A Tight Approximation for Submodular Maximization with Mixed Packing and Covering Constraints

Eyal Mizrachi
Computer Science Department, Technion, Haifa 32000, Israel
eyalmiz@cs.technion.ac.il

Roy Schwartz
Computer Science Department, Technion, Haifa 32000, Israel
schwartz@cs.technion.ac.il

Joachim Spoerhase
Department of Computer Science, Aalto University, Espoo, Finland
joachim.spoerhase@aalto.fi

Sumedha Uniyal
Department of Computer Science, Aalto University, Espoo, Finland
`https://users.aalto.fi/~uniyals1/`
sumedha.uniyal@aalto.fi

Abstract

Motivated by applications in machine learning, such as subset selection and data summarization, we consider the problem of maximizing a monotone submodular function subject to mixed packing and covering constraints. We present a tight approximation algorithm that for any constant $\varepsilon > 0$ achieves a guarantee of $1 - 1/e - \varepsilon$ while violating only the covering constraints by a multiplicative factor of $1 - \varepsilon$. Our algorithm is based on a novel enumeration method, which unlike previously known enumeration techniques, can handle both packing and covering constraints. We extend the above main result by additionally handling a matroid independence constraint as well as finding (approximate) pareto set optimal solutions when multiple submodular objectives are present. Finally, we propose a novel and purely combinatorial dynamic programming approach. While this approach does not give tight bounds it yields *deterministic* and in some special cases also considerably faster algorithms. For example, for the well-studied special case of only packing constraints (Kulik et al. [Math. Oper. Res. '13] and Chekuri et al. [FOCS '10]), we are able to present the first deterministic non-trivial approximation algorithm. We believe our new combinatorial approach might be of independent interest.

2012 ACM Subject Classification Theory of computation → Submodular optimization and polymatroids; Theory of computation → Approximation algorithms analysis

Keywords and phrases submodular function, approximation algorithm, covering, packing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.85

Category Track A: Algorithms, Complexity and Games

Related Version A full version of the paper is available at `https://arxiv.org/abs/1804.10947`.

Funding *Roy Schwartz*: Supported by ISF grant 1336/16 and NSF-BSF grant number 2016742.
Joachim Spoerhase: Supported by European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant number 759557) and by Academy of Finland (grant number 310415).
Sumedha Uniyal: Partially supported by Academy of Finland under the grant agreement number 314284.

Acknowledgements Joachim Spoerhase and Sumedha Uniyal thank an anonymous reviewer for pointing them to the fact that Theorem 6 also applies to polytopes that are not down-closed, which makes it possible to apply a randomized rounding approach.

1 Introduction

The study of combinatorial optimization problems with a submodular objective has attracted much attention in the last decade. A set function $f : 2^{\mathcal{N}} \to \mathbb{R}_+$ over a ground set $\mathcal{N}$ is called *submodular* if it has the *diminishing returns* property: $f(A \cup \{i\}) - f(A) \geq f(B \cup \{i\}) - f(B)$ for every $A \subseteq B \subseteq \mathcal{N}$ and $i \in \mathcal{N} \setminus B$.[1] Submodular functions capture the principle of economy of scale, prevalent in both theory and real world applications. Thus, it is no surprise that combinatorial optimization problems with a submodular objective arise in numerous disciplines, *e.g.*, machine learning and data mining [4, 5], algorithmic game theory and social networks [13, 22, 25, 27, 39], and economics [2]. Additionally, many classical problems in combinatorial optimization are in fact submodular in nature, *e.g.*, maximum cut and maximum directed cut [20, 21, 24, 26, 28], maximum coverage [15, 29], generalized assignment problem [8, 10, 14, 16], maximum bisection [3, 17], and facility location [1, 11, 12].

In this paper we consider the problem of maximizing a monotone[2] submodular function given mixed packing and covering constraints. In addition to being a natural problem in its own right, it has further real world applications.

