

Approximations for Monotone and Non-monotone Submodular Maximization with Knapsack Constraints*

Ariel Kulik[†] Hadas Shachnai[‡] Tami Tamir[§]

Abstract

Submodular maximization generalizes many fundamental problems in discrete optimization, including Max-Cut in directed/undirected graphs, maximum coverage, maximum facility location and marketing over social networks.

In this paper we consider the problem of maximizing any submodular function subject to d knapsack constraints, where d is a fixed constant. We establish a strong relation between the discrete problem and its continuous relaxation, obtained through *extension by expectation* of the submodular function. Formally, we show that, for any non-negative submodular function, an α -approximation algorithm for the continuous relaxation implies a randomized $(\alpha - \varepsilon)$ -approximation algorithm for the discrete problem. We use this relation to improve the best known approximation ratio for the problem to $1/4 - \varepsilon$, for any $\varepsilon > 0$, and to obtain a nearly optimal $(1 - e^{-1} - \varepsilon)$ -approximation ratio for the monotone case, for any $\varepsilon > 0$. We further show that the probabilistic domain defined by a continuous solution can be reduced to yield a polynomial size domain, given an oracle for the extension by expectation. This leads to a deterministic version of our technique.

Our approach has a potential of wider applicability, which we demonstrate on the examples of the Generalized Assignment Problem and Maximum Coverage with additional knapsack constraints.

Keywords: Submodular maximization, knapsack constraints, maximum coverage, generalized assignment problem, randomization, approximation algorithms

AMS subject classifications: 68W20, 68W25

1 Introduction

A real-valued function f , whose domain is all the subsets of a universe U , is called *submodular* if, for any $S, T \subseteq U$,

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T).$$

The concept of submodularity, which can be viewed as a discrete analog of convexity, plays a central role in combinatorial theorems and algorithms (see, e.g., [13] and the references therein, and the comprehensive surveys in [11, 27, 21]). Submodular maximization generalizes many

*A preliminary version of this paper appeared in the Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, New York, January 2009.

[†]Computer Science Department, Technion, Haifa 32000, Israel. E-mail: ariel.kulik@gmail.com

[‡]Computer Science Department, Technion, Haifa 32000, Israel. E-mail: hadas@cs.technion.ac.il. Work partially supported by the Technion V.P.R. Fund, by Smoler Research Fund, and by the Ministry of Trade and Industry MAGNET program through the NEGEV Consortium (www.negev-initiative.org).

[§]School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. E-mail: tami@idc.ac.il

fundamental problems in discrete optimization, including Max-Cut in directed/undirected graphs, maximum coverage, maximum facility location and marketing over social networks (see, e.g., [15]).

In many settings, including set covering or matroid optimization, the underlying submodular functions are monotone, meaning that $f(S) \leq f(T)$ whenever $S \subseteq T$. In other settings, the function $f(S)$ is not necessarily monotone. A classic example of such a submodular function is $f(S) = \sum_{e \in \delta(S)} w(e)$, where $\delta(S)$ is a cut in a graph (or hypergraph) $G = (V, E)$ induced by a set of vertices $S \subseteq V$, and $w(e)$ is the weight of an edge $e \subseteq E$. An example for a monotone submodular function is $f_{G, \bar{p}} : 2^L \rightarrow \mathbb{R}$, defined on a subset of vertices in bipartite graph $G = (L, R, E)$. For any $S \subseteq V$, $f_{G, \bar{p}}(S) = \sum_{v \in N(S)} p_v$, where $N(S)$ is the neighborhood function (i.e., $N(S)$ is the set of neighbors of S), and $p_v \geq 0$ is the profit of v , for any $v \in R$. The problem $\max\{f_{G, \bar{p}}(S) \mid |S| \leq k\}$ is classical maximum coverage.

In this paper we consider the following problem of maximizing a non-negative *submodular* set function subject to d *knapsack constraints* (SUB). Given a d -dimensional budget vector \bar{L} , for some $d \geq 1$, and an oracle for a non-negative submodular set function f over a universe U , where each element $i \in U$ is associated with a d -dimensional cost vector $\bar{c}(i)$, we seek a subset of elements $S \subseteq U$ whose total cost is at most \bar{L} , such that $f(S)$ is maximized.

There has been extensive work on maximizing submodular *monotone* functions subject to matroid constraint.¹ For the special case of uniform matroid, i.e., the problem $\{\max f(S) : |S| \leq k\}$, for some $k > 1$, Nemhauser et. al showed in [24] that a greedy algorithm yields a ratio of $1 - e^{-1}$ to the optimum. Later works presented greedy algorithms that achieve this ratio for other special matroids or for variants of maximum coverage (see, e.g., [1, 17, 26, 5]). For a general matroid constraint, Calinescu et al. showed in [4] that a scheme based on solving a continuous relaxation of the problem followed by *pipage rounding* (a technique introduced by Ageev and Sviridenko [1]) achieves the ratio of $1 - e^{-1}$ for maximizing submodular monotone functions that can be expressed as a sum of weighted rank functions of matroids. Subsequently, this result was extended by Vondrák [27] to general monotone submodular functions.

The bound of $1 - e^{-1}$ is the best possible for all of the above problems. This follows from the lower bound of Nemhauser and Wolsey [23] in the oracle model, and the later result of Feige [9] for the specific case of maximum coverage, under the assumption that $P \neq NP$.

Other variants of monotone submodular optimization were also considered. In [2], Bansal et al. studied the problem of maximizing a monotone submodular function subject to n knapsack constraints, for arbitrary $n \geq 1$, where each element appears in up to k constraints, and k is fixed. The paper presents a $\frac{8ek}{e-1}$ and $\frac{e^2k}{e-1} + o(k)$ approximations for this problem. Demaine and Zadimoghaddam [8] studied bi-criteria approximations for monotone submodular set function optimization.

The problem of maximizing a *non-monotone* submodular function has been studied as well. Feige et al. [11] considered (unconstrained) maximization of a general non-monotone submodular function. The paper gives several (randomized and deterministic) approximation algorithms, as well as hardness results, also for the special case where the function is *symmetric*.

Lee et al. [21] studied the problem of maximizing a general submodular function under linear and matroid constraints. They proposed algorithms that achieve approximation ratio of $1/5 - \varepsilon$ for the problem with d linear constraints and a ratio of $1/(d + 2 + 1/d + \varepsilon)$ for d matroid constraints, for any fixed integer $d \geq 1$.

Improved lower and upper bounds for non-constrained and constrained submodular max-

¹A (weighted) matroid is a system of ‘independent subsets’ of a universe, which satisfies certain *hereditary* and *exchange* properties [25].

imization were recently derived by Gharan and Vondrák [14]. However, this paper does not consider knapsack constraints.

Several fundamental algorithms for submodular maximization (see, e.g., [1, 4, 27, 21]) use a continuous extension of submodular function, to which we refer as *extension by expectation*. Given a submodular function $f : 2^U \rightarrow \mathbb{R}$, we define $F : [0, 1]^U \rightarrow \mathbb{R}$. For any $\bar{y} \in [0, 1]^U$, let $R \subseteq U$ be a random variable such that $i \in R$ with probability y_i (we say that $R \sim \bar{y}$). Then

$$F(\bar{y}) = E[f(R)] = \sum_{R \subseteq U} \left(f(R) \prod_{i \in R} y_i \prod_{i \notin R} (1 - y_i) \right).$$

The general framework of these algorithms is to obtain first a fractional solution for the continuous extension, followed by rounding which yields a solution for the discrete problem.

Using the definition of F , we define the continuous relaxation of our problem called *continuous SUB*. Let $P = \{\bar{y} \in [0, 1]^U \mid \sum_{i \in U} y_i \bar{c}(i) \leq \bar{L}\}$ be the polytope of the instance, then the problem is to find $\bar{y} \in P$ for which $F(\bar{y})$ is maximized. For $\alpha \in (0, 1]$, an algorithm \mathcal{A} yields α -approximation for the continuous problem with respect to a submodular function f , if for any assignment of non-negative costs to the elements, and for any non-negative budget, \mathcal{A} finds a feasible solution for continuous SUB of value at least $\alpha\mathcal{O}$, where \mathcal{O} is the value of an optimal (integral) solution for SUB with the given costs and budget.

For some specific families of submodular functions, linear programming can be used to derive such approximation algorithms (see e.g [1, 4]). For monotone submodular functions, Vondrák presented in [27] a $(1 - e^{-1} - o(1))$ -approximation algorithm for the continuous problem. Subsequently, Lee et al. [21] considered the problem of maximizing *any* submodular function with multiple knapsack constraints and developed a $(\frac{1}{4} - o(1))$ -approximation algorithm for the continuous problem; however, noting that the rounding method of [20],² which proved useful for monotone functions, cannot be applied in the non-monotone case, a $(\frac{1}{5} - \epsilon)$ -approximation was obtained for the discrete problem, by using simple randomized rounding. This gap of approximation ratio between the continuous and the discrete case led us to further develop the technique in [20], so that it can be applied also for non-monotone functions.

1.1 Variants of Maximum Coverage and GAP

Our study of maximizing submodular functions encompasses also a generalization of *maximum coverage* and a budgeted variant of the *generalized assignment problem (GAP)*. These two problems can be cast as submodular optimization problems, however, the resulting universe sizes are non-polynomial in the input size. As our algorithms cannot be directly applied to these problems, more specialized techniques need to be used to obtain approximation algorithm for each of the problems.

The problem of *maximum coverage with multiple packing and cost constraints (MC)* is the following generalization of maximum coverage. Given is a collection of sets $S = \{S_1, \dots, S_m\}$ over a ground set $A = \{a_1, \dots, a_n\}$. Each element a_j has a profit $p_j \geq 0$ and a d_1 -dimensional size vector $\bar{s}_j = (s_{j,1}, \dots, s_{j,d_1})$, such that $s_{j,r} \geq 0$ for all $1 \leq r \leq d_1$. Each set S_i has d_2 -dimensional weight vector $\bar{w}_i = (w_{i,1}, \dots, w_{i,d_2})$. Also, given is a d_1 -dimensional capacity vector $\bar{B} = (B_1, \dots, B_{d_1})$, and a d_2 -dimensional weight bound vector $\bar{W} = (W_1, \dots, W_{d_2})$. A solution for the problem is a collection of subsets $H \subseteq S$ and a subset of elements $\mathcal{E} \subseteq A$, such that for any $a_j \in \mathcal{E}$ there is $S_i \in H$ such that $a_j \in S_i$. A solution is feasible if the total weight

²The paper [20] is a preliminary version of this paper.

of subsets in H is bounded by \bar{W} , and the total size of the elements in \mathcal{E} is bounded by \bar{B} . The profit of a solution (H, \mathcal{E}) is the total profit of the elements in \mathcal{E} . The objective is to find a feasible solution with maximal profit.

MC has a natural application in Video-on-Demand systems. Consider a server which has several resources. Movie files can be stored at the server: this requires purchasing these movies, as well as allocating some storage space.

The system offers video services to a large set of customers. Each customer is willing to pay a certain amount for viewing a movie on her individually chosen list. The transmission of a movie to the customer requires some resources, such as bandwidth and manpower to handle the request. The objective is to select a collection of movies, and a subset of customers to be serviced, such that the total profit is maximized. Indeed, the above problem can be modeled as an instance of MC, where each movie is represented by a set containing all the customers willing to view this movie; the elements to be covered are the customers.

The second problem that we consider is a budgeted variant of GAP. We start with a few definitions. An instance of the *separable assignment problem (SAP)* consists of n items $A = \{a_1, \dots, a_n\}$ and m bins. Each bin i has an associated collection of feasible sets \mathcal{I}_i which is down-closed (i.e., $S \in \mathcal{I}_i$ implies $S' \in \mathcal{I}_i$ for any $S' \subseteq S$). Also, a profit $p_{i,j} \geq 0$ is gained from assigning the item a_j to bin i . The goal is to choose disjoint feasible sets $S_i \in \mathcal{I}_i$ so as to maximize $\sum_{i=1}^m \sum_{a_j \in S_i} p_{i,j}$.

The set of inputs for GAP is the restricted class of inputs for SAP in which the sets \mathcal{I}_i are defined by a knapsack constraint (i.e., the total size of the items in \mathcal{I}_i is bounded by the capacity of bin i). The best approximation for GAP is $(1 - e^{-1} + \varepsilon)$, for some $\varepsilon > 0$, due to [10]. Our algorithm, however, uses the $(1 - e^{-1})$ -approximation for the problem given in [12].

We consider budgeted GAP (BGAP), where each item a_j has a d_1 -dimensional cost vector $\bar{c}_{i,j} \geq 0$, incurred when a_j is assigned to bin i . Also, given is a d_1 -dimensional budget vector \bar{L} . The objective is to find a maximal profit solution whose total cost is at most \bar{L} . A $(\frac{1-e^{-1}}{2-e^{-1}} - \varepsilon)$ -approximation was given in [18] for the case where $d_1 = 1$, as an example for the usage of a Lagrangian relaxation technique. BGAP arises in many real-life scenarios, in particular, in operations research (e.g., inventory planning with delivery costs).

