Improved Approximations for Non-monotone Submodular Maximization with Knapsack Constraints

Ariel Kulik∗ Hadas Shachnai†

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 $\alpha$-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 $\varepsilon > 0$.

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 Problems with additional knapsack constraints.

∗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).
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., [10] and the references therein, and the comprehensive surveys in [8, 20, 15]). 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 (see, e.g., [11]).

In many settings, including set covering or matroid optimization, the underlying submodular functions are monotone, meaning that $f(S) \leq f(T)$ whenever $S \subseteq T$. Here, we are more interested in the general case where $f(S)$ 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) induced by a set of vertices $S$, and $w(e)$ is the weight of an edge $e$.

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 $L$, 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 $L$, such that $f(S)$ is maximized.

There has been extensive work on maximizing submodular monotone functions subject to matroid constraint. 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 [17] 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 certain submodular monotone functions (see, e.g., [1, 12, 19, 4]). For a general matroid constraint, Calinescu et al. showed in [3] 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 [20] to general monotone submodular functions. In [14] we developed a framework for maximizing monotone submodular functions subject to $d$ knapsack constraints that yields a $(1 - e^{-1} - \varepsilon)$-approximation to the optimum for any fixed $\varepsilon > 0$, where $d \geq 1$ is some constant. 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 [16] in the oracle model, and the later result of Feige [6] for the specific case of maximum coverage, under the assumption that $P \neq NP$. Recently, Demaine and Zadimoghaddam [5] considered 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. [8] considered the (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. [15] 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 $d \geq 1$.

\footnote{A (weighted) matroid is a system of ‘independent subsets’ of a universe, which satisfies certain hereditary and exchange properties [18].}
Several fundamental algorithms for submodular maximization (see, e.g., [1, 3, 20, 15]) use a continuous extension of submodular function, to which we refer as extension by expectation. Given a submodular function \( f : 2^U \to \mathbb{R} \), we define \( F : [0, 1]^U \to \mathbb{R} \). For any \( y \in [0, 1]^U \), let \( R \subseteq U \) be a random variable such that \( i \in R \) with probability \( y_i \) (we say that \( R \sim y \)). Then define \( F(y) = E[f(R)] \). The general framework of these algorithms is to first obtain 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 \( \text{SUB} \). Let \( P = \{ \bar{y} \in [0, 1]^U | \sum_{i \in U} y_i c(i) \leq \bar{L} \} \) 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 \( A \) yields \( \alpha \)-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, \( A \) finds a feasible solution for the continuous problem of value at least \( \alpha \mathcal{O} \), where \( \mathcal{O} \) is the value of the optimal discrete solution for 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, 3]). For monotone submodular functions, Vondrák presented in [20] a \((1 - e^{-1} - o(1))\)-approximation algorithm for the continuous problem, which was later used as a main building block in the \((1 - e^{-1} - \varepsilon)\)-approximation algorithm of [14]. Subsequently, Lee et al. [15] considered the problem of maximizing any submodular function with multiple knapsack constraints and developed a \((1/4 - o(1))\)-approximation algorithm for the continuous problem; however, noting that the rounding method of [14], which proved useful for monotone functions, cannot be applied in the non-monotone case, a \((1/5 - \varepsilon)\)-approximation was obtained for the discrete problem, by using simple randomized rounding.

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 2.5) that for any non-negative submodular function, an \( \alpha \)-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 \( \text{SUB} \), for any \( \varepsilon > 0 \), thus improving the best known result for the problem, due to Lee et al. [15]. 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 \( \alpha \), immediately implies an approximation ratio of \((\alpha - \varepsilon)\) for the original instance.

The technique that we use (as given in Section 2) generalizes (and simplifies) the framework developed in [14]. In particular, our results hold for any non-negative submodular function. Moreover, the relation between the approximation ratio for the continuous relaxation and the ratio for the discrete version for specific submodular functions (as given in Theorem 2.5) cannot be deduced using the framework of [14].

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 deterministic version of our technique. Specifically, we obtain a deterministic \((1/4 - \varepsilon)\)-approximation for general instances and \((1 - e^{-1} - \varepsilon)\)-approximation for

\footnote{For example, for maximum coverage with multiple knapsack constraints, a \((1 - (1 - 1/r) - \varepsilon)\)-approximation follows from our technique for any instance in which the maximum frequency of an element is \( r \geq 1 \), while the framework of [14] yields the ratio \( 1 - e^{-1} - \varepsilon \).}
instances where the submodular function is monotone.

Our approach has a potential of wider applicability, which we demonstrate on the examples of the Generalized Assignment Problem and Maximum Coverage Problems with additional knapsack constraints (given in Section 3 and Appendix D, respectively).

Due to space constraints some of the proofs are relegated to Appendix C.

2 Maximizing Submodular Functions

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 $\tilde{c}(i) \leq \varepsilon^3 L$; 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 $L$. We say that $S$ is $\varepsilon$-nearly feasible (or nearly feasible, if $\varepsilon$ is known from the context) if the total cost of the elements in $S$ is bounded by $(1 + \varepsilon)L$. 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 $\bar{c}(S) = \sum_{i \in S} \tilde{c}(i)$, and $c_r(S) = \sum_{i \in S} c_r(i)$. 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 $\bar{c}(\bar{y}) = \sum_{i \in U} \tilde{c}(i) \cdot y_i$.

Overview Our algorithm consists of two main 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 $\alpha$-approximate solution for the continuous problem. It relies heavily on Theorem 2.1 that gives conditions on the probabilistic domain of solutions, which 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 for the rounding procedure.

The profit enumeration phase uses enumeration over the most profitable elements in an optimal solution, to reduce 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 [20], or [15]) to obtain approximation algorithm for SUB.

2.2 A Probabilistic Theorem

We first prove a general probabilistic theorem that 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 $\bar{c}(S) \leq L$ and $f(S)$ is maximized.

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

1. $E[f(D)] \geq O/5$, where $O$ is an optimal solution for the given instance.
2. For $1 \leq r \leq d$, $E[c_r(D)] \leq L_r$
3. For $1 \leq r \leq d$, $c_r(D) = \sum_{k=1}^m c_r(D_k)$, where $D_k \sim \chi_k$ and $D_1, \ldots, D_m$ 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 $\varepsilon$-nearly feasible, and $D' = \emptyset$ otherwise. Then $D'$ is always $\varepsilon$-nearly feasible, $D' \in M$, and $E[f(D')] \geq (1 - O(\varepsilon))E[f(D)]$.