As a motivating example consider the subset selection task in machine learning [18, 19, 30] (also refer to Kulesza and Taskar [31] for a thorough survey). In the subset selection task the goal is to select a diverse subset of elements from a given collection. One of the prototypical applications of this task is the document summarization problem [30, 34, 35]: given textual units the objective is to construct a short summary by selecting a subset of the textual units that is both representative and diverse. The former requirement, representativeness, is commonly achieved by maximizing a submodular objective function, *e.g.*, graph based [34, 35] or log subdeterminant [30]. The latter requirement, diversity, is typically tackled by penalizing the submodular objective for choosing similar textual units (this is the case for both of the above two mentioned submodular objectives). However, such an approach results in a submodular objective which is not necessarily non-negative thus making it extremely hard to cope with. As opposed to penalizing the objective, a remarkably simple and natural approach to tackle the diversity requirement is by the introduction of covering constraints. For example, one can require that for each topic that needs to appear in the summary, a sufficient number of textual units that refer to it are chosen. Unfortunately, to the best of our knowledge there is no previous work in the area of submodular maximization that incorporates general covering constraints.[3]

Let us now formally define the main problem considered in this paper. We are given a monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}_+$ over a ground set $\mathcal{N} = \{1, 2, \ldots, n\}$. Additionally, there are p packing constraints given by $\mathbf{P} \in \mathbb{R}_+^{p \times n}$, and c covering constraints given by $\mathbf{C} \in \mathbb{R}_+^{c \times n}$ (all entries of $\mathbf{P}$ and $\mathbf{C}$ are non-negative). Our goal is to find a subset $S \subseteq \mathcal{N}$ that satisfies all packing and covering constraints that maximizes the value of f:

$$\max \{f(S) : S \subseteq \mathcal{N}, \mathbf{P}\mathbf{1}_S \leq \mathbf{1}_p, \mathbf{C}\mathbf{1}_S \geq \mathbf{1}_c\}. \tag{1}$$

In the above $\mathbf{1}_S \in \mathbb{R}^n$ is the indicator vector for $S \subseteq \mathcal{N}$ and $\mathbf{1}_k \in \mathbb{R}^k$ is a vector of dimension k whose coordinates are all 1. We denote this problem as Packing-Covering Submodular Maximization (PCSM). It is assumed we are given a feasible instance, *i.e.*, there exists $S \subseteq \mathcal{N}$ such that $\mathbf{P}\mathbf{1}_S \leq \mathbf{1}_p$ and $\mathbf{C}\mathbf{1}_S \geq \mathbf{1}_c$.

---

[1] An equivalent definition is: $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$ for every $A, B \in \mathcal{N}$.
[2] f is monotone if $f(S) \leq f(T)$ for every $S \subseteq T \subseteq \mathcal{N}$.
[3] There are works on exact cardinality constraints for non-monotone submodular functions, which implies a special, uniform covering constraint [6, 33, 41].

As previously mentioned, (PCSM) captures several well known problems as a special case when only a single packing constraint is present ($p = 1$ and $c = 0$): maximum coverage [29], and maximization of a monotone submodular function given a knapsack constraint [40, 42] or a cardinality constraint [37]. For all of these special cases an approximation of $(1 - 1/e)$ is achievable and known to be tight [38] (even for the special case of a coverage function [15]). When a constant number of knapsack constraints is given ($p = O(1)$ and $c = 0$) Kulik et al. [32] presented a tight $(1 - 1/e - \varepsilon)$-approximation for any constant $\varepsilon > 0$. An alternative algorithm with the same guarantee was given by Chekuri et al. [9].

Our Results. We present a tight approximation guarantee for (PCSM) when the number of constraints is constant. Recall that we assume we are given a feasible instance, *i.e.*, there exists $S \subseteq \mathcal{N}$ such that $\mathbf{P1}_S \leq \mathbf{1}_p$ and $\mathbf{C1}_S \geq \mathbf{1}_c$. The following theorem summarizes our main result. From this point onwards we denote by O some fixed optimal solution to the problem at hand.

▶ **Theorem 1.** *For every constant $\varepsilon > 0$, assuming p and c are constants, there exists a randomized polynomial time algorithm for* (PCSM) *running in time $n^{poly(1/\varepsilon)}$ that outputs a solution $S \subseteq \mathcal{N}$ that satisfies:* (1) $f(S) \geq (1 - 1/e - \varepsilon)\, f(O)$*; and* (2) $\mathbf{P1}_S \leq \mathbf{1}_p$ *and* $\mathbf{C1}_S \geq (1 - \varepsilon)\mathbf{1}_c$.