We consider the slightly more general *budgeted linear constrained separable assignment problem (BSAP)*. The difference between BGAP and BSAP is that in the latter the set of feasible assignments for each bin is defined by d_2 knapsack constraints (i.e., the bin capacities are given by d_2 -dimensional vectors), rather than a single constraint.

1.2 Our Results

In this paper we establish a strong relation between the problem of maximizing any submodular function subject to d knapsack constraints and its continuous relaxation. Formally, we show (in Theorem 2.6) that for any non-negative submodular function, an α -approximation algorithm for the continuous relaxation implies a randomized $(\alpha - \varepsilon)$ -approximation algorithm for the discrete problem. We use this relation to obtain approximation ratio of $1/4 - \varepsilon$ for SUB, for any $\varepsilon > 0$, thus improving the best known result for the problem, due to Lee et al. [21]. For the case where the objective function is monotone, we use this relation to obtain a nearly optimal $(1 - e^{-1} - \varepsilon)$ approximation, for any $\varepsilon > 0$. An important consequence of the above relation is that for any class of submodular functions, a future improvement of the approximation ratio for the continuous problem, to a factor of α , immediately implies an approximation ratio of

$(\alpha - \varepsilon)$ for the original instance.

Our technique applies random sampling on the solution space, using a distribution defined by the fractional solution for the problem. In Section 2.5 we show how to convert a feasible solution for the continuous problem to another feasible solution with up to $O(\log |U|)$ fractional entries, given an oracle to the extension by expectation. This facilitates the usage of exhaustive search instead of sampling, which leads to a deterministic version of our technique. Specifically, we obtain a deterministic $(1/4 - \varepsilon)$ -approximation for general instances and $(1 - e^{-1} - \varepsilon)$ -approximation for instances where the submodular function is monotone. For the special case of maximum coverage with d knapsack constraints,³ that is, SUB where the objective function is $f = f_{G,\bar{p}}$ for a given bipartite graph G and profits \bar{p} , this result leads to a deterministic $(1 - e^{-1} - \varepsilon)$ -approximation algorithm, since the extension by expectation of $f_{G,\bar{p}}$ can be deterministically evaluated.

The problem of maximum coverage with multiple packing and cost constraints (MC) can be cast as an instance of SUB, however, the resulting universe size is non-polynomial in the input size. In Section 3 we show how the ideas of Section 2 can be applied also for MC, despite the large size of the universe. The resulting algorithm yields an $(\alpha_\varphi - \varepsilon)$ -approximation for the problem, where φ is the maximal number of sets to which a single element belongs, and $\alpha_\varphi = 1 - \left(1 - \frac{1}{\varphi}\right)^\varphi$. Note that for any $\varphi \geq 1$, $\alpha_\varphi > 1 - e^{-1}$.

Finally, we consider BGAP, which can also be cast as an instance of SUB with non-polynomial size universe. In Section 4 we give a $(1 - e^{-1} - \varepsilon)$ -approximation algorithm for the problem for any $\varepsilon > 0$, based on the ideas of Section 2. In deriving our approximation algorithms for MC and BGAP, we apply our general technique while exploiting special properties of these problems.

Some basic properties of submodular functions are given in Appendix A.

1.3 Recent Developments

Subsequent to our study of maximizing monotone submodular functions subject to multiple knapsack constraints [20], Chekuri et al. [6] showed that, by using a more sophisticated rounding technique, the algorithm in [20] can be applied to derive a $(1 - e^{-1} - \varepsilon)$ -approximation for maximizing a submodular function subject to d knapsack constraints and a matroid constraint. Specifically, given a fractional solution for the problem, the authors define a probability distribution over the solution space, such that all of elements in the domain of the distribution are inside the matroid; these elements also satisfy Chernoff-type concentration bounds, which can be used to prove some of the probabilistic claims in [20]. The desired approximation ratio is obtained by using the algorithm of [20] with sampling replaced by the above distribution in the rounding step. Recently, the same set of authors improved in [7] the bound of $(1/4 - \varepsilon)$ presented here to 0.325.

2 Maximizing Submodular Functions

In this section we describe our framework for maximizing a submodular set function subject to multiple linear constraints. For short, we call this problem SUB.

³This problem is exactly MC with $d_1 = 0$

2.1 Preliminaries

Notation: An essential component in our framework is the distinction between elements by their costs. We say that an element i is *small* if $\bar{c}(i) \leq \varepsilon^3 \bar{L}$; otherwise, the element is *big*.

Given a universe U , we call a subset of elements $S \subseteq U$ *feasible* if the total cost of elements in S is bounded by \bar{L} . We say that S is ε -*nearly feasible* (or *nearly feasible*, if ε is known from the context) if the total cost of the elements in S is bounded by $(1 + \varepsilon)\bar{L}$. We refer to $f(S)$ as the value of S . Similar to the discrete case, $\bar{y} \in [0, 1]^U$ is feasible if $\bar{y} \in P$.

For any subset $T \subseteq U$, we define $f_T : 2^U \rightarrow \mathbb{R}_+$ by $f_T(S) = f(S \cup T) - f(T)$. It is easy to verify that if f is a submodular set function then f_T is also a submodular set function. Finally, for any set $S \subseteq U$, we define $c_r(S) = \sum_{i \in S} c_r(i)$, where $1 \leq r \leq d$, and $\bar{c}(S) = \sum_{i \in S} \bar{c}(i)$. For a fractional solution $\bar{y} \in [0, 1]^U$, we define $c_r(\bar{y}) = \sum_{i \in U} c_r(i) \cdot y_i$ and $\bar{c}(\bar{y}) = \sum_{i \in U} \bar{c}(i) \cdot y_i$.

Overview: Our algorithm consists of two phases, to which we refer as *rounding procedure* and *profit enumeration*. The rounding procedure yields an $(\alpha - O(\varepsilon))$ -approximation for instances in which there are no big elements, using an α -approximate solution for the continuous problem. It relies heavily on Theorem 2.1 that gives some conditions on the probabilistic domain of solutions; these conditions guarantee that the expected profit of the resulting nearly feasible solution is high. This solution is then converted to a feasible one, by using a fixing procedure. We first present a randomized version and later show how to derandomize the rounding procedure.

The profit enumeration phase uses enumeration over the most profitable elements in an optimal solution; then it reduces a general instance to another instance with no big elements, on which we apply the rounding procedure.

Finally, we combine the above results with an algorithm for the continuous problem (e.g., the algorithm of [27], or [21]) to obtain approximation algorithm for SUB.

2.2 A Probabilistic Theorem

We first prove a general probabilistic theorem which refers to a slight generalization of our problem (called *generalized SUB*). In addition to the standard input for the problem, there is also a collection of subsets $\mathcal{M} \subseteq 2^U$, such that if $T \in \mathcal{M}$ and $S \subseteq T$ then $S \in \mathcal{M}$. The goal is to find a subset $S \subseteq \mathcal{M}$, such that $\bar{c}(S) \leq \bar{L}$ and $f(S)$ is maximized.

Theorem 2.1 *For a given input of generalized SUB, let χ be a distribution over \mathcal{M} and D a random variable $D \sim \chi$, such that*

1. $E[f(D)] \geq \mathcal{O}/5$, where \mathcal{O} is an optimal solution for the given instance.
2. For any $1 \leq r \leq d$, $E[c_r(D)] \leq L_r$
3. For any $1 \leq r \leq d$, $c_r(D) = \sum_{k=1}^m c_r(D_k)$, where $D_k \sim \chi_k$ and D_1, \dots, D_m are independent random variables.
4. For any $1 \leq k \leq m$ and $1 \leq r \leq d$, it holds that either $c_r(D_k) \leq \varepsilon^3 L_r$ or $c_r(D_k)$ is fixed.

Let $D' = D$ if D is ε -nearly feasible, and $D' = \emptyset$ otherwise. Then D' is always ε -nearly feasible, $D' \in \mathcal{M}$, and $E[f(D')] \geq (1 - O(\varepsilon))E[f(D)]$.

To prove the results in this section, it suffices to use a special case of Theorem 2.1 (formulated as our next result). We use this theorem in its full generality in developing approximation algorithms for our variants of maximum coverage and GAP (see Sections 3 and 4).

Lemma 2.2 Let $\bar{x} \in [0, 1]^U$ be a feasible fractional solution such that $F(\bar{x}) \geq \mathcal{O}/5$, where \mathcal{O} is the optimal solution for generalized SUB. Let $D \subseteq U$ be a random set such that $D \sim \bar{x}$ (i.e., for all $i \in U$, $i \in D$ with probability x_i), and let D' be a random set such that $D' = D$ if D is ε -nearly feasible, and $D' = \emptyset$ otherwise. Then D' is always ε -nearly feasible, and $E[f(D')] \geq (1 - O(\varepsilon))F(\bar{x})$.

Proof of Theorem 2.1: Define an indicator random variable F such that $F = 1$ if D is ε -nearly feasible, and $F = 0$ otherwise.

Claim 2.1 $Pr[F = 0] \leq d\varepsilon$.

Proof: For any dimension $1 \leq r \leq d$, it holds that $E[c_r(D)] = \sum_{k=1}^m E[c_r(D_k)] \leq L_r$. Define $V_r = \{k | c_r(D_k) \text{ is not fixed}\}$. Then,

$$\begin{aligned} Var[c_r(D)] &= \sum_{k=1}^m Var[c_r(D_k)] \leq \sum_{k \in V_r} E[c_r^2(D_k)] \\ &\leq \sum_{k \in V_r} E[c_r(D_k)] \cdot \varepsilon^3 L_r \leq \varepsilon^3 L_r \sum_{k=1}^m E[c_r(D_k)] \leq \varepsilon^3 L_r^2. \end{aligned}$$

The first inequality holds since $Var[X] \leq E[X^2]$, and the second inequality follows from the fact that $c_r(D_k) \leq \varepsilon^3 L_r$ for $k \in V_r$. Recall that, by the Chebyshev-Cantelli inequality, for any $t > 0$ and a random variable Z ,

$$Pr[Z - E[Z] \geq t] \leq \frac{Var[Z]}{Var[Z] + t^2}.$$

Thus,

$$\begin{aligned} Pr[c_r(D) \geq (1 + \varepsilon)L_r] &= Pr[c_r(D) - E[c_r(D)] \geq (1 + \varepsilon)L_r - E[c_r(D)]] \\ &\leq Pr[c_r(D) - E[c_r(D)] \geq \varepsilon \cdot L_r] \leq \frac{\varepsilon^3 L_r^2}{\varepsilon^2 L_r^2} = \varepsilon. \end{aligned}$$

By the union bound, we have that

$$Pr[F = 0] \leq \sum_{r=1}^d Pr[c_r(D) \geq (1 + \varepsilon)L_r] \leq d\varepsilon.$$

□

For any dimension $1 \leq r \leq d$, let $R_r = \frac{c_r(D)}{L_r}$, and define $R = \max_r R_r$, then R denotes the maximal relative deviation of the cost from the r -th entry in the budget vector, where the maximum is taken over $1 \leq r \leq d$.

Claim 2.2 For any $\ell > 1$,

$$Pr[R > \ell] < \frac{d\varepsilon^3}{(\ell - 1)^2}.$$

Proof: By the Chebyshev-Cantelli inequality we have that, for any dimension $1 \leq r \leq d$,

$$\begin{aligned} Pr[R_r > \ell] &= Pr[c_r(D) > \ell \cdot L_r] \\ &\leq Pr[c_r(D) - E[c_r(D)] > (\ell - 1)L_r] \leq \\ &\leq \frac{\varepsilon^3 L_r^2}{(\ell - 1)^2 L_r^2} \leq \frac{\varepsilon^3}{(\ell - 1)^2}, \end{aligned}$$

and by the union bound, we get that

$$Pr[R > \ell] \leq \frac{d\varepsilon^3}{(\ell - 1)^2}.$$

□

Claim 2.3 For any integer $\ell > 1$, if $R \leq \ell$ then

$$f(D) \leq 2d\ell \cdot \mathcal{O}.$$

Proof: The set D can be partitioned to $2d\ell$ sets $D_1, \dots, D_{2d\ell}$ such that each of these sets is a feasible solution. Hence, $f(D_i) \leq \mathcal{O}$. By Lemma A.1, we have that $f(D) \leq f(D_1) + \dots + f(D_{2d\ell}) \leq 2d\ell\mathcal{O}$. □

Combining the above results we have

Claim 2.4 $E[f(D')] \geq (1 - O(\varepsilon))E[f(D)]$.

