a cost that is at most $1+\epsilon$ times smaller than B_l, while the number of iterations remains polynomial since the size of the intervals grows exponentially.

Algorithm 2: Exponential Budget Greedy.

Input : $S, X, S', X', k, \epsilon$
Output : L

$L := \emptyset$;
foreach $s \in S$ **do**
  $i := 0$;
  while $\hat{c}(1+\epsilon)^i < k$ **do**
    $B_i := \hat{c}(1+\epsilon)^i$;
    $T_i(s) := \texttt{GreedyMaxCover}(X, s, B_i - c_{S'}(s))$;
    $L := L \cup \{T_i(s)\}$;
    $i := i + 1$
  $B_i := k$;
  $T_i(s) := \texttt{GreedyMaxCover}(X, s, B_i - c_{S'}(s))$;
  $L := L \cup \{T_i(s)\}$;
return L;

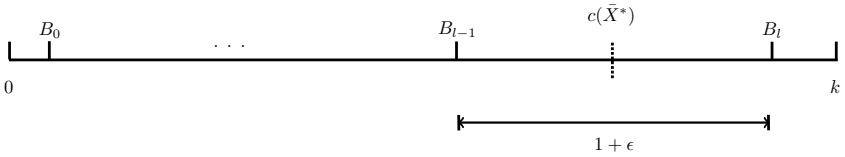


Figure 2 Notation used in Theorem 6.

The next two theorems show the bounds on approximation ratio, computational complexity and size of L.

▶ **Theorem 6.** *The list L built by Algorithm 2, is a $\left(1-\frac{1}{e}\right)(1-\epsilon)$-list.*

Proof. Since, by construction, $c_{\min}(T_j) \leq B_j$, and, by Equation 5, $\frac{g(T_j)}{B_j}$ is maximum, then

$$\frac{g(T_j)}{c_{\min}(T_j)} \geq \frac{g(T_j)}{B_j} \geq \frac{g(T_l^*)}{B_l}.$$

Procedure `GreedyMaxCover` guarantees a $\left(1-\frac{1}{e}\right)$-approximation, then

$$\frac{g(T_l^*)}{B_l} \geq \left(1-\frac{1}{e}\right)\frac{g(X_l^*)}{B_l}.$$

Moreover, since function g is monotone and $c_{\min}(\bar{X}^*) \leq B_l$, then $g(X_l^*) \geq g(\bar{X}^*)$, and therefore:

$$\left(1-\frac{1}{e}\right)\frac{g(X_l^*)}{B_l} \geq \left(1-\frac{1}{e}\right)\frac{g(\bar{X}^*)}{B_l}.$$

We defined B_l as the smallest value of B_i that is at least $c_{\min}(\bar{X}^*)$, this implies that $B_{l-1} \leq c_{\min}(\bar{X}^*)$. Moreover the ratio between B_l and B_{l-1} is $1+\epsilon$. It follows that $B_l \leq (1+\epsilon)c_{\min}(\bar{X}^*)$, which implies:

$$\left(1-\frac{1}{e}\right)\frac{g(\bar{X}^*)}{B_l} \geq \left(1-\frac{1}{e}\right)\left(\frac{1}{1+\epsilon}\right)\frac{g(\bar{X}^*)}{c_{\min}(\bar{X}^*)} \geq \left(1-\frac{1}{e}\right)(1-\epsilon)\frac{g(\bar{X}^*)}{c_{\min}(\bar{X}^*)}$$

Algorithm 3: Bi-criterion Algorithm.

Input : S, X
Output : S', X'

1 $S' := \emptyset$;
2 $X' := \emptyset$;
3 **foreach** $s \in S$ s.t. $c(s) = 0$ **do** $S' := S' \cup \{s\}$;
4 **repeat**
5 **foreach** $x \in X \setminus X'$ s.t. $c(s', x) = 0$ and $s' \in S'$ **do** $X' := X' \cup \{x\}$;
6 Build an α-list L w.r.t. (S', X');
7 $\hat{T} := \arg\max_{T \in L} \frac{f(X' \cup T) - f(X')}{c_{\min}(T)}$;
8 **if** $c(S' \cup \{s_{\min}(\hat{T})\}, X' \cup \hat{T}) \leq \beta k$ **then**
9 $S' := S' \cup \{s_{\min}(\hat{T})\}$;
10 $X' := X' \cup \hat{T}$;
11 **until** $c(S' \cup \{s_{\min}(\hat{T})\}, X' \cup \hat{T}) > \beta k$ or $X' = X$;
12 **if** $f(\hat{T}) \geq f(X')$ **then**
13 $S' := S' \cup \{s_{\min}(\hat{T})\}$;
14 $X' := \hat{T}$;
15 **return** (S', X');