We note four important remarks regarding the tightness of Theorem 1:

1. The loss of $1-1/e$ in the approximation cannot be avoided, implying that our approximation guarantee is (virtually) tight. The reason is that no approximation better than $1 - 1/e$ can be achieved even for the case where only a single packing constraint is present [38].
2. The assumption that the number of constraints is constant is unavoidable. The reason is that if the number of constraints is not assumed to be constant, then even with a linear objective (PCSM) captures the maximum independent set problem. Hence, no approximation better than $n^{-(1-\varepsilon)}$, for any constant $\varepsilon > 0$, is possible [23].[4]
3. No *true* approximation with a finite approximation guarantee is possible, *i.e.*, finding a solution $S \subseteq \mathcal{N}$ such that $\mathbf{P1}_S \leq \mathbf{1}_p$ and $\mathbf{C1}_S \geq \mathbf{1}_c$ with no violation of the constraints. The reason is that one can easily encode the subset sum problem using a single packing and a single covering constraint. Thus, just deciding whether a feasible solution exists, regardless of its cost, is already NP-hard.
4. Guaranteeing one-sided feasibility, *i.e.*, finding a solution which does not violate the packing constraints and a violates the covering constraint only by a factor of $1 - \varepsilon$, cannot be achieved in time $n^{o(1/\varepsilon)}$ unless the exponential time hypothesis fails (see [36] for details).

Therefore, we can conclude that our main result (Theorem 1) provides the best possible guarantee for the (PCSM) problem. We also note that all previous work on the special case of only packing constraints [9, 32] have the same running time of $n^{\text{poly}(1/\varepsilon)}$.

We present additional extensions of the above main result. The first extension deals with (PCSM) where we are also required that the output is an independent set in a given matroid $\mathcal{M} = (\mathcal{N}, \mathcal{I})$. We denote this problem by Matroid Packing-Covering Submodular Maximization (MatroidPCSM), and it is defined as follows:

[4] If the number of packing constraints p is super-constant then approximations are known only for special cases with "loose" packing constraints, *i.e.*, $P_{i,\ell} \leq O(\varepsilon^2 / \ln p)$ (see, *e.g.*, [9]).

$\max\{f(S) : S \subseteq \mathcal{N}, \mathbf{P1}_S \leq \mathbf{1}_p, \mathbf{C1}_S \geq \mathbf{1}_c, S \in \mathcal{I}\}$. As in (PCSM), we assume we are given a feasible instance, *i.e.*, there exists $S \subseteq \mathcal{N}$ such that $\mathbf{P1}_S \leq \mathbf{1}_p$, $\mathbf{C1}_S \geq \mathbf{1}_c$, and $S \in \mathcal{I}$. Our result is summarized in the following theorem.

▶ **Theorem 2.** *For every constant $\varepsilon > 0$, assuming p and c are constants, there exists a randomized polynomial time algorithm for* (MatroidPCSM) *that outputs a solution $S \in \mathcal{I}$ that satisfies:* (1) $f(S) \geq (1 - 1/e - \varepsilon)\, f(O)$*; and* (2) $\mathbf{P1}_S \leq \mathbf{1}_p$ *and* $\mathbf{C1}_S \geq (1-\varepsilon)\mathbf{1}_c$.

The second extension deals with the multi-objective variant of (PCSM) where we wish to optimize over several monotone submodular objectives. We denote this problem by Packing-Covering Multiple Submodular Maximization (MultiPCSM). Its input is identical to that of (PCSM) except that instead of a single objective f we are given t monotone submodular functions $f_1, \ldots, f_t : 2^{\mathcal{N}} \to \mathbb{R}_+$. As before, we assume we are given a feasible instance, *i.e.*, there exists $S \subseteq \mathcal{N}$ such that $\mathbf{P1}_S \leq \mathbf{1}_p$ and $\mathbf{C1}_S \geq \mathbf{1}_c$. Our goal is to find pareto set solutions considering the t objectives. To this end we prove the following theorem.