Proof: By Claims 2.1 and 2.2, we have that

$$\begin{aligned} E[f(D)] &= E[f(D) | F = 1] \cdot Pr[F = 1] + E[f(D) | F = 0 \wedge (R < 2)] \cdot Pr[F = 0 \wedge (R < 2)] \\ &+ \sum_{\ell \geq 1} E[f(D) | F = 0 \wedge (2^\ell \leq R < 2^{\ell+1})] \cdot Pr[F = 0 \wedge (2^\ell \leq R < 2^{\ell+1})] \\ &\leq E[f(D) | F = 1] \cdot Pr[F = 1] + 4d^2\varepsilon \cdot \mathcal{O} + d^2\varepsilon^3 \cdot \mathcal{O} \cdot \sum_{\ell \geq 1} \frac{2^{\ell+2}}{(2^{\ell-1})^2}. \end{aligned}$$

Since the last summation is a constant, and $E[f(D)] \geq \mathcal{O}/2$, we have that

$$E[F(D)] \leq E[f(D) | F = 1] Pr[F = 1] + \varepsilon \cdot c \cdot E[F(D)],$$

where $c > 0$ is some constant. It follows that

$$(1 - O(\varepsilon))E[f(D)] \leq E[f(D) | F = 1] \cdot Pr[F = 1].$$

Finally, since $D' = D$ if $F = 1$ and $D' = 0$ otherwise, we have that

$$E[f(D')] = E[f(D) | F = 1] \cdot Pr[F = 1] \geq (1 - O(\varepsilon))E[f(D)].$$

□

By definition, D' is always ε -nearly feasible, and $D' \in \mathcal{M}$. This completes the proof of the theorem. □

2.3 Rounding Instances with No Big Elements

In this section we present an $(\alpha - O(\varepsilon))$ -approximation algorithm for SUB inputs with no big elements, given an α -approximate solution for the continuous problem. Inputs with no big elements are easier to tackle. Indeed, any nearly feasible solution for such input can be converted to a feasible one, with only a small harm to the total value.

Lemma 2.3 *Let $S \subseteq U$ be an ε -nearly feasible solution with no big elements, then S can be converted in polynomial time to a feasible solution $S' \subseteq S$, such that $f(S') \geq (1 - O(\varepsilon))f(S)$.*

Proof: In fixing the solution S we handle each dimension separately. For any dimension $1 \leq r \leq d$, if $c_r(S) \leq L_r$ then no modification is needed; otherwise, $c_r(S) > L_r$. Since all elements in S are small, we can partition S into ℓ disjoint subsets S_1, S_2, \dots, S_ℓ such that $\varepsilon L_r \leq c_r(S_j) < (\varepsilon + \varepsilon^3)L_r$ for any $1 \leq j \leq \ell$, where $\ell = \Omega(\varepsilon^{-1})$. Since the function f is submodular, by Lemma A.3, we have that $f(S) \geq \sum_{j=1}^{\ell} f_{S \setminus S_j}(S_j)$. Hence, there exists a value $j \in \{1, 2, \dots, \ell\}$ such that $f_{S \setminus S_j}(S_j) \leq \frac{f(S)}{\ell} = f(S) \cdot O(\varepsilon)$ (note that $f_{S \setminus S_j}(S_j)$ may be negative). Now, $c_r(S \setminus S_j) \leq L_r$, and $f(S \setminus S_j) \geq (1 - O(\varepsilon))f(S)$. We repeat this step for all $1 \leq r \leq d$ to obtain a feasible set S' satisfying $f(S') \geq (1 - O(\varepsilon))f(S)$. \square

Combined with Theorem 2.1, we have the following rounding algorithm.

Randomized Rounding Algorithm for SUB with No Big Elements

Input: A SUB instance, a feasible solution \bar{x} for the continuous problem, with $F(\bar{x}) \geq O/5$.

1. Define a random set $D \sim \bar{x}$. Let $D' = D$ if D is ε -nearly feasible, and $D' = \emptyset$ otherwise.
2. Convert D' to a feasible set D'' as in the proof of Lemma 2.3 and return D'' .

Clearly, the algorithm returns a feasible solution for the problem. By Theorem 2.1, $E[f(D')] \geq (1 - O(\varepsilon))F(\bar{x})$. By Lemma 2.3, $E[f(D'')] \geq (1 - O(\varepsilon))F(\bar{x})$. Hence, we have

Lemma 2.4 *For any instance of SUB with no big elements, any feasible solution \bar{x} for the continuous problem with $F(\bar{x}) \geq O/5$ can be converted to a feasible solution for SUB in polynomial running time with expected profit at least $(1 - O(\varepsilon)) \cdot F(\bar{x})$.*

2.4 Approximation Algorithm for SUB

Given an instance of SUB and a subset $T \subseteq U$, define another instance of SUB, to which we refer as the *residual problem with respect to T* , with f remaining the objective function. The budget for the residual problem is $\bar{L}' = \bar{L} - \bar{c}(T)$, and the universe U' consists of all elements $i \in U \setminus T$ such that $\bar{c}(i) \leq \varepsilon^3 \bar{L}'$, and all elements in T . Formally,

$$U' = T \cup \{i \in U \setminus T \mid \bar{c}(i) \leq \varepsilon^3 \bar{L}'\}.$$

The new cost of element i is $c'(i) = c(i)$ for any $i \in U' \setminus T$, and $c'(i) = 0$ for any $i \in T$. It follows that there are no big elements in the residual problem. Let S be a feasible solution for the residual problem with respect to T . Then $\bar{c}(S) \leq c'(S) + \bar{c}(T) \leq \bar{L}' + \bar{c}(T) = \bar{L}$. Thus, any feasible solution for the residual problem is also feasible for the original instance.

Consider the following algorithm.

Approximation Algorithm for SUB

Input: A SUB instance and an α -approximation algorithm \mathcal{A} for the continuous problem with respect to the function f .

1. For any $T \subseteq U$ such that $|T| \leq h = \lceil d \cdot \varepsilon^{-4} \rceil$
 - (a) Use \mathcal{A} to obtain an α -approximate solution \bar{x} for the continuous residual problem with respect to T .
 - (b) Use the Randomized Rounding Algorithm of Section 2.3 to convert \bar{x} to a feasible solution S for the residual problem.
2. Return the best solution found.

Lemma 2.5 *The above approximation algorithm returns an $(\alpha - O(\varepsilon))$ -approximate solution for SUB and uses a polynomial number of calls to algorithm \mathcal{A} .*

Proof: By Lemma 2.4, in each iteration the algorithm finds a feasible solution S for the residual problem. Hence, the algorithm always returns a feasible solution for the given SUB instance.

Let $\mathcal{O} = \{i_1, \dots, i_k\}$ be an optimal solution for the input I (we use \mathcal{O} to denote both an optimal sub-collection of elements and the optimal value). For $\ell \geq 1$, let $K_\ell = \{i_1, \dots, i_\ell\}$, and assume that the elements are ordered by their residual profits, i.e., $i_\ell = \operatorname{argmax}_{i \in \mathcal{O} \setminus K_{\ell-1}} f_{K_{\ell-1}}(\{i\})$.

Consider the iteration in which $T = K_h$, and define $\mathcal{O}' = \mathcal{O} \cap T$. The set \mathcal{O}' is clearly a feasible solution for the residual problem with respect to T . We show a lower bound for $f(\mathcal{O}')$. The set $R = \mathcal{O} \setminus \mathcal{O}'$ consists of elements in $\mathcal{O} \setminus T$ that are big with respect to the residual instance. The total cost of elements in R is bounded by \bar{L}' (since \mathcal{O} is a feasible solution), and thus $|R| \leq \varepsilon^{-3} \cdot d$.

Since $T = K_h$, for any $j \in \mathcal{O} \setminus T$ it holds that $f_T(j) \leq \frac{f(T)}{|T|}$, and we get $f_T(R) \leq \sum_{j \in R} f_T(\{j\}) \leq \varepsilon^{-3} \cdot d \frac{f(T)}{|T|} = \varepsilon f(T) \leq \varepsilon \mathcal{O}$. Thus, $f_{\mathcal{O}'}(R) \leq f_T(R) \leq \varepsilon \mathcal{O}$. Since $f(\mathcal{O}) = f(\mathcal{O}') + f_{\mathcal{O}'}(R) \leq f(\mathcal{O}') + \varepsilon f(\mathcal{O})$, we have that $f(\mathcal{O}') \geq (1 - \varepsilon)f(\mathcal{O})$.

Thus, in this iteration we get a solution \bar{x} for the residual problem with $F(\bar{x}) \geq \alpha(1 - \varepsilon)f(\mathcal{O})$, and the solution S obtained after the rounding satisfies $f(S) \geq (1 - O(\varepsilon))\alpha f(\mathcal{O})$. \square

We summarize in the next result.

Theorem 2.6 *Let f be a submodular function, and suppose there is a polynomial time α -approximation algorithm for the continuous problem with respect to f . Then there is a polynomial time randomized $(\alpha - \varepsilon)$ -approximation algorithm for SUB with respect to f , for any $\varepsilon > 0$.*

Since there is a $(1/4 - o(1))$ -approximation algorithm for general instances of continuous SUB [21], we have

Theorem 2.7 *There is a polynomial time randomized $(1/4 - \varepsilon)$ -approximation algorithm for SUB, for any $\varepsilon > 0$.*

Since there is a $(1 - e^{-1} - o(1))$ approximation algorithm for SUB with monotone objective function [27] we have

Theorem 2.8 *There is a polynomial time randomized $(1 - e^{-1} - \varepsilon)$ -approximation algorithm for SUB with monotone objective function, for any $\varepsilon > 0$.*

2.5 Derandomization

In this section we show how the algorithm of Section 2.3 can be derandomized, assuming we have an oracle for F , the extension by expectation of f . For some families of submodular

functions, F can be directly evaluated; for a general function f , F can be evaluated with high accuracy by sampling f , as in [27].

The main idea is to reduce the number of fractional entries in the fractional solution \bar{x} , so that the number of values a random set $D \sim \bar{x}$ can get is polynomial in the input size (for a fixed value of ε). Then, we go over all the possible values, and we are promised to obtain a solution of high value.

A key tool in our derandomization is the *pipage rounding* technique of Ageev and Sviridenko [1]. We give below a brief overview of the technique. For any element $i \in U$, define the unit vector $\bar{i} \in \{0, 1\}^U$, in which $i_j = 0$ for any $j \neq i$, and $i_i = 1$. Given a fractional solution \bar{x} for the problem and two elements i, j , such that x_i and x_j are both fractional, consider the vector function $\bar{x}_{i,j}(\delta) = \bar{x} + \delta\bar{i} - \delta\bar{j}$ (Note that $\bar{x}_{i,j}(\delta)$ is equal to \bar{x} in all entries except i, j). Let $\delta_{\bar{x},i,j}^+$ and $\delta_{\bar{x},i,j}^-$ (for short, δ^+ and δ^-) be the maximal and minimal value of δ for which $\bar{x}_{i,j}(\delta) \in [0, 1]^U$. In both $\bar{x}_{i,j}(\delta^+), \bar{x}_{i,j}(\delta^-)$, the entry of either i or j is integral.

Define $F_{i,j}^{\bar{x}}(\delta) = F(\bar{x}_{i,j}(\delta))$ over the domain $[\delta^-, \delta^+]$. The function $F_{i,j}^{\bar{x}}$ is convex (see [3] for a detailed proof), thus $\bar{x}' = \operatorname{argmax}_{\{\bar{x}_{i,j}(\delta^+), \bar{x}_{i,j}(\delta^-)\}} F(\bar{x})$ has fewer fractional entries than \bar{x} , and $F(\bar{x}') \geq F(\bar{x})$. By appropriate selection of i, j , such that \bar{x}' maintains feasibility (in some sense), we can repeat the above step to gradually decrease the number of fractional entries. We use the technique to prove the next result.

Lemma 2.9 *Let $\bar{x} \in [0, 1]^U$ be a solution having k or less fractional entries (i.e., $|\{i \mid 0 < x_i < 1\}| \leq k$), and $\bar{c}(\bar{x}) \leq \bar{L}$ for some \bar{L} . Then \bar{x} can be converted to a vector \bar{x}' with at most $k' = \left(\frac{8 \ln(2k)}{\varepsilon}\right)^d$ fractional entries, such that $\bar{c}(\bar{x}') \leq (1 + \varepsilon)\bar{L}$, and $F(\bar{x}') \geq F(\bar{x})$, in time polynomial in k .*

Proof: Let $U' = \{i \mid 0 < x_i < 1\}$ be the set of all fractional entries. We define a new cost function \bar{c}' over the elements in U .

$$c'_r(i) = \begin{cases} c_r(i) & i \notin U' \\ 0 & c_r(i) \leq \frac{\varepsilon \cdot L_r}{2k} \\ \frac{\varepsilon \cdot L_r}{2k} (1 + \varepsilon/2)^j & \frac{\varepsilon \cdot L_r}{2k} (1 + \varepsilon/2)^j \leq c_r(i) < \frac{\varepsilon \cdot L_r}{2k} (1 + \varepsilon/2)^{j+1} \end{cases}$$

Note that for any $i \in U'$, $\bar{c}'(i) \leq \bar{c}(i)$, and

$$c_r(i) \leq \left(1 + \frac{\varepsilon}{2}\right) c'_r(i) + \frac{\varepsilon \cdot L_r}{2k},$$

for all $1 \leq r \leq d$. The number of different values $c'_r(i)$ can get for $i \in U'$ is bounded by $\frac{8 \ln(2k)}{\varepsilon}$ (since all elements are small, and $\ln(1 + x) \geq x/2$). Hence the number of different values $\bar{c}'(i)$ can get for $i \in U'$ is bounded by $k' = \left(\frac{8 \ln(2k)}{\varepsilon}\right)^d$.