**Proof:** We use in the proof the next claims, whose proofs are given in the appendix. Define an indicator random variable $F$ such that $F = 1$ if $D$ is $\varepsilon$-nearly feasible, and $F = 0$ otherwise.

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

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

**Claim 2.3** For any integer $\ell > 1$, if $R \leq \ell$ then $f(D) \leq 2d\ell \cdot O$.

Combining the above results we have

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

By definition, $D'$ is always $\varepsilon$-nearly feasible, and $D' \in M$. This completes the proof. □

### 2.3 Rounding Instances with No Big Elements

A main advantage of inputs with no big elements is that any nearly feasible solution can be easily converted to a feasible one with only a slight decrease in the total value.

**Lemma 2.2** If $S \subseteq U$ is an $\varepsilon$-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 $\ell$ disjoint subsets $S_1, S_2, \ldots, S_\ell$ 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 $f$ is submodular, by Lemma A.3 we have $f(S) \geq \sum_{j=1}^\ell f_{S_j}(S_j)$. Hence, there is some $1 \leq j \leq \ell$ such that $f_{S_j} S_j(S_j) \leq \frac{f(S)}{\ell} = f(S) \cdot O(\varepsilon)$ (note that $f_{S_j} S_j(S_j)$ can have a negative value). 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 in each dimension to obtain a feasible set $S'$ with $f(S') \geq (1 - O(\varepsilon)) f(S)$. □

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

**Randomized rounding algorithm for SUB with no big elements**

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

1. Define a random set $D \sim \bar{x}$. Set $D' = D$ if $D$ is $\varepsilon$-nearly feasible, and $D' = \emptyset$ otherwise.
2. Convert $D'$ to a feasible set $D''$ as in Lemma 2.2 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 2.2, $E[f(D'')] \geq (1 - O(\varepsilon))F(\bar{x})$. Hence, we have

**Lemma 2.3** 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 run 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. Let $\tilde{L}' = \tilde{L} - \bar{c}(T)$ be the new budget, and the universe $U'$ consists of all elements $i \in U \setminus T$ such that $\bar{c}(i) \leq \varepsilon^3 \tilde{L}'$, where...
and all elements in $T$ (formally, $U' = T \cup \{i \in U \setminus T | \bar{c}(i) \leq \varepsilon^3 \bar{L}'\}$). 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 $c(S) \leq c'(S) + c(T) \leq \bar{L}' + c(T) = \bar{L}$, which means that any feasible solution for the residual problem is also feasible for the original problem.

Consider the following algorithm.

**Approximation algorithm for SUB**

**Input:** A SUB instance and an $\alpha$-approximation algorithm $A$ for the continuous problem with respect to the function $f$.

1. For any $T \subseteq U$ such that $|T| \leq h = [d \cdot \varepsilon^{-4}]$.
   
   (a) Use $A$ to obtain an $\alpha$-approximate solution $\bar{x}$ for the continuous residual problem with respect to $T$.
   
   (b) Use the rounding algorithm of Section 2.3 to convert $\bar{x}$ to a feasible solution $S$ for the residual problem (note the residual problem has no big elements).

2. Return the best solution found.

**Lemma 2.4** The above approximation algorithm returns $(\alpha - O(\varepsilon))$-approximate solution for SUB and uses a polynomial number of invocations of algorithm $A$.

**Proof:** By Lemma 2.3, 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 $O = \{i_1, \ldots, i_k\}$ be an optimal solution for the input $I$. For $\ell \geq 1$, let $K_\ell = \{i_1, \ldots, i_\ell\}$, and assume that the elements are ordered by their residual profits, i.e., $i_\ell = \text{argmax}_{i \in O \setminus K_{\ell-1}} f_{K_{\ell-1}}(\{i\})$.

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

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

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

We summarize in the next result.

**Theorem 2.5** Let $f$ be a submodular function, and suppose there is a polynomial time $\alpha$-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 the continuous problem on general instances [15], we have

**Theorem 2.6** There is a polynomial time randomized $(1/4 - \varepsilon)$-approximation algorithm for SUB, 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 $f$. 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 \( \epsilon \)). Then, we go over all the possible values, and we are promised to obtain a solution of high value.

A key ingredient 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 \( \vec{i} \in \{0,1\}^U \), 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 \( \bar{x}_{i,j}(\delta) = \bar{x} + \delta \vec{i} - \delta \vec{j} \). Let \( \delta_{i,i,j}^+ \) and \( \delta_{i,i,j}^- \) (for short, \( \delta^+ \) and \( \delta^- \)) be the maximal and minimal value of \( \delta \) 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 \( F_{i,i,j}(\delta) = F(\bar{x}_{i,j}(\delta)) \) over the elements in \( U \). The function \( F_{i,i,j}(\delta) \) is convex (see [2] for a detailed proof), thus \( \bar{x}' = \text{argmax}_{(\bar{x}_{i,j}(\delta^+), \bar{x}_{i,j}(\delta^-))} F(\bar{x}) \) 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 2.7** 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 \( k' = (8 \ln(2k)/\epsilon)^d \) fractional entries, such that \( \bar{c}(\bar{x}') \leq (1 + \epsilon)\bar{L} \), and \( F(\bar{x}') \geq F(\bar{x}) \), in time polynomial in \( k \).

**Proof Sketch:** 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 \( U' \).

\[
\bar{c}'(i) = \begin{cases} 
c_r(i) & \text{if } i \notin U' \\
0 & \text{if } i \in U' \\
\frac{\epsilon L_r}{2k} + (1 + \epsilon/2)^i & \text{if } c_r(i) \leq \frac{\epsilon L_r}{2k}(1 + \epsilon/2)^i < c_r(i) < \frac{\epsilon L_r}{2k}(1 + \epsilon/2)^{i+1}
\end{cases}
\]

The number of distinct values \( \bar{c}'(i) \) can get for \( i \in U' \) is bounded by \( \frac{8 \ln(2k)}{\epsilon} \). Hence, the number of distinct values \( \bar{c}'(i) \) can get for \( i \in U' \) is bounded by \( k' = \left(\frac{8 \ln(2k)}{\epsilon}\right)^d \). Now, we can use pipage rounding, i.e., choose in each step two elements \( i, j \) such that \( \bar{c}'(i) = \bar{c}'(j) \). We repeat this step until there are no two elements with fractional entries and the same cost. This gives a new vector \( \bar{x}' \) with up to \( k' \) fractional entries, satisfying \( F(\bar{x}') \geq F(\bar{x}) \). Since we use this step only for elements with the same cost, we are guaranteed to have \( \bar{c}'(\bar{x}) = \bar{c}'(\bar{x}') \), and by the definition of the new cost, we get that \( \bar{c}(\bar{x}') \leq (1 + \epsilon)\bar{L} \).