▶ **Theorem 3.** *For every constant $\varepsilon > 0$, assuming p, c and t are constants, there exists a randomized polynomial time algorithm for* (MultiPCSM) *that for every target values $v_1, \ldots, v_t$ either:* (1) *finds a solution $S \subseteq \mathcal{N}$ where $\mathbf{P1}_S \leq \mathbf{1}_p$ and $\mathbf{C1}_S \geq (1-\varepsilon)\mathbf{1}_c$ such that for every $1 \leq i \leq t$: $f_i(S) \geq (1 - 1/e - \varepsilon)\, v_i$; or* (2) *returns a certificate that there is no solution $S \subseteq \mathcal{N}$, where $\mathbf{P1}_S \leq \mathbf{1}_p$ and $\mathbf{C1}_S \geq \mathbf{1}_c$ such that for every $1 \leq i \leq t$: $f_i(S) \geq v_i$.*

We also note that Theorems 2 and 3 can be combined such that we can handle (MultiPCSM) where a matroid independence constraint is present, in addition to the given packing and covering constraints, achieving the same guarantees as in Theorem 3.

All our previously mentioned results employ a continuous approach and are based on the multilinear relaxation, and thus are inherently *randomized*.[5] We present a new combinatorial greedy-based dynamic programming approach for submodular maximization that enables us, for several well studied special cases of (PCSM), to obtain deterministic and considerably faster algorithms. Perhaps the most notable result is the first deterministic non-trivial algorithm for maximizing a monotone submodular function subject to a constant number of packing constraints (previous works [9, 32] are randomized).

▶ **Theorem 4.** *For every constants $\varepsilon > 0$ and $p \in \mathbb{N}$, there exists a deterministic algorithm for maximizing a monotone submodular function subject to p packing constraints, that runs in time $O(n^{\mathrm{poly}(1/\varepsilon)})$ and achieves an approximation of $1/e - \varepsilon$.*

The interesting special case of (PCSM) is when a single packing and a single covering constraints are present ($p = c = 1$) is summarized in the following theorem.

▶ **Theorem 5.** *For every constant $\varepsilon > 0$ and $p = c = 1$, there exists a deterministic algorithm for* (PCSM) *running in time $O(n^{1/\varepsilon})$ that outputs a solution $S \subseteq \mathcal{N}$ that satisfies:* (1) $f(S) \geq 0.353 f(O)$*; and* (2) $\mathbf{P1}_S \leq (1+\varepsilon)\mathbf{1}_p$ *and* $\mathbf{C1}_S \geq (1-\varepsilon)\mathbf{1}_c$*. For the case when the packing constraint is a cardinality constraint,* i.e.*, $\mathbf{P} = \mathbf{1}_n^\intercal/k$, we can further guarantee that $\mathbf{P1}_S \leq \mathbf{1}_p$ and a running time of $O(n^4/\varepsilon)$.*

Our Techniques. Our main result is based on a continuous approach: first a continuous relaxation is formulated, second it is (approximately) solved, and finally the fractional solution is rounded into an integral solution. Similarly to the previous works of [9, 32], which

[5] Known techniques to efficiently evaluate the multilinear extension are randomized, *e.g.*, [7].

focus on the special case of only packing constraints, the heart of the algorithm lies in an enumeration preprocessing phase that chooses and discards some of the elements prior to formulating the relaxation. The enumeration preprocessing step of [9, 32] is remarkably simple and elegant. It enumerates over all possible collections of large elements the optimal solution chooses, *i.e.*, elements whose size exceeds some fixed constant in at least one of the packing constraints and are chosen by the optimal solution.[6] All remaining large elements not in the guessed collection are discarded. This enumeration terminates in polynomial time and ensures that no large elements are left in any of the packing constraints. Thus, once no large elements remain concentration bounds can be applied. For the correct guess, any of the several known randomized rounding techniques can be employed (alongside a simple rescaling) to obtain an approximation of $1 - 1/e - \varepsilon$ (here $\varepsilon > 0$ is a constant that is used to determine which elements are considered large). Unfortunately, this approach fails in the presence of covering constraints since an optimal solution can choose *many* large elements in any given covering constraint. One can naturally adapt the above known preprocessing by enumerating over all possible collections of covering constraints that the optimal solution O covers using only large elements. However, this leads to an approximation of $1 - 1/e - \varepsilon$ while *both* packing and covering constraints are violated by a multiplicative factor of $1 \pm \varepsilon$. We aim to obtain one sided violation of the constraints, *i.e.*, only the covering constraints are violated by a factor of $1 - \varepsilon$ whereas the packing constraints are fully satisfied.