We start with $\bar{x}' = \bar{x}$, and while there are $i, j \in U'$ such that x'_i and x'_j are both fractional and $\bar{c}'(i) = \bar{c}'(j)$, define $\delta^+ = \delta_{\bar{x}',i,j}^+$ and $\delta^- = \delta_{\bar{x}',i,j}^-$. Since i and j have the same cost (by \bar{c}'), it holds that $\bar{c}'(\bar{x}_{i,j}(\delta^+)) = \bar{c}'(\bar{x}_{i,j}(\delta^-)) = \bar{c}'(\bar{x})$. If $F_{i,j}^{\bar{x}'}(\delta^+) \geq F(\bar{x})$, then set $\bar{x}'' = \bar{x}_{i,j}(\delta^+)$, otherwise $\bar{x}'' = \bar{x}_{i,j}(\delta^-)$. In both cases $F(\bar{x}'') \geq F(\bar{x}')$ and $\bar{c}'(\bar{x}'') = \bar{c}'(\bar{x}')$. Now, repeat this step with $\bar{x}' = \bar{x}''$. Since in each iteration the number of fractional entries in \bar{x}' decreases, the process will terminate (after at most k iterations) with a vector \bar{x}' such that $F(\bar{x}') \geq F(\bar{x})$, $\bar{c}'(\bar{x}') = \bar{c}'(\bar{x}) \leq \bar{L}$, and there are no two elements $i, j \in U'$ with $\bar{c}'(i) = \bar{c}'(j)$, where x'_i and x'_j

are both fractional. Also, for any $i \notin U'$, the entry x'_i is integral (since x_i was integral and the entry was not modified by the process). Thus, the number of fractional entries in \bar{x}' is at most k' . Now, for any dimension $1 \leq r \leq d$,

$$\begin{aligned} c_r(\bar{x}') &= \sum_{i \notin U'} x'_i c_r(i) + \sum_{i \in U'} x'_i c_r(i) \\ &\leq (1 + \varepsilon/2) \cdot \sum_{i \notin U'} x'_i \cdot c'_r(i) + \sum_{i \in U'} x'_i \left((1 + \varepsilon/2) c'_r(i) + \frac{\varepsilon \cdot L_r}{2k} \right) \\ &= (1 + \varepsilon/2) \cdot \sum_{i \in U} x'_i \cdot c'_r(i) + \sum_{i \in U'} x_i \frac{\varepsilon \cdot L_r}{2k} \leq (1 + \varepsilon) L_r. \end{aligned}$$

This completes the proof. \square

Using the above lemma, we can reduce the number of fractional entries in \bar{x} to a number that is poly-logarithmic in k . However, the number of values $D \sim \bar{x}$ remains super-polynomial. To reduce further the number of fractional entries, we apply the above step twice, that is, we convert \bar{x} with at most $|U|$ fractional entries to \bar{x}' with at most $k' = (8 \ln(2|U|)/\varepsilon)^d$. We can then apply the conversion again, to obtain \bar{x}'' with at most $k'' = O(\log |U|)$ fractional entries.

Lemma 2.10 *Given a vector \bar{L} and a constant $\varepsilon > 0$, let $\bar{x} \in [0, 1]^U$ be a vector satisfying $\bar{c}(\bar{x}) \leq \bar{L}$. Then \bar{x} can be converted in time polynomial in $|U|$ to a vector \bar{x}' with at most $k'' = O(\log |U|)$ fractional entries, such that $\bar{c}(\bar{x}') \leq (1 + \varepsilon)^2 \bar{L}$, and $F(\bar{x}') \geq F(\bar{x})$,*

The next result follows immediately from Lemma 2.2 (\mathcal{O} is the value of an optimal solution for SUB).

Lemma 2.11 *Given $\bar{x} \in [0, 1]^U$ such that \bar{x} is a feasible fractional solution with $F(\bar{x}) \geq \mathcal{O}/5$, there exists a realization of the random variable $D \sim \bar{x}$, such that the solution \mathcal{D} is nearly feasible, and $F(\mathcal{D}) \geq (1 - O(\varepsilon))F(\bar{x})$.*

Consider the following rounding algorithm.

Deterministic Rounding Algorithm for SUB with No Big Elements

Input: A SUB instance, a feasible solution \bar{x} for the continuous problem, with $F(\bar{x}) \geq \mathcal{O}/5$.

1. Define $\bar{x}' = (1 + \varepsilon)^{-2} \cdot \bar{x}$ (note that $F(\bar{x}') \geq (1 + \varepsilon)^{-2} \cdot F(\bar{x})$).
2. Convert \bar{x}' to \bar{x}'' such that \bar{x}'' is fractionally feasible, the number of fractional entries in \bar{x}'' is $O(\log |U|)$, and $F(\bar{x}) \geq (1 + \varepsilon)^{-2} F(\bar{x}'') \geq (1 - e^{-1} - O(\varepsilon))\mathcal{O}$, as in Lemma 2.10.
3. Enumerate over all possible realizations of $D \sim \bar{x}''$. For each such realization, if the solution \mathcal{D} is ε -nearly feasible convert it to a feasible solution \mathcal{D}' (see Lemma 2.3). Return the solution with maximum value among the feasible solutions found.

By Theorem 2.1, the algorithm returns a feasible solution of value at least $(1 - O(\varepsilon))F(\bar{x})$. Also, the running time of the algorithm is polynomial when ε is a fixed constant. Replacing the randomized rounding step in the algorithm of Section 2.4 with the above Deterministic Rounding Algorithm, we get the following result.

Theorem 2.12 *Let f be a submodular function, and assume we have an oracle for F . If there is a deterministic polynomial time α -approximation algorithm for the continuous problem with respect to f , then there is a polynomial time deterministic $(\alpha - \varepsilon)$ -approximation algorithm for SUB with respect to f , for any $\varepsilon > 0$.*

We note that, given an oracle to F , both the algorithms of [27] and [21] for the continuous problem are deterministic, thus we get the following.

Theorem 2.13 *Given an oracle for F , there is a polynomial time deterministic $(1 - e^{-1} - \varepsilon)$ -approximation algorithm for SUB with a monotone function, for any $\varepsilon > 0$.*

Theorem 2.14 *Given an oracle for F , there is a polynomial time deterministic $(1/4 - \varepsilon)$ -approximation algorithm for SUB for any $\varepsilon > 0$.*

For the problem of maximum coverage with d knapsack constraints, i.e., SUB where the objective function is $f = f_{G, \bar{p}}$, for a given bipartite graph G and profits \bar{p} , the function F can be evaluated deterministically (see [1]). This yields the following result.

Theorem 2.15 *There is a polynomial time deterministic $(1 - e^{-1} - \varepsilon)$ -approximation algorithm for maximum coverage with d knapsack constraints.*

3 Maximum Coverage with Multiple Packing and Cost Constraints

In this section we consider the problem of *maximum coverage with multiple packing and cost constraints (MC)*. Let φ denote the maximal number of sets to which an element belongs, and let

$$\alpha_\varphi = 1 - \left(1 - \frac{1}{\varphi}\right)^\varphi. \quad (1)$$

We give below an $(\alpha_\varphi - \varepsilon)$ -approximation algorithm for the problem. We note that, for any $\varphi \geq 1$, $\alpha_\varphi > 1 - e^{-1}$. In solving MC, our algorithm uses the following continuous version of the problem. Let $\bar{y} \in [0, 1]^{S \times A}$ and $\bar{x} \in [0, 1]^S$. For short, we write $y_{i,j} = y_{S_i, a_j}$ and $x_i = x_{S_i}$. Given an input for MC, we say that (\bar{y}, \bar{x}) is a solution if, for any $S_i \in S$ and $a_j \notin S_i$ it holds that $y_{i,j} = 0$ (for short, we write $y_{i,j} = y_{S_i, a_j}$), and for any $S_i \in S$ and $a_j \in A$ it holds that $y_{i,j} \leq x_i$. Intuitively, x_i is an indicator for the selection of the set S_i into the solution, and $y_{i,j}$ is an indicator for the selection of the element a_j by the set S_i into the solution. We say that such a solution is feasible if, for any $1 \leq r \leq d_1$ it holds that $\sum_{a_j \in A} s_{j,r} \cdot \sum_{S_i \in S} y_{i,j} \leq B_r$ (the total size of elements does not exceed the capacity), and for any $1 \leq r \leq d_2$ it holds that $\sum_{S_i \in S} x_i \cdot w_{i,r} \leq W_r$ (the total weight of subsets does not exceed the weight bound). The value (or profit) of the solution is defined by $p(\bar{y}, \bar{x}) = p(\bar{y}) = \sum_{a_j \in A} \min\{1, \sum_{S_i \in S} y_{i,j}\} \cdot p_j$. By the above definition, a solution consists of fractional values. We say that a solution (\bar{x}, \bar{y}) is *semi-fractional* if $\bar{x} \in \{0, 1\}^S$ (that is, sets cannot be fractionally selected, but elements can be). Also, we say that a solution is *integral* if both $\bar{x} \in \{0, 1\}^S$ and $\bar{y} \in \{0, 1\}^{S \times A}$.

Two computational problems arise from the above definitions. The first is to find a semi-fractional solution of maximal profit. We refer to this problem as the *semi-fractional problem*. The second is to find an integral solution of maximal profit, to which we refer as the *integral problem*. It is easy to see that the integral problem is equivalent to MC, therefore our objective is to find an optimal solution for the integral problem.

Overview: To obtain an approximation algorithm for the integral problem, we first show how it relates to the semi-fractional problem. In particular, we show that, given an $(\alpha_\varphi - O(\varepsilon))$ -approximation algorithm for the semi-fractional problem, we can derive approximation algorithm with the same approximation ratio for the integral problem. Next, we interpret the semi-fractional problem as a SUB instance whose universe has infinite size. We then use the

framework developed in Section 2 to solve this problem. As direct enumeration over the most profitable elements in an optimal solution is impossible here, we guess which sets are the most profitable in an optimal solution. We use this guessing to obtain a fractional solution (with polynomial number of non-zero entries), such that the conditions of Theorem 2.1 are satisfied. Together with a fixing procedure, applied to the resulting nearly feasible solution, this leads to our approximation algorithm. The process can be derandomized by using the same tools as in Section 2.5.

3.1 Reduction to the Semi-fractional Problem

We first show that any semi-fractional solution for the problem can be converted to a solution with at least the same profit and at most d_1 fractional entries. Next, we show how this property enables to enumerate over the most profitable elements in an optimal solution. Throughout this section we assume that, for some constant $\alpha \in (0, 1)$, we have an α -approximation algorithm for the semi-fractional problem.

Lemma 3.1 *Let (\bar{y}^f, \bar{x}^f) be a feasible semi-fractional solution. Then \bar{y}^f can be converted in polynomial time to another feasible semi-fractional solution (\bar{y}, \bar{x}^f) with at most d_1 fractional entries, such that $p(\bar{y}) \geq p(\bar{y}^f)$.*

Proof: Let (\bar{y}^f, \bar{x}^f) be a semi-fractional feasible solution. W.l.o.g, we assume that, for any $a_j \in A$, $\sum_{S_i \in S} y_{i,j}^f \leq 1$, and if $y_{i,j}^f \neq 0$ then for any $S_{i'} \neq S_i$ it holds that $y_{i',j}^f = 0$. Note that any solution can be easily converted to such a solution having the same profit. If there are more than d_1 fractional entries, let $\bar{s}_{j_1}, \dots, \bar{s}_{j_k}$ be the size vectors of the corresponding elements, and let $S_{i_1} \dots S_{i_k}$ be the corresponding sets. Since $k > d_1$, there must be a linear dependency between the size vectors. W.l.o.g we can write $\lambda_1 \bar{s}_{j_1} + \dots + \lambda_p \bar{s}_{j_p} = 0$ for $p = d_1 + 1$. We can define $\bar{y}^f(\varepsilon)$ by $y_{i_\ell, j_\ell}^f(\varepsilon) = y_{i_\ell, j_\ell}^f + \varepsilon \lambda_\ell$ for $1 \leq \ell \leq p$, and $y_{i,j}^f(\varepsilon) = y_{i,j}^f$ for any other entry. As long as $\bar{y}^f(\varepsilon) \in [0, 1]^{S \times A}$, $\bar{y}^f(\varepsilon)$ is a semi-fractional feasible solution. Let ε^+ and ε^- be the maximal and minimal values of ε for which $\bar{y}^f(\varepsilon) \in [0, 1]^{S \times A}$. The number of fractional entries in $\bar{y}^f(\varepsilon^+)$ and $\bar{y}^f(\varepsilon^-)$ is smaller than the number of fractional entries in \bar{y}^f . Also, $p(\varepsilon) = p(\bar{y}^f(\varepsilon))$ is a linear function, thus either $p(\bar{y}^f(\varepsilon^+)) \geq p(\bar{y}^f)$ or $p(\bar{y}^f(\varepsilon^-)) \geq p(\bar{y}^f)$. Thus, we can convert \bar{y}^f to a feasible solution \bar{y}' that has less fractional entries, such that $p(\bar{y}') \geq p(\bar{y}^f)$. By repeating the above process we can obtain a fractional solution with at most d_1 fractional entries. \square

We use Lemma 3.1 to prove the next result.