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 \( k' = (8 \ln(2|U|)/\epsilon)^d \). We can then apply the conversion again, to obtain \( \bar{x}'' \) with at most \( k'' = O(\log |U|) \) fractional entries.

**Lemma 2.8** Let \( \bar{x} \in [0,1]^U \) such that \( \bar{c}(\bar{x}) \leq \bar{L} \), for some \( \bar{L} \), and \( \epsilon > 0 \) a constant. Then \( \bar{x} \) can be converted to a vector \( \bar{x}' \) with at most \( k'' = O(\log |U|) \) fractional entries, such that \( \bar{c}(\bar{x}') \leq (1 + \epsilon)^3\bar{L} \), and \( F(\bar{x}') \geq F(\bar{x}) \), in polynomial time in \( |U| \).

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 \( F(\bar{x}) \geq \mathcal{O}/5 \).

---

\[\text{A detailed proof is given in Appendix C.}\]
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 \( F(\bar{x}) \geq (1 + \varepsilon)^{-2} F(\bar{x}'') \geq (1 - e^{-1} - O(\varepsilon))O \), as in Lemma 2.8.

3. Enumerate over all possible values \( D \) for \( D \sim \bar{x}'' \). For each such value if \( D \) is \( \varepsilon \)-nearly feasible convert it to a feasible solution \( D' \) (see Lemma 2.2). 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 \( \varepsilon \) is a fixed constant. Replacing the randomized rounding in the algorithm of Section 2.4 with the above we get the following result.

**Theorem 2.9** Let \( f \) be a submodular function, and assume we have an oracle for \( F \). If there is a deterministic polynomial time \( \alpha \)-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 \( \varepsilon > 0 \).

Since, given an oracle to \( F \), both the algorithms of [20] and [15] for the continuous problem are deterministic, we get the following.

**Theorem 2.10** 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 2.11** Given an oracle for \( F \), there is a polynomial time deterministic \((1/4 - \varepsilon)\)-approximation algorithm for \( SUB \) for any \( \varepsilon > 0 \).

### 3 The Budgeted Generalized Assignment Problem

In this section we apply our technique for solving a budgeted variant of the well-known generalized assignment problem (GAP). We start with a few definitions. An instance of the separable assignment problem (SAP) consists of \( n \) items \( A = \{a_1, \ldots, a_n\} \) and \( m \) bins. Each bin \( i \) has an associated collection of feasible sets \( T_i \) which is down-closed (\( S \in T_i \) implies \( S' \in T_i \) 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 \( S_i \in T_i \) 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 \( T_i \) are defined by a knapsack constraint. The best approximation for GAP is \((1 - e^{-1} + \varepsilon)\), for some \( \varepsilon > 0 \), due to [7]. Our algorithm uses the \((1 - e^{-1})\)-approximation for the problem given in [9].

We consider a budgeted variant of 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 global \( d_1 \)-dimensional budget vector \( \bar{L} \). The objective is to find a maximal profit solution whose total cost is at most \( \bar{L} \). A \((\frac{1}{2}e^{-1} - \varepsilon)\)-approximation was given in [13] for the case where \( d_1 = 1 \), as an example for the usage of a Lagrangian relaxation technique.

We give below an algorithm for a 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, rather than a single constraint. Formally, given are \( n \) items \( A = \{a_1, \ldots, a_n\} \) 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 \( i \) gained when \( a_j \) is assigned to bin \( i \). Also, given is a global \( d_1 \)-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 also define the cost and profit of \( S_i \) as assignment 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 \( S = (S_1, \ldots, S_m) \), such that each set \( S_i \) is a feasible assignment for bin \( i \). We define the cost of \( S \) by \( c(S) = \sum_{i=1}^m c(i, S_i) = \sum_{i=1}^m \sum_{a_j \in S_i} c_{i,j} \), 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 \( \bar{c}(S) \leq \bar{L} \) (its total cost is bounded by \( \bar{L} \)). The problem is to find a feasible solution of maximal profit.

As before, we say that a solution \( S \) is \( \varepsilon \)-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 \( c_{i,j} \leq \varepsilon^3 \cdot \bar{L} \); 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 \emph{small items BSAP}, in which all items are small; the second is \emph{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 \( i \).

\textbf{Overview} \quad Our algorithm proceeds in four stages. The first stage obtains an \( \varepsilon \)-nearly feasible solution with high profit for small assignments instances of BSAP, by using Theorem 2.1.

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 those bins in this solution. This guess is used to eliminate all big assignments, by forcing additional \( d_1 \) linear constraints for each bin. The result of this stage is a \( 2\varepsilon \)-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 into a feasible solution. The fourth and last stage gives a reduction from general instances to small items instances. This is done by a simple enumeration, as in Section 2.3. Due to space constraints, we describe the fourth stage in Appendix B.

\subsection{Small Assignments Instances of BSAP}
We solve BSAP instances with small assignments by casting BSAP as a submodular maximization problem. We note that the submodular interpretations of GAP and SAP are well-known (see, e.g., [7] and [9]). Let \( I_i \) be the set of feasible assignments for bin \( i \), and \( U = \{ (i, S_i) | S_i \in I_i \} \). Define \( f_j : 2^U \rightarrow \mathbb{R}_+ \) for any \( a_j \in A \) by \( f_j(V) = \max\{p_{i,j} | (i, S_i) \in V, a_j \in S_i\} \), where the maximum over an empty set is defined to be zero, and \( f : 2^U \rightarrow \mathbb{R}_+ \) by \( f(V) = \sum_{a_j \in A} f_j(V) \). It is easy to verify that \( f \) is a non-decreasing submodular function. Also, we define the \((d_1\text{-dimensional})\) 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 \( \bar{c}(V) = \sum_{(i,S_i) \in V} \bar{c}(i, S_i) \). Finally, we define a set \( M \) of all subsets in which there is no more than a single assignment set for the same bin. We can now consider the following instance of generalized SUB (as defined in Section 2.2): Maximize \( f(V) \) subject to the constraints \( V \in M \) and \( \bar{c}(V) \leq \bar{L} \).