Avoiding constraint violation is possible in the presence of pure packing constraints [9, 32]. Known approaches for the latter are crucially based on *removing* elements in a pre-processing and post-processing step in order to guarantee that concentration bounds hold. For mixed constraints, these known removal operations may, however, arbitrarily violate the covering constraints. Our approach aims at pre-processing the input instance via partial enumeration so as to avoid discarding elements by ensuring that the remaining elements are "locally" small relatively to the *residual* constraints. If this property would hold scaling down the solution by a factor $1/(1+\varepsilon)$ would be sufficient to avoid violation of the packing constraints. Unfortunately, we cannot guarantee this to hold for all constraints. Rather, for some *critical* constraints locally large elements may still be present. We introduce a novel enumeration process that detects these critical constraints, *i.e.*, constraints that are prone to violation. Such constraints are given special attention as the randomized rounding might cause them to significantly deviate from the target value. Unlike the previously known preprocessing method, our enumeration process handles covering constraints with much care and it takes into account the *actual* coverage of the optimal solution O of each of the covering constraints. Combining the above, alongside a postprocessing phase that discards large elements from critical packing constraints, suffices to yield the desired result.

We also independently present a novel purely combinatorial greedy-based dynamic programming approach that yields deterministic and in some special cases considerably faster algorithms. Previously, greedy algorithms were known for one cardinality constraint [37] and one packing constraint [40]. But in the presence multiple constraints, it is not clear how to design a rule to greedily pick the next element. In fact, we tried several natural greedy strategies but all of them failed. This holds even when some oracle gives us the set of vectors of packing and covering values from an optimum solution and the algorithm follows any *fixed* sequence of these values.

[6] An additional part of the preprocessing involves enumerating over collections of elements whose marginal value is large with respect to the objective f, however this part of the enumeration is not affected by the presence of covering constraints and thus is ignored in the current discussion.

In our approach we maintain a table that contains greedy approximate solutions for all possible packing and covering values. Using this table we extend the simple greedy process by populating each table entry with the most profitable extension of the previous table entries. In this way we are able to simulate (in a certain sense) *all* possible sequences of packing and covering values for the greedy algorithm, ultimately leading to a good feasible solution. Our result implies that there exists one sequence (depending on the instance) where greedy performs well. To estimate the approximation factor we employ a factor-revealing linear program. To the best of our knowledge, this is the first time a dynamic programming based approach is used for submodular optimization. We believe our new combinatorial dynamic programming approach is of independent interest.

2 Preliminaries

In this paper we assume the standard *value oracle* model, where the algorithm can only access the given submodular function f with queries of the form: what is $f(S)$ for a given S? The running time of the algorithm is measured by the total number of value oracle queries and arithmetic operations it performs. Additionally, let us define $f_A(S) \triangleq f(A \cup S) - f(A)$ for any subsets $A, S \subseteq \mathcal{N}$. Furthermore, let $f_A(\ell) \triangleq f_A(\{\ell\})$.

The multiliear extension $F : [0,1]^{\mathcal{N}} \to \mathbb{R}_+$ of a given set function $f : 2^{\mathcal{N}} \to \mathbb{R}_+$ is:

$$F(\mathbf{x}) \triangleq \sum_{R \subseteq \mathcal{N}} f(R) \prod_{\ell \in R} x_\ell \prod_{\ell \notin R} (1 - x_\ell) \qquad \forall \mathbf{x} \in [0,1]^{\mathcal{N}}.$$

Additionally, we make use of the following theorem that provides the guarantees of the continuous greedy algorithm of [7].[7]

▶ **Theorem 6** (Chekuri et al. [7]). *We are given a ground set $\mathcal{N}$, a monotone submodular function $f : 2^{\mathcal{N}} \to \mathbb{R}_+$, and a polytope $\mathcal{P} \subseteq [0,1]^{\mathcal{N}}$. If $\mathcal{P} \neq \emptyset$ and one can solve in polynomial time $\operatorname{argmax}\{\mathbf{w}^T\mathbf{x} : \mathbf{x} \in \mathcal{P}\}$ for any $\mathbf{w} \in \mathbb{R}^{\mathcal{N}}$, then there exists a polynomial time algorithm that finds $\mathbf{x} \in \mathcal{P}$ where $F(\mathbf{x}) \geq (1 - 1/e)\, F(\mathbf{x}^*)$. Here $\mathbf{x}^*$ is an optimal solution to the problem:* $\max\{F(\mathbf{y}) : \mathbf{y} \in \mathcal{P}\}$.