Lemma 3.2 *Given an α -approximation algorithm for the semi-fractional problem, an α -approximation algorithm for the integral problem can be derived in polynomial time.*

Proof: Given a collection T of pairs (a_j, S_i) of an element a_j and a set S_i , such that $a_j \in S_i$, denote the collection of sets in T by T_S , and the collection of elements in T by $T_{\mathcal{E}}$. We define a residual instance for the problem as follows. The elements are:

$$A_T = \{a_j \in A \mid a_j \notin T_{\mathcal{E}}, \text{ for any } a_{j'} \in T_{\mathcal{E}}, p_j \leq p_{j'} \},$$

where the size of $a_j \in A_T$ is \bar{s}_j , and the profit of a_j is p_j . The sets are $S_T = \{S'_1, \dots, S'_m\}$, where $S'_i = S_i \cap A_T$, and

$$\bar{w}'_i = \begin{cases} \bar{w}_i & S_i \notin T_S \\ 0 & \text{otherwise} \end{cases}$$

The weight bound of the residual instance is $\bar{W}_T = \bar{W} - w(T_S)$, where $w(T_S) = \sum_{S_i \in T_S} \bar{w}_i$, and the capacity is $\bar{B}_T = \bar{B} - s(T_{\mathcal{E}})$, where $s(T_{\mathcal{E}}) = \sum_{a_j \in T_{\mathcal{E}}} \bar{s}_j$.

Clearly, a solution of profit v for the residual instance with respect to a collection T gives a solution of profit $v + p(T_{\mathcal{E}})$ for the original instance, where $p(T_{\mathcal{E}}) = \sum_{a_j \in T_{\mathcal{E}}} p_j$. Let $\mathcal{O} = (\bar{x}, \bar{y})$ be an optimal solution for the integral problem. W.l.o.g. we assume that for all $a_j \in A$, $\sum_{S_i \in \mathcal{S}} y_{i,j} \leq 1$ (that is, no element is selected by more than one set). Let R be the collection of $h = \frac{d_1}{1-\alpha}$ most profitable elements a_j for which there exists S_i such that $y_{i,j} = 1$ (note that there is a unique set S_i for each a_j). Define $T^{\mathcal{O}} = \{(a_j, S_i) | a_j \in R \wedge y_{i,j} = 1\}$. It is easy to verify that the optimal integral solution for the residual problem with respect to $T^{\mathcal{O}}$ is $\mathcal{O} - p(T_{\mathcal{E}}^{\mathcal{O}})$.

Now, assume that we have an α -approximation algorithm for the semi-fractional problem. Then the algorithm returns a fractional solution (\bar{y}^f, \bar{x}^f) with $p(\bar{y}^f) \geq \alpha(\mathcal{O} - p(T_{\mathcal{E}}^{\mathcal{O}}))$. By Lemma 3.1, this solution can be converted to a solution (\bar{z}, \bar{x}^f) with up to d_1 fractional entries satisfying $p(\bar{z}) \geq p(\bar{y}^f)$. Now, consider rounding down to zero the value of each fractional entry in \bar{z}' . This results in a new feasible integral solution \bar{z}'' with $p(\bar{z}'') \geq p(\bar{y}^f) - \frac{d_1 \cdot p(T_{\mathcal{E}}^{\mathcal{O}})}{|T|}$ (since the profit of any element in the residual solution is bounded by $\frac{p(T_{\mathcal{E}}^{\mathcal{O}})}{|T|}$). Hence, we obtain a solution for the integral problem of value at least

$$p(T_{\mathcal{E}}^{\mathcal{O}}) + p(\bar{y}^f) - \frac{d_1 \cdot p(T_{\mathcal{E}}^{\mathcal{O}})}{|T|} \geq p(T_{\mathcal{E}}^{\mathcal{O}}) \left(1 - \frac{d_1}{|T|}\right) + \alpha(\mathcal{O} - p(T_{\mathcal{E}}^{\mathcal{O}})) \geq \alpha\mathcal{O}.$$

Thus, we have an α -approximation for the optimum of the integral problem. To apply this technique, we need to guess the correct set T , which can be done in time $(n \cdot m)^{O(1)}$ for a constant α . \square

In Theorem 3.7 we show that there is a polynomial time $(\alpha_{\varphi} - \varepsilon)$ -approximation algorithm for the semi-fractional problem, where α_{φ} is defined in (1). Thus, we have the following.

Theorem 3.3 *There is a polynomial time $(\alpha_{\varphi} - \varepsilon)$ -approximation algorithm for MC for any $\varepsilon > 0$, where φ is the maximal number of sets to which a single element belongs.*

3.2 Solving the Semi-fractional Problem

3.2.1 A Submodular Point of View

Let $\tilde{\mathcal{O}} = (\bar{y}, \bar{x})$ be an optimal solution for an instance of the semi-fractional problem. W.l.o.g., we may assume that for any element $a_j \in A$, $\sum_{S_i \in \mathcal{S}} y_{i,j} \leq 1$. For any $S_i \in \mathcal{S}$, we define the profit of S_i with respect to the solution $\tilde{\mathcal{O}}$ by $p_{\tilde{\mathcal{O}}}(S_i) = p(S_i) = \sum_{a_j \in A} y_{i,j} p_j$ (note that if $x_i = 0$ then $p(S_i) = 0$). Since $\sum_{S_i \in \mathcal{S}} y_{i,j} \leq 1$ holds for any $a_j \in A$, this means that $p(\tilde{\mathcal{O}}) = \sum_{S_i \in \mathcal{S}} p_{\tilde{\mathcal{O}}}(S_i)$.

Given T , the collection of the $h = \varepsilon^{-4} \cdot d$ most profitable sets in $\tilde{\mathcal{O}}$, we define a maximization problem for a submodular function, with $d = d_1 + d_2$ linear constraints. Let $\bar{W}' = \bar{W} - w(T)$ (we guess the set T).

- For each $S_i \in T$ and any vector $\bar{z} \in [0, 1]^A$ satisfying: $z_j = 0$ for all $a_j \notin S_i$, add the element (S_i, \bar{z}) to the universe U . The cost of this element is \bar{c} , where $c_r = \sum_{a_j \in A} z_j \cdot s_{i,j}$ for $1 \leq r \leq d_1$, and $c_r = 0$ for $d_1 + 1 \leq r \leq d_1 + d_2 = d$.
- For each set $S_i \notin T$ and a vector $\bar{z} \in [0, 1]^A$ satisfying

1. $z_j = 0$ for all $a_j \notin S_i$,

2. for any $1 \leq r \leq d_1$ it holds that $\sum_{a_j \in A} z_j \cdot s_{j,r} \leq \varepsilon^3 B_r$, and

3. for any $1 \leq r \leq d_2$ it holds that $w_{i,r} \leq \varepsilon^3 W'_r$,

add (S_i, \bar{z}) to U . The cost of this element is \bar{c} , where $c_r = \sum_{a_j \in A} z_j \cdot s_{j,r}$ for $1 \leq r \leq d_1$, and $c_{d_1+r} = w_{i,r}$ for $1 \leq r \leq d_2$.

The budget vector \bar{L} for this instance is the vector \bar{B} concatenated to \bar{W}' , that is, $L_r = B_r$ for $1 \leq r \leq d_1$, and $L_{d_1+r} = W'_r$ for $1 \leq r \leq d_2$. Define $f_j : 2^U \rightarrow \mathbb{R}$ for any $a_j \in A$, by

$$f_j(V) = \min\left\{ \sum_{(S_i, \bar{z}) \in V} z_j, 1 \right\}, \quad (2)$$

and $f : 2^U \rightarrow \mathbb{R}$ by $f(V) = \sum_{a_j \in A} f_j(V) \cdot p_j$. Since each f_j is a submodular non-decreasing set function, f is a submodular non-decreasing set function as well. It is easy to see that a solution of value v for the above instance of SUB yields a solution of the same value (profit) for the semi-fractional problem.

Let the size of a set $S_i \in S \setminus T$ (with respect to the solution \tilde{O}) be $s(S_i) = \sum_{a_j \in A} y_{i,j} \cdot \bar{s}_j$. We say that a set $S_i \in S \setminus T$ is *small* if $s(S_i) \leq \varepsilon^3 \bar{B}$ and $\bar{w}_i \leq \varepsilon^3 \bar{W}'$; otherwise it is *big*. Consider the following solution for the above SUB instance. For each $S_i \in T$, define the vector \bar{z} by $z_j = y_{i,j}$, for all $a_j \in A$, and add the element (S_i, \bar{z}) to the solution; for each $S_i \notin T$ such that S_i is *small*, define \bar{z} by $z_j = y_{i,j}$ and add (S_i, \bar{z}) to the solution. Denote the resulting solution by V . It can be easily verified that V is a feasible solution, and it holds that

$$\begin{aligned} f(V) &= \sum_{S_i \in T} p(S_i) + \sum_{S_i \in S \setminus T, S_i \text{ is small}} p(S_i) = \tilde{O} - \sum_{S_i \in S \setminus T, S_i \text{ is big}} p(S_i) \\ &\geq \tilde{O} - \frac{d \cdot \varepsilon^{-3}}{h} \tilde{O} \geq (1 - \varepsilon) \tilde{O}. \end{aligned} \quad (3)$$

The last inequality holds since the number of big sets in \tilde{O} is bounded by $d \cdot \varepsilon^{-3}$, and the profit of each of these sets is bounded by \tilde{O}/h . This implies that the value of the optimal solution for the SUB instance is between $(1 - \varepsilon) \tilde{O}$ and \tilde{O} .

3.2.2 Obtaining a Distribution on the Universe of the Submodular Problem

We now use the technique of Section 2 for solving the SUB instance. To do so, we first need to obtain a fractional feasible solution. As the size of U may be unbounded, we cannot use the algorithm of Vondrák [27]. Thus, we obtain a fractional solution by using a linear programming formulation of the problem. Let S' be the collection of all sets S_i in S such that $S_i \in T$, or $\bar{w}_i \leq \varepsilon^3 \bar{W}'$. Consider the following linear program:

$$\begin{aligned}
(LP(T)) \quad & \text{maximize} && \sum_{S_i \in S'} \sum_{a_j \in A} y_{i,j} \cdot p_j \\
& \text{subject to:} && \\
& \forall S_i \in S', a_j \in A && 0 \leq y_{i,j} \leq x_i \\
& \forall S_i \in S', a_j \notin S_i && y_{i,j} = 0 \\
& \forall a_j \in A && \sum_{S_i \in S'} y_{i,j} \leq 1 \\
& \forall 1 \leq r \leq d_1 && \sum_{S_i \in S'} \sum_{a_j \in A} y_{i,j} \cdot s_{j,r} \leq B_r \\
& \forall 1 \leq r \leq d_2 && \sum_{S_i \in S' \setminus T} w_{i,r} \cdot x_i \leq W'_r \\
& \forall S_i \in T && x_i = 1 \\
& \forall S_i \in S' \setminus T \text{ and } \forall 1 \leq r \leq d_1 && \sum_{a_j \in A} y_{i,j} \cdot s_{j,r} \leq \varepsilon^3 B_r \cdot x_i
\end{aligned}$$

Let \bar{x}^f, \bar{y}^f be an optimal solution for $LP(T)$ of value \mathcal{O}^f . Clearly, \mathcal{O}^f is greater or equal to the optimal solution of the SUB instance; thus, by (3), $\mathcal{O}^f \geq (1 - \varepsilon)\tilde{\mathcal{O}}$. We use this solution to generate fractional solution for the SUB instance. Define $\bar{X} \in [0, 1]^U$ as follows. For any $S_i \in S'$ such that $x_i^f > 0$, let $\bar{z}_i \in [0, 1]^A$, where $z_{i,j} = \frac{y_{i,j}^f}{x_i^f}$, and we set $X_i = X_{(S_i, \bar{z}_i)} = x_i^f$. For any other $u \in U$, set $X_u = 0$. For any $a_j \in A$, define $Y_j = \sum_{S_i \in S'} y_{i,j}^f$, then clearly $\mathcal{O}^f = \sum_{a_j \in A} Y_j \cdot p_j$.