We note that any set \( V \in M \) can be converted to a solution \( S \) for BSAP with \( \bar{c}(S) \leq \bar{c}(V) \) and \( p(S) = f(V) \). Similarly, every solution \( S \) for BSAP can be converted to a set \( V \in M \) such that \( \bar{c}(S) = \bar{c}(V) \) and \( p(S) = f(V) \).

Now, we use a technique of [9] to obtain a fractional solution for the submodular problem. Consider the linear program LP-BSAP over the variables \( X_{i}^{S} \), for all \( 1 \leq i \leq m \) and \( S \in I_i \). We note that the optimal solution for LP-BSAP is at least \( p(O) \), where \( O \) 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 \((1-\varepsilon)\) the optimal solution can be found in polynomial time (see [9] for more details). Let \( \bar{X} \) be such a solution, then the value of the solution \( \bar{X} \) is at least \((1-\varepsilon) \cdot p(O) \).
Let $D_i$ be a random variable over $I_i$ with $Pr[D_i = S] = X_i^S$. Define a random set $D = \{(1, D_1), (2, D_2), \ldots, (m, D_m)\}$, and let $\chi$ be the distribution of $D$. In [9] it was shown that $E[f(D)]$ is at least $(1 - \epsilon^{-1})$ times the value of the solution $X$, which means that $E[f(D)] \geq (1 - \epsilon^{-1}) \cdot (1 - \epsilon) \cdot p(O)$. It is easy to verify that the conditions of Theorem 2.1 hold for the distribution $\chi$. We define $D' = D$ if $D$ is $\epsilon$-nearly feasible and $D' = \emptyset$ otherwise. Then by Theorem 2.1 $D'$ is always $\epsilon$-nearly feasible and $E[f(D')] \geq (1 - \epsilon^{-1})(1 - O(\epsilon))p(O)$. As before, $D'$ can be converted to a solution $S$ for the BSAP instance, such that $p(S) = f(D')$, and $\tilde{c}(S) \leq \epsilon(D')$. We summarize the above steps in the following algorithm:

**Approximation algorithm for Small Assignments BSAP Instances**

1. Find a $(1 - \epsilon)$-approximate solution $X$ for LP-BASP.
2. For any bin $1 \leq i \leq m$, select an assignment $D_i = S_i$ with probability $X_i^S$ and define $D = \{(1, D_1), \ldots, (m, D_m)\}$.
3. If $D$ is $\epsilon$-nearly feasible return $D$ as the solution, else return an empty assignment.

**Lemma 3.1** The approximation algorithm for small assignments BSAP instances returns in polynomial time an $\epsilon$-nearly feasible solution with expected profit at least $(1 - \epsilon^{-1})(1 - O(\epsilon))p(O)$, where $O$ is an optimal solution.

### 3.2 Reduction into Small Assignments BSAP Instances

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

$$k_{i,r} \cdot \delta \cdot L_r \leq e^O(i) \leq (k_{i,r} + 1) \cdot \delta \cdot L_r \tag{1}$$

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 $\epsilon > 0$.

We use our guess for the set $T$ and the vectors $\tilde{k}_i$ 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, $L$, 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 $T'_i = \{S^S \in I_i, \tilde{c}(i, S_i) \leq \epsilon^3 L'_r\}$ and for any $i \in T$ the feasible set is
$I_i' = \{S | S \in I_i, \bar{c}(i, S) \leq (k_{i,r} + 1) \cdot \delta \cdot L_r \}$. 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. The relation between the solutions of the residual and original problems is stated in the next two lemmas.

**Lemma 3.2** Let $S = (S_1, \ldots, S_m)$ be an $\varepsilon$-nearly feasible solution for the residual problem with respect to a set $T$ (of size at most $h$) and a collection of vectors $k_i$. Then $S$ is a $(2\varepsilon)$-nearly feasible solution for the original problem with $p(S) = p'(S)$.

**Lemma 3.3** Let $T$ be the set of $h$ most profitable bins in an optimal solution $O$, and let $k_i$ be the set of cost guesses for which $(1)$ holds. Then there is a feasible solution $S'$ for the residual problem with respect to $T$ and $k_i$ of profit $p'(S') \geq (1 - \varepsilon)p(O)$.

**Proof Sketch:** Simply use all the assignments in $O$ which remain feasible with respect to the residual problem. The number of discarded assignment is bounded by $d_1 \cdot \varepsilon^{-3}$, and the profit of each of them is bounded by $p(O)/|T|$. Based on these two claims the profit bound can be attained.4

Now, consider 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 problem with respect to $T$ and $\bar{k}_i$, and run the 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 3.4** Let $O$ be an optimal solution for a BSAP instance. Then the Nearly Feasible Algorithm for BSAP returns a $(2\varepsilon)$-nearly feasible solution $S$ with expected profit at least $E[p(S)] \geq (1 - \varepsilon^{-1})(1 - O(\varepsilon))p(O)$ in polynomial time.

### 3.3 Small Items BSAP Instances

In this section we consider inputs in which all items are small. This allows us to fix a nearly feasible solution, similar to Lemma 2.2.

**Lemma 3.5** Let $S = (S_1, \ldots, S_m)$ be an $\varepsilon$-nearly feasible solution for a small items instance of BSAP. Then $S$ can be converted to a feasible solution $S'$ such that $p(S') \geq (1 - O(\varepsilon))p(S)$.

**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 3.5, and return $S'$.

The properties of the algorithm follow immediately from Lemmas 3.4 and 3.5.

**Lemma 3.6** The Algorithm for small items BSAP returns a feasible solution for the problem with expected profit of at least $(1 - \varepsilon^{-1})(1 - O(\varepsilon))p(O)$, where $O$ is an optimal solution.