The last inequality holds since $\frac{1}{1+\epsilon} = 1 - \frac{\epsilon}{1+\epsilon} \geq 1 - \epsilon$, for any $\epsilon > 0$, and this concludes the proof. ◀

▶ **Theorem 7.** *Algorithm 2 requires* $O\left(\frac{1}{\epsilon} m \cdot gr(n) \cdot \log \frac{k}{\hat{c}}\right)$ *computational time, where* $gr(n)$ *is the computational time of* `GreedyMaxCover`*, and* $|L_{\max}| = O\left(\frac{1}{\epsilon} m \log \frac{k}{\hat{c}}\right)$.

Proof. The outer for cycle requires m iterations. We now bound the number q of iteration of the inner cycle of the algorithm. By the exit condition of the cycle, we have: $\hat{c} \cdot (1+\epsilon)^q < k$, which is equivalent to: $q < \log_{1+\epsilon}\left(\frac{k}{\hat{c}}\right)$. Since for $\epsilon < 1$, $\log_{1+\epsilon}\left(\frac{k}{\hat{c}}\right) = O\left(\frac{1}{\epsilon} \log \frac{k}{\hat{c}}\right)$, the statement follows. ◀

We observe that $O(\log \frac{k}{\hat{c}})$ is polynomially bounded in the size of the input.

6 Bi-criterion approximation algorithm

In this section we extend the results given in Section 3 providing a bi-criterion approximation algorithm that guarantees a $\frac{1}{2}\left(1 - \frac{1}{e^{\alpha\beta}}\right)$-approximation to the GBSM problem, if we allow an extra budget up to a factor $\beta \geq 1$. We notice that, if $\beta = 1$, i.e. we do not increase the budget, the approximation factor is $\frac{1}{2}\left(1 - \frac{1}{e^{\alpha}}\right)$, while if $\beta = \frac{1}{\alpha} \ln\left(\frac{1}{2\epsilon}\right)$ the algorithm achieves an approximation factor of $\frac{1}{2} - \epsilon$, for any arbitrarily small $\epsilon > 0$.

The algorithm is slightly different from Algorithm 1 and it is reported in Algorithm 3. In this algorithm, we allow to exceed the given budget k by a factor β. In particular we modify lines 8 and 11, admitting a greater budget respect to Algorithm 1.

In the next theorem we show the approximation ratio of this algorithm.

▶ **Theorem 8.** *There exists an algorithm that guarantees a* $\left[\frac{1}{2}\left(1 - \frac{1}{e^{\alpha\beta}}\right), \beta\right]$ *bi-criterion approximation for GBSM, for any* $\beta \geq 1$.

It is possible to prove the theorem by using the same argument as in Theorem 2, by taking into account the new budget βk. We also exploit the fact that Lemma 1 also holds when considering the budget βk.

7 Conclusion

In this paper we defined a new challenging problem which leads to many open problems and new research questions, we referred to it as the generalized budgeted submodular set function maximization problem.

The main open problem is to close the gap between the known hardness result of $1 - \frac{1}{e^\alpha}$, where $\alpha = 1$ for the MC problem [11] and $\alpha = 1 - \frac{1}{e}$ for the non-stochastic adaptive seeding problem with knapsack constraint problem [21], and our approximation bound of $\frac{1}{2}\left(1 - \frac{1}{e^\alpha}\right)$. One possibility to get rid of the $\frac{1}{2}$ factor could be to use the partial enumeration technique exploited in specific subproblems (e.g. budgeted maximum coverage problem [18] and monotone submodular set function subject to a knapsack constraint maximization problem [23]). However, this requires that each greedy step selects a single element of X, to be added to a partial solution X', while our greedy algorithm selects a subset of $X \setminus X'$ that maximizes the ratio between its marginal increment in the objective function and its marginal cost. Note that this set can contain more than one element in order to ensure that the ratio is non-increasing at each iteration of the greedy algorithm, which is needed to apply the analysis in [18] and [23].