Lemma 3.4 *Let D be a random variable such that $D \sim \bar{X}$, then for any $a_j \in A$, $E[f_j(D)] \geq \alpha_f \cdot Y_j$, where f_j is defined in (2).*

We use in the proof the next claim.

Claim 3.1 *For any $x \in [0, 1]$ and $\varphi \in \mathbb{N}$,*

$$1 - \left(1 - \frac{x}{\varphi}\right)^\varphi \geq x \cdot \alpha_\varphi,$$

where α_φ is defined in (1).

Proof: Let $h(x) = 1 - \left(1 - \frac{x}{\varphi}\right)^\varphi - x \cdot \alpha_\varphi$, then $h(0) = h(1) = 0$. Also, $h''(x) = -\frac{\varphi-1}{\varphi} \left(1 - \frac{x}{\varphi}\right)^{\varphi-2} \leq 0$ for $x \in [0, 1]$. Hence, $h(x) \geq 0$ for $x \in [0, 1]$, and the claim holds. \square

Proof of Lemma 3.4: For the case where $Y_j = 0$ the claim trivially holds, thus we assume below that $Y_j \neq 0$. Let X_i be an indicator random variable for $(S_i, \bar{z}_i) \in D$, for any $S_i \in S'$. The random variables X_i are independent, and $Pr[X_i = 1] = x_i^f$. Then,

$$f_j(D) = \min \left\{ 1, \sum_{(S_i, \bar{z}_i) \in D} z_{i,j} \right\} = \min \left\{ 1, \sum_{S_i \in S', a_j \in S_i} \frac{y_{i,j}^f}{x_i^f} \cdot X_i \right\}$$

Let $S'[j] = \{S_i \in S' \mid a_j \in S_i\}$ and $\tau_j = |S'[j]|$. Let $H \geq 1$ be an integer such that, for any $S_i \in S'[j]$ with $x_i^f \neq 0$, $\frac{y_{i,j}^f}{x_i^f}$ is an integral multiple of $\delta = 1/H$ (assuming all values are rational, such a value of H exists). Let Z_1, \dots, Z_H be a set of indicator random variables used

as follows. Whenever X_i is selected in our random process (i.e., $X_i = 1$), randomly select $\frac{y_{i,j}^f}{x_i^f} \cdot \delta^{-1}$ indicators among Z_1, \dots, Z_H with uniform distribution. For $1 \leq h \leq H$, $Z_h = 1$ if Z_h was selected by some X_i , $i \in S'[j]$ (we say that Z_h is *selected* in this case), otherwise $Z_h = 0$. In this process, the probability of a specific indicator Z_h to be selected by a specific X_i is zero when $x_i = 0$, and $x_i^f \cdot \frac{y_{i,j}^f}{x_i^f} = y_{i,j}^f$ otherwise. Hence, we get that, for all $1 \leq h \leq H$,

$$\begin{aligned} E[Z_h] = Pr(Z_h = 1) &= 1 - \prod_{S_i \in S'[j]} (1 - y_{i,j}^f) \\ &\geq 1 - \left(\frac{1}{\tau_j} \sum_{S_i \in S'[j]} (1 - y_{i,j}^f) \right)^{\tau_j} \\ &= 1 - \left(1 - \frac{Y_j}{\tau_j} \right)^{\tau_j} \geq Y_j \alpha_\varphi. \end{aligned} \quad (4)$$

The first inequality follows from the inequality of the three means, and the second inequality follows from Claim 3.1.

Let $Y_j' = \delta \cdot \sum_{h=1}^H Z_h$ be δ times the number of selected indicators. An important property of Y_j' is that $Y_j' \leq f_j(D)$. Indeed, if $f_j(D) = 1$ then $Y_j' \leq 1$, since there are only $H = \delta^{-1}$ indicators, and if $f_j(D) < 1$, then no more than $f_j(D)\delta^{-1}$ indicators are selected; therefore, $E[Y_j'] \leq E[f_j(D)]$. From (4), we have that

$$E[Y_j'] = \delta \sum_{h=1}^H E[Z_h] \geq \delta \cdot H \cdot Y_j \alpha_\varphi = Y_j \cdot \alpha_\varphi.$$

Hence, $E[f_j(D)] \geq \alpha_\varphi \cdot Y_j$ as desired. \square

The next lemma follows immediately from Lemma 3.4 and (3). Recall that F is an extension by expectation of f , then

Lemma 3.5 $F(\bar{X}) \geq \alpha_\varphi \cdot \mathcal{O}^f \geq (1 - \varepsilon) \cdot \alpha_\varphi \cdot \tilde{\mathcal{O}}$.

It is easy to verify that \bar{X} is a feasible fractional solution for the SUB instance. Recall that the element (S_i, \bar{z}) is big if the cost c_r of this element in some dimension $1 \leq r \leq d$ is larger than ε^3 times the budget in this dimension. We note that if (S_i, \bar{z}) is big then it holds that $S_i \in T$, and $X_{(S_i, \bar{z})} = 1$. Let $D \subseteq U$ be a random set such that $D \sim \bar{X}$. Also, let $D' = D$ if D is ε -nearly feasible, and $D' = \emptyset$ otherwise. By Theorem 2.1, we get that D' is always ε -nearly feasible, and

$$E[f(D')] \geq (1 - O(\varepsilon))F(\bar{x}) \geq (1 - O(\varepsilon))\alpha_\varphi \tilde{\mathcal{O}}.$$

Given a nearly feasible fractional solution for the SUB instance, we now show that it can be converted to a feasible one.

Lemma 3.6 *Let D be an ε -nearly feasible solution for the SUB instance, then D can be converted to a feasible solution D' such that $f(D') \geq (1 - O(\varepsilon))f(D)$.*

Proof: Our conversion will be done in two steps. First, let $D_1 = \{(S_i, \frac{\bar{z}}{1+\varepsilon}) \mid (S_i, \bar{z}) \in D\}$. Clearly, D_1 is feasible in the first d_1 dimensions (for any $1 \leq r \leq d_1$, $c_r(D_1) \leq L_r$); also, $f(D_1) \geq (1 - \varepsilon)f(D)$. Next, we note that for any $d_1 + 1 \leq r \leq d$, for any element $u \in U$ it holds that $c_{u,r} \leq \varepsilon^3 L_r$. Thus, we can apply in these dimensions the fixing procedure of Lemma 2.3. Hence, we can convert D to D' as desired. \square

We now summarize the steps of the algorithm.

Randomized Approximation algorithm for the Semi-fractional Problem

1. For any subset $T \subseteq S$ of size at most $h = d \cdot \varepsilon^{-4}$:
 - (a) Solve $LP(T)$, let \bar{x}^f, \bar{y}^f be the solution found.
 - (b) Define \bar{X} , and let D be a random set $D \sim \bar{X}$, then $D' = D$ if D is ε -nearly feasible, and $D' = \emptyset$ otherwise.
 - (c) Convert D' to a feasible set D'' as in the proof of Lemma 3.6.
2. Let D'' be the solution of maximal profit found for the SUB instance. Convert it to a solution for the semi-fractional problem and return this solution.

Clearly, the algorithm returns a feasible solution for the problem. Consider the iteration in which T is the set of h most profitable elements in $\tilde{\mathcal{O}}$. In this iteration, $E[f(D')] \geq (1 - \varepsilon)\alpha_\varphi \tilde{\mathcal{O}}$. Hence, by Lemma 3.6, $E[f(D'')] \geq (1 - O(\varepsilon))\alpha_\varphi \tilde{\mathcal{O}}$. By a properly selecting the value of ε , we get the following.

Theorem 3.7 *There is a polynomial time randomized $(1 - \varepsilon)\alpha_\varphi$ -approximation algorithm for the semi-fractional problem, for any fixed $\varepsilon > 0$.*

The algorithm can be derandomized, using the technique in Section 2.5. Note that here, the extension by expectation $F(\bar{X})$ can be deterministically evaluated in polynomial time, since the number of non-zero entries in \bar{X} is polynomial.

4 The Budgeted Generalized Assignment Problem

In this section we develop an approximation algorithm for BGAP, and for its generalization, BSAP. Recall that a BSAP instance consists of n items $A = \{a_1, \dots, a_n\}$ and m bins, such that bin i has a d_2 -dimensional capacity \bar{b}_i . Each item a_j has a d_2 -dimensional size $\bar{s}_{i,j} \geq 0$, a d_1 -dimensional cost vector $\bar{c}_{i,j} \geq 0$, and a profit $p_{i,j} \geq 0$ that is gained when a_j is assigned to bin i . Also, given is a d_1 -dimensional budget vector \bar{L} .

We say that a subset of items $S_i \subseteq A$ is a *feasible assignment* for bin i if $\sum_{a_j \in S_i} \bar{s}_{i,j} \leq \bar{b}_i$. We define the cost and profit of assigning S_i to bin i by $\bar{c}(i, S_i) = \sum_{a_j \in S_i} \bar{c}_{i,j}$, and $p(i, S_i) = \sum_{a_j \in S_i} p_{i,j}$, respectively. A *solution* for the problem is a tuple of m *disjoint* subsets of items $S = (S_1, \dots, S_m)$, such that each set S_i is a feasible assignment for bin i . We define the cost of S by $\bar{c}(S) = \sum_{i=1}^m \bar{c}(i, S_i) = \sum_{i=1}^m \sum_{a_j \in S_i} \bar{c}_{i,j}$, and its profit by $p(S) = \sum_{i=1}^m p(i, S_i) = \sum_{i=1}^m \sum_{a_j \in S_i} p_{i,j}$. We say that a solution S is *feasible* if $\bar{c}(S) \leq \bar{L}$. The goal is to find a feasible solution of maximal profit.

As before, we say that a solution S is ε -nearly feasible if $\bar{c}(S) \leq (1 + \varepsilon)\bar{L}$. An item a_j is *small* if, for any bin $1 \leq i \leq m$, it holds that $\bar{c}_{i,j} \leq \varepsilon^3 \bar{L}$; otherwise, a_j is *big*. Also, an assignment $S_i \subseteq A$ of items to bin i is *small* if $\bar{c}(i, S_i) \leq \varepsilon^3 \bar{L}$. Our algorithm uses two special cases of BSAP. The first is *small items BSAP*, in which all items are small; the second is *small assignments BSAP* in which, for any bin i and a feasible assignment S_i , it holds that S_i is a small assignment for bin i .

Overview: Our algorithm proceeds in four stages. The first stage obtains an ε -nearly feasible solution with high profit for small assignments instances of BSAP, by using Theorem 2.1. To do so, we use an interpretation of the classic SAP problem as a submodular optimization

problem and the technique of [12] to obtain a distribution over its solution space. The small assignments property is used to show that the conditions of Theorem 2.1 are satisfied.

The second stage shows how enumeration can be used to reduce a small items instance of BSAP into a small assignments instance, so that the algorithm of the first stage can be applied. The main idea is to guess the most profitable bins in some optimal solution and the approximate cost of these bins in this solution. The algorithm uses this guess to eliminate all big assignments, by adding d_1 linear constraints for each bin. The result of this stage is a 2ε -nearly feasible solution of high profit.

The third stage handles small items instances. The fact that all items are small is essential for converting the nearly feasible solution of the previous stage to a feasible one. The fourth and last stage gives a reduction from general instances to small items instances. This is done by simple enumeration, as in Section 2.3.

4.1 Small Assignments Instances of BSAP

We solve BSAP instances with small assignments by casting BSAP as an instance of generalized SUB (see Section 2.2).⁴ Let \mathcal{I}_i be the set of feasible assignments for bin i , and $U = \{(i, S_i) \mid S_i \in \mathcal{I}_i\}$. Define $f_j : 2^U \rightarrow \mathbb{R}_+$ for any $a_j \in A$ by

$$f_j(V) = \max\{p_{i,j} \mid (i, S_i) \in V, a_j \in S_i\},$$

where the maximum over an empty set is equal to zero. Also, define $f : 2^U \rightarrow \mathbb{R}_+$ by

$$f(V) = \sum_{a_j \in A} f_j(V).$$

It is easy to verify that f is a non-decreasing submodular function. We define the (d_1 -dimensional) cost of an element $(i, S_i) \in U$ by $\bar{c}((i, S_i)) = \bar{c}(i, S_i)$, and as in Section 2, the cost of a subset $V \in 2^U$ is $\bar{c}(V) = \sum_{(i, S_i) \in V} \bar{c}(i, S_i)$. Let \mathcal{M} denote the collection of all subsets in which there is a single assignment to each bin. Formally,

$$\mathcal{M} = \{V \in 2^U \mid \text{there is no } i \text{ such that } (i, S_{i_1}), (i, S_{i_2}) \in V, \text{ where } S_{i_1} \neq S_{i_2}\}.$$

We consider the following instance of generalized SUB: Maximize $f(V)$ subject to the constraints $V \in \mathcal{M}$ and $\bar{c}(V) \leq \bar{L}$.