Since any general instance for the problem can be reduced to a small items instance (see Appendix B, we have

**Theorem 3.7** There is a polynomial time randomized $(1 - \varepsilon^{-1} - \varepsilon)$-approximation algorithm for BSAP, for any fixed $\varepsilon > 0$.

---

4A detailed proof is given in Appendix C.
References


A Basic Properties of Submodular Functions

In this section we give a few 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 \( S, T \subseteq U \), and 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 \( S = S_1 \cup S_2 \cup \ldots \cup S_k \) where \( S_i \) are disjoint sets. Then

\[
f(S) \geq f(S_1) + f(S_2) + \ldots f(S_k)
\]

**Proof:** By induction on \( k \). For \( k = 2 \), since \( f \) is a submodular function,

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

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

For \( k > 2 \), by using the induction hypothesis twice, we have

\[
f(S) \leq f(S_1) + f(S_2) + \ldots f(S_{k-2}) + f(S_{k-1} \cup S_k) \leq f(S_1) + f(S_2) + \ldots f(S_k).
\]

**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 \( S = S_1 \cup S_2 \cup \ldots \cup S_k \), 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

\[
f(S) = \sum_{i=1}^{k} f_{S \cup \ldots \cup S_{i-1}}(S_i)
\]
By lemma A.2 we have that for each $i$, $f_{S \cup \ldots \cup S_{i-1}}(S_i) \geq f_{S \setminus S_i}(S_i)$. Hence we get
\[
 f(S) \geq \sum_{i=1}^{k} f_{S \setminus S_i}(S_i)
\]
As desired. \qed

**B General Inputs for BSAP**

In this section we use the algorithm for small items BSAP to obtain $(1 - e^{-1} - \varepsilon)$-approximation for arbitrary inputs. Given an input for BSAP, let $O = (S_1, \ldots, S_m)$ be an optimal solution.

Let $T^* = (T_1^*, \ldots, T_m^*)$ be a feasible assignment of elements to bins. We define a residual instance 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 were not assigned in $T^*$ to any bin. Consider the assignment of $a_j$ to bin $i$. If $\bar{c}_{i,j} \leq \varepsilon \bar{L}$, 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 $\bar{b}_{i}^* = \bar{b}_{i} - \sum_{a_j \in T_{i}^*} \bar{s}_{i,j}$.

The residual instance with respect to $T^*$ is small items instance. Any feasible assignment to the residual problem (with respect to $T^*$) of profit $v$ can be converted to a feasible assignment to 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 an optimal solution $O$ (the profit of $a_j$ in $O$ is $p_{i,j}$ if it was assigned to bin $i$ and zero otherwise). Then there is a solution to the residual problem of value $p(O) - (1 + \varepsilon)f(T^*)$ (the proof is similar to the proof of Lemma 3.3). Thus, if we guessed $T^*$ correctly, the algorithm for small items BSAP can be used to obtain a solution with expected profit of at least $(1 - e^{-1} - O(\varepsilon))(p(O) - (1 + \varepsilon)f(T^*))$ for the residual problem, which we can covert to a solution for the original instance of profit at least $(1 - e^{-1} - O(\varepsilon))p(O)$. And we get the following algorithm.

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

**C Some Proofs**

**Proof of Claim 2.1:** 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,
\[
 Var[c_r(D)] = \sum_{k=1}^{m} Var[c_r(D_k)] \leq \sum_{k \in V_r} E[c_r^2(D_k)] \leq \sum_{k \in V_r} E[c_r(D_k)|c_r^3 L_r \leq \varepsilon^3 L_r \sum_{k=1}^{m} E[c_r(D_k)] \leq \varepsilon^3 L_r^2.
\]
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,

$$Pr[c_r(D) \geq (1 + \varepsilon)L_r] = Pr[c_r(D) - E[c_r(D)] \geq (1 + \varepsilon)L_r - E[c_r(D)]]$$

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

By the union bound, we have that

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

\[\square\]

**Proof of Claim 2.2:** By the Chebyshev-Cantelli inequality we have that, for any dimension $1 \leq r \leq d$,

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

and by the union bound, we get that

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

\[\square\]

**Proof of Claim 2.3:** The set $D$ can be partitioned to $2d\ell$ sets $D_1, \ldots, D_{2d\ell}$ such that each of these sets is a feasible solution. Hence, $f(D_i) \leq O$. Thus by lemma A.1, $f(D) \leq f(D_1) + \ldots + f(D_{2d\ell}) \leq 2d\ell f(O)$. \[\square\]

**Proof of Claim 2.4:** By Claims 2.1 and 2.2, we have that

$$E[f(D)] = E[f(D)|F = 1] \cdot Pr[F = 1] + E[f(D)|F = 0 \land R < 2] \cdot Pr[F = 0 \land (R < 2)]$$

$$+ \sum_{\ell=1}^{\infty} E[f(D)|F = 0 \land (2^\ell \leq R \leq 2^{\ell+1})] \cdot Pr[F = 0 \land (2^\ell \leq R \leq 2^{\ell+1})]$$

$$\leq E[f(D)|F = 1] \cdot Pr[F = 1] + 4d^2 \varepsilon \cdot O + d^2 \varepsilon^3 \cdot O \cdot \sum_{\ell=1}^{\infty} \frac{2^{\ell+2}}{(2^\ell - 1)^2}.$$ 

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

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

14
Proof of Lemma 3.2: Clearly, $S$ is a solution for the original problem since for every bin $i$ it holds that $I_i' \subseteq I_i$. For any bin $i$ and $a_j \in A$ it holds that $p_{i,j} = p^i_{i,j}$, then clearly $p(S) = p'(S)$. As
for the cost, for any $1 \leq r \leq d_1$:

$$c_r(S) = \sum_{i \in T} c_r(i, S_i) + \sum_{i \notin T} c'_r(i, S_i)$$

$$\leq \sum_{i \in T} (k_{i,r} + 1) \cdot \delta L_r + \sum_{i \notin T} c'_r(i, S_i)$$

$$\leq |T| \cdot \delta L_r + \sum_{i \in T} k_{i,r} \cdot \delta L_r + \sum_{1 \leq i \leq m} c'_r(i, S_i)$$

$$\leq h \cdot \delta L_r + \sum_{i \in T} k_{i,r} \cdot \delta L_r + (1 + \varepsilon)L'$$

$$= \varepsilon \cdot L_r + \sum_{i \in T} c_r(i, S_i) \leq \sum_{i \notin T} c'_r(i, S_i) \leq L'_r,$$

Proof of Lemma 3.3: Let $O = (S_1, \ldots, S_m)$ be an optimal solution for the original problem. We define a solution $S' = (S'_1, \ldots, S'_m)$ for the residual problem by $S'_i = S_i$ if $S_i \in T_i'$ and $S'_i = \emptyset$ otherwise. Clearly, $S'$ is a solution for the residual problem. For any dimension $1 \leq r \leq d_1$,

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

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 conclude that

$$c'_r(S') = \sum_{i \in T} c'_r(i, S_i) + \sum_{i \notin T} c'_r(i, S_i) \leq \sum_{i \notin T} c_r(i, S_i) \leq L'_r,$$

i.e., $S'$ is a feasible solution.