Other research directions, that deserve further investigation, include the study of the GBSM considering different cost functions and also different objective functions where the profit given by an element x depends on the bin s which it is associated with. It would be interesting also to analyse GBSM in the case that each bin $s \in S$ has its own budget k to use in order to maximize the objective function.

References

1 Ashwinkumar Badanidiyuru, Christos H. Papadimitriou, Aviad Rubinstein, Lior Seeman, and Yaron Singer. Locally adaptive optimization: Adaptive seeding for monotone submodular functions. In *27th ACM-SIAM Symp. on Disc. Alg., (SODA)*, pages 414–429, 2016.

2 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *53rd IEEE Symp. on Foundations of Computer Science, FOCS*, pages 649–658, 2012.

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

4 Ioannis Caragiannis. Wavelength management in WDM rings to maximize the number of connections. *SIAM J. Discrete Math.*, 23(2):959–978, 2009.

5 Ioannis Caragiannis and Gianpiero Monaco. A 6/5-approximation algorithm for the maximum 3-cover problem. *J. Comb. Optim.*, 25(1):60–77, 2013.

6 Chandra Chekuri and Amit Kumar. Maximum coverage problem with group budget constraints and applications. In *7th Intl. Work. on Approximation Algorithms for Combinatorial Optimization Problems, APPROX*, pages 72–83, 2004.

7 Reuven Cohen and Liran Katzir. The generalized maximum coverage problem. *Information Processing Letters*, 108(1):15–22, 2008.

8 Gianlorenzo D'Angelo, Lorenzo Severini, and Yllka Velaj. Influence maximization in the independent cascade model. In *Proceedings of the 17th Italian Conference on Theoretical Computer Science (ICTCS2016)*, volume 1720, pages 269–274. CEUR-WS.org, 2016.

9 Gianlorenzo D'Angelo, Lorenzo Severini, and Yllka Velaj. Selecting nodes and buying links to maximize the information diffusion in a network. In *42st Intl. Symp. on Mathematical Foundations of Computer Science, MFCS*, volume 83 of *LIPIcs*, pages 75:1–75:14, 2017.

10 Gianlorenzo D'Angelo, Lorenzo Severini, and Yllka Velaj. Recommending links through influence maximization. *Theoretical Computer Science*, 2018. doi:10.1016/j.tcs.2018.01.017.

11 Uriel Feige. A threshold of ln n for approximating set cover. *Journal of ACM*, 45(4):634–652, 1998.

12 Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. In *48th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 461–471, 2007.

13 Yuval Filmus and Justin Ward. Monotone submodular maximization over a matroid via non-oblivious local search. *SIAM J. Comput.*, 43(2):514–542, 2014.

14 D.S. Hochbaum. *Approximation Algorithms for NPHard Problems.* PWS Publishing Company,, Boston, MA, USA, 1997.

15 Rishabh K. Iyer and Jeff A. Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *27th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2436–2444, 2013.

16 V Kann. Maximum bounded 3-dimensional matching is max snp-comple. *Information Processing Letters*, 37:27–35, 1991.

17 David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11:105–147, 2015.

18 Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

19 Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Math. Oper. Res.*, 35(4):795–806, 2010.

20 G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions–I. *Mathematical Programming*, 14(1):265–294, 1978.

21 Aviad Rubinstein, Lior Seeman, and Yaron Singer. Approximability of adaptive seeding under knapsack constraints. In *16th ACM Conf. on Economics and Computation*, pages 797–814. ACM, 2015.

22 Lior Seeman and Yaron Singer. Adaptive seeding in social networks. In *IEEE 54th Symp. on Foundations of Computer Science (FOCS)*, pages 459–468. IEEE, 2013.

23 Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operation Research Letters*, 32(1):41–43, 2004.

24 Irving van Heuven van Staereling, Bart de Keijzer, and Guido Schäfer. The Ground-Set-Cost Budgeted Maximum Coverage Problem. In *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, volume 58 of *LIPIcs*, pages 50:1–50:13, 2016.

25 Jan Vondrák, Chandra Chekuri, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *43rd ACM Symp. on Theory of Computing, STOC*, pages 783–792, 2011.

26 Justin Ward. A (k+3)/2-approximation algorithm for monotone submodular k-set packing and general k-exchange systems. In *29th Intl. Symp.on Theoretical Aspects of Computer Science, STACS*, pages 42–53, 2012.