We note that any set $V \in \mathcal{M}$ can be converted to a solution S for BSAP with $\bar{c}(S) \leq \bar{c}(V)$ and $p(S) = f(V)$. To do so, we assign each item a_j to a bin i if there exists $(i, S_i) \in V$ such that $a_j \in S_i$ and $f_j(V) = p_{i,j}$. If no such bin i exists, we do not assign a_j to any bin. Similarly, any solution S for BSAP can be converted to a set $V \in \mathcal{M}$ such that $\bar{c}(S) = \bar{c}(V)$ and $p(S) = f(V)$.

Now, we use a technique of [12] to obtain a fractional solution for the generalized SUB instance. Consider the linear program LP-BSAP over the variables X_i^S , for all $1 \leq i \leq m$ and $S \in \mathcal{I}_i$. We note that the optimal solution for LP-BSAP is at least $p(\mathcal{O})$, where \mathcal{O} is an optimal solution for the BSAP instance. Since we have d_2 linear constraints over the bins, where $d_2 \geq 1$ is some constant, a feasible solution of value $(1 - \varepsilon)$ times the optimal can be found in polynomial time (see [12] for more details). Let \bar{X} be such a solution, then the value of the solution \bar{X} is at least $(1 - \varepsilon) \cdot p(\mathcal{O})$.

⁴The submodular interpretations of GAP and SAP are well-known (see, e.g., [10] and [12]).

$$\begin{aligned}
\text{(LP-BSAP)} \quad & \text{maximize} && \sum_{i=1}^m \sum_{S \in \mathcal{I}_i} X_i^S \cdot p(i, S) \\
\text{subject to:} \quad & \forall a_j \in A && \sum_{i=1}^m \sum_{S \in \mathcal{I}_i | a_j \in S} X_i^S \leq 1 \\
& \forall 1 \leq i \leq m && \sum_{S \in \mathcal{I}_i} X_i^S = 1 \\
& \forall 1 \leq r \leq d && \sum_{i=1}^m \sum_{S \in \mathcal{I}_i} X_i^S c_r(i, S) \leq L_r
\end{aligned}$$

Let \mathcal{D}_i be a random variable over \mathcal{I}_i with $Pr[\mathcal{D}_i = S] = X_i^S$. Define a random set $\mathcal{D} = \{(1, \mathcal{D}_1), (2, \mathcal{D}_2), \dots, (m, \mathcal{D}_m)\}$, and let χ be the distribution of \mathcal{D} . In [12] it is shown that $E[f(\mathcal{D})]$ is at least $(1 - e^{-1})$ times the value of the solution \bar{X} ; thus, $E[f(\mathcal{D})] \geq (1 - e^{-1}) \cdot (1 - \varepsilon) \cdot p(\mathcal{O})$. It is easy to verify that the conditions of Theorem 2.1 are satisfied for the distribution χ . We define $\mathcal{D}' = \mathcal{D}$ if \mathcal{D} is ε -nearly feasible, and $\mathcal{D}' = \emptyset$ otherwise. Then, by Theorem 2.1, \mathcal{D}' is always ε -nearly feasible and $E[f(\mathcal{D}')] \geq (1 - e^{-1})(1 - O(\varepsilon))p(\mathcal{O})$. As before, \mathcal{D}' can be converted to a solution S for the BSAP instance, such that $p(S) = f(\mathcal{D}')$, and $\bar{c}(S) \leq \bar{c}(\mathcal{D}')$. We summarize the above steps in the following.

Approximation Algorithm for Small Assignments BSAP Instances

1. Find a $(1 - \varepsilon)$ -approximate solution \bar{X} for LP-BSAP.
2. For any bin $1 \leq i \leq m$, select an assignment $\mathcal{D}_i = S_i$ with probability $X_i^{S_i}$ and define $\mathcal{D} = \{(1, \mathcal{D}_1), \dots, (m, \mathcal{D}_m)\}$
3. If \mathcal{D} is ε -nearly feasible return \mathcal{D} as the solution, else return an empty assignment.

Lemma 4.1 *The above Approximation Algorithm for Small Assignments BSAP Instances outputs in polynomial time an ε -nearly feasible solution with expected profit at least $(1 - e^{-1})(1 - O(\varepsilon))p(\mathcal{O})$, where \mathcal{O} is an optimal solution.*

4.2 Reduction to Small Assignments BSAP Instances

We now describe an algorithm which reduces a general BSAP instance to a small assignments instance. We use this reduction to obtain an augmentation algorithm for general instances, by applying Algorithm for Small Assignment Instances of Section 4.1.

Let $\mathcal{O} = (S_1, \dots, S_m)$ be an optimal solution for an instance of BSAP. We say that the profit of bin i (with respect to \mathcal{O}) is $p^{\mathcal{O}}(i) = p(i, S_i)$, and the cost of bin i is $\bar{c}^{\mathcal{O}}(i) = \bar{c}(i, S_i)$. The first step in our algorithm is to guess the set T of $h = d_1 \cdot \varepsilon^{-4}$ most profitable bins in the solution \mathcal{O} . We then guess the cost of any bin $i \in T$ in each dimension $1 \leq r \leq d_1$. with accuracy $\delta = \varepsilon \cdot h^{-1}$. That is, for each bin $i \in T$ we guess a d_1 -dimensional vector of integers $\bar{k}_i = (k_{i,1}, \dots, k_{i,d_1})$ such that, for any $1 \leq r \leq d_1$,

$$k_{i,r} \cdot \delta \cdot L_r \leq c^{\mathcal{O}}(i) \leq (k_{i,r} + 1) \cdot \delta \cdot L_r \quad (5)$$

As the number of values $k_{i,r}$ can get is at most $\lceil \delta^{-1} \rceil$, we can go over all possible cost vectors in polynomial time, for some constant $\varepsilon > 0$.

We use our guess of the set T and the vectors \bar{k}_i to define a residual instance of BSAP. The ground set of elements is A , and there are m bins. We define the budget of the residual instance, \bar{L}' , to be $L'_r = L_r(1 - \sum_{i \in T} k_{i,r} \delta)$, for $1 \leq r \leq d_1$. For any $i \notin T$, the feasible set of assignments for bin i is $\mathcal{I}'_i = \{S \mid S \in \mathcal{I}_i, \bar{c}(i, S_i) \leq \varepsilon^3 \bar{L}'\}$, and for any $i \in T$ the feasible set is $\mathcal{I}'_i = \{S \mid S \in \mathcal{I}_i, \bar{c}(i, S_i) \leq (k_{i,r} + 1) \cdot \delta \cdot L_r\}$. In both cases, the set of feasible assignments for bin i is defined by $d_1 + d_2$ linear constraints. The new cost of a_j when assigned to bin i is $\bar{c}'_{i,j} = \bar{c}_{i,j}$ if $i \notin T$ and $\bar{c}'_{i,j} = 0$ if $i \in T$. The new profit of a_j when assigned to bin i is $p'_{i,j} = p_{i,j}$.

A crucial property of the residual instance is that it is a small assignments instance. For any bin $i \in T$ and $S_i \in \mathcal{I}'_i$ it holds that $\bar{c}'(i, S_i) = 0$. For any bin $i \notin T$ and $S_i \in \mathcal{I}'_i$, by the definition of \mathcal{I}'_i it holds that $\bar{c}'(i, S_i) = \bar{c}(i, S_i) \leq \varepsilon^3 \bar{L}'$. The relation between the solutions of the residual and original problems is stated in the next two lemmas.

Lemma 4.2 *Let $S = (S_1, \dots, S_m)$ be an ε -nearly feasible solution for the residual problem with respect to a set T (of size at most h) and a collection of vectors \bar{k}_i . Then S is a (2ε) -nearly feasible solution for the original problem with $p(S) = p'(S)$.*

Proof: Clearly, S is a solution for the original problem since, for every bin i , it holds that $\mathcal{I}'_i \subseteq \mathcal{I}_i$. Also, for any bin i and $a_j \in A$ it holds that $p_{i,j} = p'_{i,j}$. Hence, $p(S) = p'(S)$. As for the cost, for any $1 \leq r \leq d_1$, we have

$$\begin{aligned}
c_r(S) &= \sum_{i \in T} c_r(i, S_i) + \sum_{i \notin T} c'_r(i, S_i) \\
&\leq \sum_{i \in T} (k_{i,r} + 1) \cdot \delta L_r + \sum_{i \notin T} c'_r(i, S_i) \\
&\leq |T| \cdot \delta L_r + \sum_{i \in T} k_{i,r} \cdot \delta L_r + \sum_{1 \leq i \leq m} c'_r(i, S_i) \\
&\leq h \cdot \delta L_r + \sum_{i \in T} k_{i,r} \cdot \delta L_r + (1 + \varepsilon) L'_r \\
&\leq \varepsilon \cdot L_r + L_r + \varepsilon L'_r \leq (1 + 2\varepsilon) L_r
\end{aligned}$$

□

Lemma 4.3 *Let T be the set of h most profitable bins in an optimal solution \mathcal{O} , and let \bar{k}_i be the vector of cost guesses for which (5) holds. Then there is a feasible solution S' for the residual problem with respect to T and \bar{k}_i whose profit is $p'(S') \geq (1 - \varepsilon)p(\mathcal{O})$.*

Proof: Let $\mathcal{O} = (S_1, \dots, S_m)$ be an optimal solution for the original problem. We define a solution $S' = (S'_1, \dots, S'_m)$ for the residual problem by $S'_i = S_i$ if $S_i \in \mathcal{I}'_i$ and $S'_i = \emptyset$ otherwise. Clearly, S' is a solution for the residual problem. For any dimension $1 \leq r \leq d_1$,

$$L_r \geq c_r^{\mathcal{O}} = \sum_{i \notin T} c_r(i, S_i) + \sum_{i \in T} c_r(i, S_i) \geq \sum_{i \notin T} c_r(i, S_i) + \sum_{i \in T} k_{i,r} \cdot \delta \cdot L_r = \sum_{i \notin T} c_r(i, S_i) + (L_r - L'_r)$$

This implies that $\sum_{i \notin T} c_r(i, S_i) \leq L'_r$. Now, for any $i \in T$ it holds that $c'_r(i, S'_i) = 0$, and for

any i it holds that $c'_r(i, S'_i) \leq c_r(i, S_i)$. Thus, we have that

$$\begin{aligned} c'_r(S') &= \sum_{i \in T} c'_r(i, S_i) + \sum_{i \notin T} c'_r(i, S_i) \\ &\leq \sum_{i \notin T} c_r(i, S_i) \leq L'_r, \end{aligned} \tag{6}$$

i.e., S' is a feasible solution for the residual problem.

Let H be the set of bins for which $S_i \neq S'_i$. Since (5) holds for the vectors \bar{k}_i , clearly, for any $i \in T$, we have that $S'_i = S_i$, and thus $H \cap T = \emptyset$. By (6), the total cost of bins not in T is bounded by \bar{L}' . Hence, for all of them except for at most $d_1 \cdot \varepsilon^{-3}$, it holds that $c_r(i) \leq \varepsilon^3 L'_r$, for $1 \leq r \leq d_1$. In other words, for every $i \notin T$, except at most $d_1 \cdot \varepsilon^{-3}$, it holds that $S_i \in \mathcal{I}'_i$ and $S_i = S'_i$. It follows that $|H| \leq d_1 \cdot \varepsilon^{-3}$, and that none of the bins in T is in H . The profit of any bin that is not in T is smaller than $p(\mathcal{O})/|T|$ (since T is the set of most profitable bins). Therefore,

$$\sum_{i \in H} p^{\mathcal{O}}(i) \leq \frac{d_1 \cdot \varepsilon^{-3} \cdot p(\mathcal{O})}{|T|} = p(\mathcal{O}) \cdot \frac{d_1 \cdot \varepsilon^{-3}}{h} = \varepsilon \cdot p(\mathcal{O}).$$

It follows that

$$p'(S') = p(S') = \sum_{i=1}^m p(i, S'_i) = \sum_{i=1}^m p(i, S_i) - \sum_{i \in H} p(i, S_i) = p(\mathcal{O}) - p(H) \geq (1 - \varepsilon) \cdot p(\mathcal{O}).$$

□

We summarize with the following algorithm.