Let $H$ be the set of bins for which $S_i \neq S'_i$. Since (1) holds for the vectors $\tilde{k}_i$, it is clear that for any $i \in T$, $S'_i = S_i$, and thus $H \cap T = \emptyset$. The total cost of bins not in $T$ is bounded by $L'$ (since $\sum_{i \notin T} c_r(i, S_i) \leq L'_r$). Hence, for all of them except at most $d_1 \cdot \varepsilon^{-3}$ it holds that $c_r(i) \leq \varepsilon^3 L'_r$, for some $1 \leq r \leq d_1$. This means that for every $i \notin T$ except at most $d_1 \cdot \varepsilon^{-3}$ it holds that $S_i \in T'_i$ and $S_i = S'_i$. From this, we conclude that $|H| \leq d_1 \cdot \varepsilon^{-3}$, and that none of the bins in $T$ is in $H$. The profit of any bin which is not in $T$ is smaller than $p(O)/|T|$ (since $T$ is the set of most profitable bins). Thus,

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

Thus,

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

Proof of Lemma 3.4: The properties of the algorithm for small assignments BSAP are described in Lemma 3.1. As the number of possibilities for $T$ and the vectors $\tilde{k}_i$ is polynomial for fixed values of $d_1, d_2$ and $\varepsilon$, and the 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 $\epsilon$-nearly feasible solution (for the residual problem), by Lemma 3.2, we get that the solutions output by the algorithm are always $(2\epsilon)$-nearly feasible for the original problem. Finally, in the iteration for which the set $T$ and vectors $k_i$ satisfy the requirements of Lemma 3.3, the optimal solution for the residual problem is of profit at least $(1 - \epsilon)p(O)$. Thus, the expected profit of the solution returned by the small assignments algorithm is at least $(1 - e^{-1})(1 - O(\epsilon))(1 - \epsilon)p(O) = (1 - e^{-1})(1 - O(\epsilon))p(O)$.

As the expected profit of the solution returned by the algorithm is at least the expected profit at any iteration, we get that the expected profit of the solution returned by the algorithm is $(1 - e^{-1})(1 - O(\epsilon))p(O)$.

\[\square\]

**D Approximation Algorithm for Maximum Coverage with Multiple Packing and Cost Constraints**

The problem of maximum coverage with multiple packing and cost constraints (CMPC) is the following generalization of the maximum coverage problem. Given is a collection of sets $S = \{S_1, ..., S_m\}$ over a ground set $A = \{a_1, ..., a_n\}$. Each element $a_j$ has a profit $p_j \geq 0$ and a $d_1$-dimensional size vector $\bar{s}_j = (s_{j,1}, ..., s_{j,d_1})$, 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 $\bar{w}_i = (w_{i,1}, ..., w_{i,d_2})$. Also given is a $d_1$-dimensional capacity vector $\bar{B} = (B_1, ..., B_{d_1})$, and a $d_2$-dimensional weight limit vector $\bar{W} = (W_1, ..., W_{d_2})$. A solution for the problem is a collection of subsets $H$ and a subset of elements $\mathcal{E}$, 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 of subsets in $H$ is bounded by $\bar{W}$ and the total size of elements in $\mathcal{E}$ is bounded by $\bar{B}$. The profit of a solution $(H, \mathcal{E})$ is the total profit of elements in $\mathcal{E}$. The objective of the problem is to find a feasible solution with maximal profit.

Denote the maximal number of sets a single element belongs to by $f$, and let $\alpha_f = 1 - \left(1 - \frac{1}{f}\right)^f$, Our objective is to obtain a $(\alpha_f - \epsilon)$-approximation algorithm for the problem. Note that for any $f \geq 1$ it holds that $\alpha_f > 1 - e^{-1}$.

For solving the problem, our algorithm uses a slightly different point of view. Given an input for the problem, we say that a pair $(\bar{y}, \bar{x})$ where $\bar{y} \in [0,1]^{S \times A}$ and $\bar{x} \in [0,1]^S$ 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 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 element 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 limit). 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 (or semi-integral) 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 problem 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 CMPC, and our objective is to find an optimal solution for the integral problem.
Overview To obtain approximation algorithm for the integral problem, we first reduce it to the semi-fractional problem. That is, we show that given a \((\alpha_f - O(\varepsilon))\)-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 submodular optimization problem with multiple linear constraints and an infinite universe. We 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 hold. Together with a fixing procedure for the nearly feasible solution obtained, this leads to the desired algorithm. The process can be derandomized by using the same tools as in Section 2.5.

D.1 Reduction to the semi-fractional problem

First, we show that a 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 \(\alpha\)-approximation algorithm for the semi-fractional problem.

Lemma D.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 \(p(\bar{y}) \geq p(\bar{y}^f)\) in polynomial time.

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

We use Lemma D.1 to prove the next result.

Lemma D.2 Given an \(\alpha\)-approximation algorithm for the semi-fractional problem, an \(\alpha\)-approximation algorithm for the integral problem can be derived in polynomial time.

Proof: Given a collection \(T\) of pairs \((a_i, S_i)\) of an element \(a_i\) 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 \(T_E\). We define a residual instance for the problem as follows. The elements are:

\[
A_T = \{a_j \in A | a_j \notin T_E, \text{ for any } a_j' \in T_E, p_j \leq p_{j'}\},
\]

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 \(S_T = \{S'_1, \ldots, S'_n\}\), where \(S'_i = S_i \cap A_T\), and \(\bar{w}'_i = \bar{w}_i\) if \(S'_i \notin T_S\) and \(\bar{w}'_i = 0\) if \(S'_i \in T_S\). The weight limit of this instance
is \( W_T = \bar{W} - w(T_S) \), where \( w(T_S) = \sum_{S_i \in T_S} \bar{w}_i \), and the capacity is \( B_T = \bar{B} - s(T_S) \), where \( s(T_S) = \sum_{a_j \in T_S} \bar{s}_j \).

Clearly, a solution of profit \( v \) for the residual instance with respect to a collection \( T \) gives a solution of profit \( v + p(T_S) \) for the original instance, where \( p(T_S) = \sum_{a_j \in T_S} p_j \). Let \( O = (\bar{x}, \bar{y}) \) be an optimal solution for the integral problem. W.l.o.g. we assume that for all \( a_j \in A \), \( \sum_{S_i \in S} y_{i,j} \leq 1 \) (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 is \( S_i \) such that \( y_{i,j} = 1 \) (note that there is a unique set \( S_i \) for each \( a_j \)). Define \( T^O = \{(a_j, S_i) | a_j \in R \wedge y_{i,j} = 1\} \). It is easy to verify that the optimal integral solution for the residual problem with respect to \( T^O \) is \( O - p(T_S^C) \).

Now, assume that we have an \( \alpha \)-approximation algorithm for the semi-fractional problem. Then it returns a fractional solution \( (\bar{y}^f, \bar{x}^f) \) with \( p(\bar{y}^f) \geq \alpha (O - p(T_S^C)) \). By Lemma D.1, this solution can be converted to a solution \( (\tilde{z}, \tilde{x}^f) \) with up to \( d_1 \) fractional entries and \( p(\tilde{z}) \geq p(\bar{y}^f) \). Now, consider rounding down to zero the value of each fractional entry in \( \tilde{z} \). This will generate a new feasible integral solution \( \tilde{z}'' \) with \( p(\tilde{z}'') \geq p(\bar{y}^f) - \frac{d_1 \cdot p(T_S^C)}{|T|} \) (as the profit of any element in the residual solution is bounded by \( \frac{p(T_S^C)}{|T|} \)). This implies a solution for the original problem of value at least

\[
p(T_S^C) + p(\bar{y}^f) - \frac{d_1 \cdot p(T_S^C)}{|T|} \geq p(T_S^C) \left( 1 - \frac{d_1}{|T|} \right) + \alpha (O - p(T_S^C)) \geq \alpha O
\]

That is, an \( \alpha \)-approximation for the optimum. To use this technique, we need to guess the correct set \( T \), which can be done in time \( (n \cdot m)^{O(1)} \) for a constant \( \alpha \).

In Theorem D.7 we show that there is a polynomial time \( (\alpha_f - \varepsilon) \)-approximation algorithm for the semi-fractional problem where

\[
\alpha_f = 1 - (1 - \frac{1}{f})^f,
\]

and \( f \) is the maximal number of sets to which a single element can belong. This lead to the following theorem.

**Theorem D.3** There is a polynomial time \( (\alpha_f - \varepsilon) \)-approximation algorithm for CMPC for any \( \varepsilon \), where \( f \) is the maximal number of sets to which a single element belongs.

### D.2 The Semi-fractional Problem

#### D.2.1 A submodular point of view

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

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

- For each \( S_i \in T \) and \( \bar{z} \in [0, 1]^A \) such that if \( a_j \notin S_i \) then \( z_j = 0 \), 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 $\bar{z} \in [0,1]^A$ such that
  1. if $a_j \notin S_i$ then $z_j = 0$,
  2. for any $1 \leq r \leq d_1$ it hold 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 hold 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 for the problem, $\bar{L}$, 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 $f_j : 2^U \rightarrow R$ for any $a_j \in A$, by

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

and $f : 2^U \rightarrow R$ 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 submodular optimization problem will imply 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 $\mathcal{O}$) be $s(S_i) = \sum_{a_j \in A} y_{i,j} \cdot \bar{s}_{j}$. We say a set $S_i \in S \setminus T$ is small if $s(S_i) \leq \varepsilon^3 \bar{B}$ and $\bar{w}_i \leq \varepsilon^3 \bar{W}'$. Consider the following solution for the submodular problem. 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

$$f(V) = \sum_{S_i \in T} p(S_i) + \sum_{S_i \in S \setminus T, S_i \text{ is small}} p(S_i) = \mathcal{O} - \sum_{S_i \in S \setminus T, S_i \text{ is not small}} p(S_i)$$

$$\geq \mathcal{O} - \frac{d \cdot \varepsilon^{-3}}{h} \mathcal{O} \geq (1 - \varepsilon) \mathcal{O},$$

where the last inequality holds since the number of not-small sets in $\mathcal{O}$ is bounded by $d \cdot \varepsilon^{-3}$, and the profit of each such set is bounded by $\mathcal{O}/h$. This means that the value of the optimal solution for the submodular problem is between $(1 - \varepsilon) \mathcal{O}$ to $\mathcal{O}$.

D.2.2 Obtaining a distribution on the universe of the submodular problem

We would like to use the technique of Section 2 on the submodular problem. To do so, we first need to obtain a fractional feasible solution. As the size of $U$ may be infinite, we cannot use the algorithm of Vondrák [20]. To obtain a fractional solution, we use 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:
\[(LP(T))\text{ maximize } \sum_{S_i \in S'} \sum_{a_j \in A} y_{i,j} \cdot p_j\]

subject to:

\[\forall a_j \in A, S_i \in S': 0 \leq y_{i,j} \leq x_i\]

\[\forall a_j \in A, S_i \in S', a_j \notin S_i: y_{i,j} = 0\]

\[\forall a_j \in A: \sum_{S_j \in S'} y_{i,j} \leq 1\]

\[\forall 1 \leq r \leq d_1: \sum_{S_i \in S' \setminus a_j \in A} y_{i,j} \cdot s_{j,r} \leq B_r\]

\[\forall 1 \leq r \leq d_2: \sum_{S_i \in S' \setminus T} y_{i,j} \leq W'_{r}\]

\[\forall S_i \in T:\]

\[\forall S_i \in S' \setminus T \text{ and } \forall 1 \leq r \leq d_1: \sum_{a_j \in A} y_{i,j} \cdot s_{j,r} \leq \varepsilon B_r \cdot x_i\]

Let \(\bar{x}\) be an optimal solution for \(LP(T)\) of value \(O\). Clearly, \(O\) is greater or equal to the optimal solution of the submodular problem; thus, by (4), \(O \geq (1-\varepsilon)O\). We use this solution to generate fractional solution for the submodular problem. Define \(X \in [0,1]^U\) as follows. For any \(S_i \in S'\) such that \(x_i > 0\), let \(\bar{z}_i \in [0,1]^A\) where \(z_{i,j} = \frac{y_{i,j}}{x_i}\), and we set \(X_i = X(S_i, \bar{z}_i) = x_i \). For any other \(u \in U\), set \(X_u = 0\). For any \(a_j \in A\), define \(Y_j = \sum_{S_i \in S'} x_i j\), then clearly \(O = \sum_{a_j \in A} Y_j \cdot p_j\).