Nearly Feasible Algorithm for BSAP

1. Enumerate over all subsets T , $|T| \leq h = d_1 \cdot \varepsilon^{-4}$, and cost vectors \bar{k}_i for any $i \in T$.
 - (a) Define a residual instance with respect to T and \bar{k}_i , and run Algorithm for Small Assignments BSAP on the residual instance. Consider the resulting solution as a solution for the original problem.
2. Return the best solution found.

Theorem 4.4 *Let \mathcal{O} be an optimal solution for a BSAP instance. Then the Nearly Feasible Algorithm for BSAP returns in polynomial time a (2ε) -nearly feasible solution S , with expected profit at least $E[p(S)] \geq (1 - e^{-1})(1 - O(\varepsilon))p(\mathcal{O})$.*

Proof: As the number of possibilities for T and the vectors \bar{k}_i is polynomial for fixed values of d_1, d_2 and ε , and Algorithm for Small Assignments BSAP is polynomial as well, we get that the Nearly Feasible Algorithm for BSAP runs in polynomial time.

Since the algorithm for Small Assignments BSAP always returns an ε -nearly feasible solution (for the residual problem), by Lemma 4.2, we get that the solutions output by the algorithm are always (2ε) -nearly feasible for the original problem. Finally, in the iteration where the set T and the vectors \bar{k}_i satisfy the conditions of Lemma 4.3, the optimal solution for the residual problem has profit at least $(1 - \varepsilon)p(\mathcal{O})$. Thus, by Lemma 4.1, the expected profit of the solution returned by Algorithm for Small Assignments BSAP is at least $(1 - e^{-1})(1 - O(\varepsilon))(1 - \varepsilon)p(\mathcal{O}) = (1 - e^{-1})(1 - O(\varepsilon))p(\mathcal{O})$.

As the expected profit of the solution returned by the algorithm is at least the expected profit in any iteration, this yields the statement of the theorem. \square

4.3 Small Items BSAP Instances

We now consider inputs in which all items are small. For such inputs we can fix a nearly feasible solution. The proof of the next result is similar to the proof of Lemma 2.3 (details omitted).

Lemma 4.5 *Let $S = (S_1, \dots, S_m)$ be an ε -nearly feasible solution for a small items instance of BSAP. Then S can be converted to a feasible solution S' such that $p(S') \geq (1 - O(\varepsilon))p(S)$.*

Lemma 4.5 can be easily coupled with the Nearly Feasible Algorithm to obtain an approximation algorithm for small items instances of BSAP, which always returns a feasible solution.

Algorithm for Small Items BSAP

1. Run the Nearly Feasible Algorithm for BSAP. Let S be the resulting solution.
2. Convert S to a feasible solution S' (as in Lemma 4.5) and return S' .

The properties of the algorithm follow immediately from Lemmas 4.4 and 4.5.

Lemma 4.6 *Algorithm for Small Items BSAP returns a feasible solution for the problem with expected profit of at least $(1 - e^{-1})(1 - O(\varepsilon))p(\mathcal{O})$, where \mathcal{O} is an optimal solution.*

4.4 General Inputs

We now use Algorithm for Small Items BSAP to obtain $(1 - e^{-1} - \varepsilon)$ -approximation for general inputs. Given an input for BSAP, let $\mathcal{O} = (S_1, \dots, S_m)$ be an optimal solution. We say that the profit of an element a_j in \mathcal{O} is zero if it is not assigned to any bin, and $p_{i,j}$ if it is assigned to bin i .

Let $T^* = (T_1^*, \dots, T_m^*)$ be a feasible assignment of elements to bins. Given T^* , we define a residual instance of the problem with respect to T^* . The new budget is $\bar{L}^* = \bar{L} - \bar{c}(T^*)$, the new set of elements A^* is the collection of all elements in A which are not assigned in T^* to any bin. Consider the assignment of a_j to bin i . If $\bar{c}_{i,j} \leq \varepsilon^3 \bar{L}^*$, then $p_{i,j}^* = p_{i,j}$ and $\bar{c}_{i,j}^* = \bar{c}_{i,j}$; otherwise, set $p_{i,j}^* = 0$ and $\bar{c}_{i,j}^* = 0$. The size remains $\bar{s}_{i,j}^* = \bar{s}_{i,j}$. The new capacity of bin i is $\bar{b}_i^* = \bar{b}_i - \sum_{a_j \in T_i^*} \bar{s}_{i,j}$.

Clearly, the residual instance with respect to T^* is small items instance. It is easy to verify that, given a feasible assignment for the residual problem with respect to T^* of profit v , we can derive a feasible assignment for the original problem of profit $p(T^*) + v$. Now, consider T^* to be the assignment of the $h = d_1 \cdot \varepsilon^{-4}$ most profitable elements in \mathcal{O} . Then there is a solution for the residual problem of value $p(\mathcal{O}) - (1 + \varepsilon)p(T^*)$ (the proof is similar to the proof of Lemma 4.3). Thus, if we guessed T^* correctly, Algorithm for Small Items BSAP can be used to obtain a solution with expected profit at least $(1 - e^{-1} - O(\varepsilon))(p(\mathcal{O}) - (1 + \varepsilon)p(T^*))$ for the residual problem, from which we can derive a solution for the original instance of profit $(1 - e^{-1} - O(\varepsilon))(p(\mathcal{O}) - (1 + \varepsilon)p(T^*)) + p(T^*) \geq (1 - e^{-1} - O(\varepsilon))p(\mathcal{O})$.

WE summarize with an algorithm for general inputs.

Approximation Algorithm for BSAP

1. For any feasible assignment T^* of at most $h = d \cdot \varepsilon^{-4}$ elements:

(a) Find a $(1 - e^{-1} - O(\varepsilon))$ -approximate solution S^* for the residual problem with respect to T^* .

(b) Derive from S^* a solution for the original problem of profit $p(S^*) + p(T^*)$.

2. Return the solution of maximal profit found.

Theorem 4.7 *There is a polynomial time randomized $(1 - e^{-1} - \varepsilon)$ -approximation algorithm for BSAP, for any fixed $\varepsilon > 0$.*

5 Discussion

In this paper we established a strong relation between the continuous relaxation of SUB and the discrete problem. This relation is nearly optimal and suggests that future research should focus on deriving better approximation ratios for the continuous problem.

The question whether better rounding exists remains open; namely, is it possible to obtain an α -approximation algorithm for SUB, given an $\alpha < 1$ approximation algorithm for the continuous problem? And more specifically, is there a polynomial time $(1 - e^{-1})$ -approximation for SUB with monotone objective function?

Finally, the running times of our algorithms are exponential in $1/\varepsilon$, thus rendering them impractical. Yet, the hardness results for d -dimensional Knapsack (see, e.g., [16, 22, 19]), a special case of SUB, hint that significant improvements over these running times may be impossible.

References

- [1] A. Ageev and M. Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *J. Combinatorial Optimization*, 8(3):307–328, 2004.
- [2] N. Bansal, N. Korula, V. Nagarajan, and A. Srinivasan. On α -column sparse packing programs. In F. Eisenbrand and F. B. Shepherd, editors, *IPCO*, volume 6080 of *Lecture Notes in Computer Science*, pages 369–382. Springer, 2010.
- [3] G. Calinescu, C. Chekuri, M. Pa’l, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. *SIAM Journal on Computing*, to appear.
- [4] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. In *IPCO*, pages 182–196, 2007.
- [5] C. Chekuri and A. Kumar. Maximum coverage problem with group budget constraints and applications. In *APPROX-RANDOM*, pages 72–83, 2004.
- [6] C. Chekuri, J. Vondrák, and R. Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *FOCS*, 2010.
- [7] C. Chekuri, J. Vondrák, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *Manuscript*, 2010.
- [8] E. D. Demaine and M. Zadimoghaddam. Scheduling to minimize power consumption using submodular functions. In *SPAA*, pages 21–29, 2010.

- [9] U. Feige. A threshold of $\ln n$ for approximating set cover. *J.of ACM*, 45(4):634–652, 1998.
- [10] U. Feige and J. Vondrák. Approximation algorithms for allocation problems: Improving the factor of $1 - 1/e$. In *FOCS*, pages 667–676. IEEE Computer Society, 2006.
- [11] U. Feige, V.S.Mirrokní, and J. Vondrák. Maximizing non-monotone submodular functions. In *FOCS*, 2007.
- [12] L. Fleischer, M. X. Goemans, V. S. Mirrokní, and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 611–620, New York, NY, USA, 2006.
- [13] T. Fujito. Approximation algorithms for submodular set cover with applications. *IEICE Trans. Inf. and Systems*, E83-D(3), 2000.
- [14] S. O. Gharan and J. Vondrák. Submodular maximization by simulated annealing. *CoRR*, abs/1007.1632, 2010.
- [15] J. D. Hartline, V. S. Mirrokní, and M. Sundararajan. Optimal marketing strategies over social networks. In *WWW*, pages 189–198, 2008.
- [16] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, October 2004.
- [17] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Inf. Process. Letters*, 70(1):39–45, 1999.
- [18] A. Kulik and H. Shachnai. On lagrangian relaxation and subset selection problems. In *WAOA*, pages 160–173, 2008.
- [19] A. Kulik and H. Shachnai. There is no EPTAS for two-dimensional knapsack. *Inf. Process. Lett.*, 110(16):707–710, 2010.
- [20] A. Kulik, H. Shachnai, and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *SODA*, pages 545–554, 2009.
- [21] J. Lee, V.S.Mirrokní, V. Nagarajan, and M. Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *STOC*, 2009.
- [22] M. J. Magazine and M.-S. Chern. A note on approximation schemes for multidimensional knapsack problems. *Mathematics of Operations Research*, 9(2):244–247, 1984.
- [23] G. Nemhauser and L. Wolsey. Best algorithms for approximating the maximum of submodular set function. *Mathematics of Operations Research*, 3(3):177–188, 1978.
- [24] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.
- [25] A. Schriejver. *Combinatorial Optimization - polyhedra and efficiency*. Springer Verlag - Berlin Heidelberg, 2003.
- [26] M. Sviridenko. A note on maximizing a submodular set function subject to knapsack constraint. *Operations Research Letters*, 32:41–43, 2004.
- [27] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, pages 67–74, 2008.

A Basic Properties of Submodular Functions

In this section we give some simple properties of submodular functions. Recall that $f : 2^U \rightarrow \mathbb{R}$ is a submodular function if $f(S) + f(T) \geq f(S \cup T) + f(T \cap S)$ for any $S, T \subseteq U$. We define $f_T(S) = f(S \cup T) - f(T)$.

Lemma A.1 *Let $f : 2^U \rightarrow \mathbb{R}$ be a submodular function with $f(\emptyset) \geq 0$, and let $S = S_1 \cup S_2 \cup \dots \cup S_k$, where S_i are disjoint sets. Then*

$$f(S) \geq f(S_1) + f(S_2) + \dots + f(S_k).$$

Proof: By induction on k . For $k = 2$, since f is a submodular function, we have that

$$f(S_1) + f(S_2) \geq f(S_1 \cup S_2) + f(S_1 \cap S_2) = f(S) + f(\emptyset),$$

and since $f(\emptyset) \geq 0$, we get that $f(S) \leq f(S_1) + f(S_2)$.

For $k > 2$, using the induction hypothesis twice, we have

$$f(S) \leq f(S_1) + f(S_2) + \dots + f(S_{k-2}) + f(S_{k-1} \cup S_k) \leq f(S_1) + f(S_2) + \dots + f(S_k).$$

□

Lemma A.2 *Let $f : 2^U \rightarrow \mathbb{R}_+$ be a submodular function, and let $S, T_1, T_2 \subseteq U$ such that $T_1 \subseteq T_2$ and $S \cap T_2 = \emptyset$. Then, $f_{T_2}(S) \leq f_{T_1}(S)$.*

Proof: Since f is submodular,

$$f(S \cup T_1) + f(T_2) \geq f(S \cup T_1 \cup T_2) + f((S \cup T_1) \cap T_2) = f(S \cup T_2) + f(T_1).$$

Hence, $f_{T_2}(S) \leq f_{T_1}(S)$.

□

Lemma A.3 *Let $f : 2^U \rightarrow \mathbb{R}_+$ be a submodular function, and let $S = S_1 \cup S_2 \cup \dots \cup S_k$, where S_i are disjoint sets. Then,*

$$f(S) \geq \sum_{i=1}^k f_{S \setminus S_i}(S_i).$$

Proof: We note that

$$f(S) = \sum_{i=1}^k f_{S_1 \cup \dots \cup S_{i-1}}(S_i).$$

By Lemma A.2, for each $i > 1$, $f_{S_1 \cup \dots \cup S_{i-1}}(S_i) \geq f_{S \setminus S_i}(S_i)$. Hence,

$$f(S) \geq \sum_{i=1}^k f_{S \setminus S_i}(S_i).$$

□