**Lemma D.4** Let \(D\) be a random variable such that \(D \sim X\), then for any \(a_j \in A\), \(E[f_j(D)] \geq \alpha_f \cdot Y_j\), where \(f_j\) is defined in (3).

We use in the proof the next claim.

**Claim D.1** For any \(x \in [0,1]\) and \(f \in \mathbb{N}\),

\[1 - \left(1 - \frac{x}{f}\right)^f \geq x \cdot \alpha_f,\]

where \(\alpha_f\) is defined in (2).

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

**Proof of Lemma D.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 \(Pr[X_i = 1] = x_i \). Then,

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

Let \(S'[j] = \{S_i \in S': a_j \in S_i\}\) and \(\tau_j = |S'[j]|\). Let \(\delta \in (0,1)\) such that, for any \(S_i \in S'[j]\) with \(x_i \neq 0\), the value \(\frac{y_{i,j}}{x_i}\) is an integral multiple of \(\delta\), and \(1\) is also an integral multiple of \(\delta\) (assuming all values are rational, such \(\delta\) exists). Let \(H = \delta^{-1}\), and let \(Z_1, \ldots, Z_H\) 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 \( y_{i,j}^f \cdot \delta^{-1} \) indicators among \( Z_1, \ldots, Z_H \) 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 \( x_i \cdot \frac{y_{i,j}^f}{x_i} = y_{i,j}^f \) otherwise. Hence, we get that, for all \( 1 \leq h \leq H \),

\[
E[Z_h] = \Pr(Z_h = 1) = 1 - \prod_{S_i \in S'[j]} \left(1 - y_{i,j}^f\right)
\geq 1 - \left(\frac{1}{\tau_j} \sum_{S_i \in S'[j]} \left(1 - y_{i,j}^f\right)\right)^{\tau_j}
= 1 - \left(1 - \frac{Y_j}{\tau_j}\right)^{\tau_j} \geq Y_j \alpha_f.
\]

The first inequality follows from the inequality of the three means, and the second inequality follows from Claim D.1.

Let \( Y_j' = \delta \cdot \sum_{h=1}^{H} Z_h \) be \( \delta \) 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 (5), we have that

\[
E[Y_j'] = E\left[\sum_{h=1}^{H} Z_h \delta\right] = \delta \sum_{h=1}^{H} E[Z_h] \geq \delta \cdot H \cdot Y_j \alpha_f = Y_j \cdot \alpha_f.
\]

Hence, \( E[f_j(D)] \geq \alpha_f \cdot Y_j \) as desired. \( \Box \)

The next lemma follows immediately from Lemma D.4 and (4). Recall that \( F \) is an extension by expectation of \( f \), then

**Lemma D.5** \( F(\bar{X}) \geq \alpha_f \cdot O_f \geq \alpha_f \cdot (1 - \epsilon) \cdot \alpha_f \cdot O \)

It can be easily verified that \( X \) is a feasible fractional solution for the submodular problem. Also, for any \( (S_i, \bar{z}) \in U \) such that \( X_{(S_i, \bar{z})} \neq 0 \), if \( (S_i, \bar{z}) \) is big (as defined in Section 2.1), it holds that \( S_i \in T \), and \( X_{(S_i, \bar{z})} = 1 \). Define \( D \subseteq U \) to be a random set such that \( D \sim X \) and let \( D' \) be a random set such that \( D' = D \) if \( D \) is \( \epsilon \)-nearly feasible, and \( D' = \emptyset \) otherwise. By Theorem 2.1, we get that \( D' \) is always \( \epsilon \)-nearly feasible, and \( E[f(D')] \geq \alpha_f \cdot O \).

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

**Lemma D.6** Let \( D \) be an \( \epsilon \)-nearly feasible solution for the submodular problem, then \( D \) can be converted to a feasible solution \( D' \) such that \( f(D') \geq (1 - O(\epsilon)) f(D) \)

**Proof:** Our conversion will be done in two steps. First, let \( D_1 = \{(S_i, \bar{z}) | (S_i, \bar{z}) \in D\} \). 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, \( f(D_1) \geq (1 - \epsilon) f(D) \).

Next, we note that for any \( d_1 + 1 \leq r \leq d \), for any element \( u \in U \) it holds that \( c_{u,r} \leq \epsilon^3 L_r \). Thus, we can apply in these dimensions the fixing procedure of Lemma 2.2. Hence, we can convert \( D \) to \( D' \) as desired. \( \Box \)

This leads to the following algorithm.

**Randomized approximation algorithm for CMPC**
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 $\varepsilon$-nearly feasible, and $D' = \emptyset$ otherwise.

   (c) Convert $D'$ to a feasible set $D''$ as in Lemma D.6.

2. Let $D''$ be the solution of maximal profit found for the submodular problem. 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 $\mathcal{O}$. In this iteration $E[f(D')] \geq (1 - \varepsilon)\alpha_f \mathcal{O}$. Hence, by Lemma D.6 $E[f(D'')] \geq (1 - O(\varepsilon))\alpha_f \mathcal{O}$. Thus, we get the following (by a proper selection of $\varepsilon$).

**Theorem D.7** There is a polynomial time randomized $(1 - \varepsilon)\alpha_f$-approximation algorithm for the semi-fractional problem, for any fixed $\varepsilon > 0$.

The algorithm can be derandomized, using the technique in Section 2.5 (details omitted). 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.