3 Algorithms for the (PCSM) Problem

Preprocessing – Enumeration with Mixed Constraints. We define a *guess* D to be a triplet $(E_0, E_1, \mathbf{c}')$, where $E_0 \subseteq \mathcal{N}$ denotes elements that are discarded, $E_1 \subseteq \mathcal{N}$ denotes elements that are chosen, and $\mathbf{c}' \in \mathbb{R}_+^c$ represents a rough estimate (up to a factor of $1 + \varepsilon$) of how much an optimal solution O covers each of the covering constraints, *i.e.*, $\mathbf{C}\mathbf{1}_O$. Let us denote by $\tilde{\mathcal{N}} \triangleq \mathcal{N} \setminus (E_0 \cup E_1)$ the remaining undetermined elements with respect to guess D.

We would like to define when a given fixed guess $D = (E_0, E_1, \mathbf{c}')$ is *consistent*, and to this end we introduce the notion of *critical* constraints. For the i^{th} packing constraint the residual value that can still be packed is: $(\mathbf{r}_D)_i \triangleq 1 - \sum_{\ell \in E_1} \mathbf{P}_{i,\ell}$, where $\mathbf{r}_D \in \mathbb{R}^p$. For the j^{th} covering constraint the residual value that still needs to be covered is: $(\mathbf{s}_D)_j \triangleq \max\left\{0, \mathbf{c}'_j - \sum_{\ell \in E_1} \mathbf{C}_{j,\ell}\right\}$, where $\mathbf{s}_D \in \mathbb{R}^c$. A packing constraint i is called *critical* if $(\mathbf{r}_D)_i \leq \delta$, and a covering constraint j is called *critical* if $(\mathbf{s}_D)_j \leq \delta \mathbf{c}'_j$ ($\delta \in (0,1)$ is

[7] We note that the actual guarantee of the continuous greedy algorithm is $(1 - 1/e - o(1))$. However, for simplicity of presentation, we can ignore the $o(1)$ term due to the existence of a loss of ε (for any constant ε) in all of our theorems.

a parameter to be chosen later). Thus, the collections of critical packing and covering constraints, for a given guess D, are given by: $Y_D \triangleq \{i = 1, \ldots, p : (\mathbf{r}_D)_i \leq \delta\}$ and $Z_D \triangleq \{j = 1, \ldots, c : (\mathbf{s}_D)_j \leq \delta \mathbf{c}'_j\}$. Moreover, elements are considered *large* if their size is at least some factor α of the residual value of some non-critical constraint ($\alpha \in (0, 1)$ is a parameter to be chosen later). Formally, the collection of large elements with respect to the packing constraints is defined as $P_D \triangleq \{\ell \in \tilde{\mathcal{N}} : \exists i \notin Y_D \text{ s.t. } \mathbf{P}_{i,\ell} \geq \alpha(\mathbf{r}_D)_i\}$, and the collection of large elements with respect to the covering constraints is defined as $C_D = \{\ell \in \tilde{\mathcal{N}} : \exists j \notin Z_D \text{ s.t. } \mathbf{C}_{j,\ell} \geq \alpha(\mathbf{s}_D)_j\}$. It is important to note, as previously mentioned, that the notion of a large element is with respect to the residual constraint, as opposed to previous works [9, 32] where the definition is with respect to the original constraint. Let us now formally define when a guess D is called *consistent*.

▶ **Definition 1.** *A guess $D = (E_0, E_1, \mathbf{c}')$ is* consistent *if:* (1) $E_0 \cap E_1 = \emptyset$; (2) $\mathbf{c}' \geq \mathbf{1}_c$; (3) $\mathbf{P}\mathbf{1}_{E_1} \leq \mathbf{1}_p$; *and* (4) $P_D = C_D = \emptyset$.

Intuitively, requirement (1) states that a variable cannot be both chosen and discarded, (2) states that the each covering constraint is satisfied by an optimal solution O, (3) states the chosen elements E_1 do not violate the packing constraints, and (4) states that no large elements remain in any non-critical constraint.

Finally, we need to define when a consistent guess is *correct*. Assume without loss of generality that $O = \{o_1, \ldots, o_k\}$ and the elements of O are ordered greedily: $f_{\{o_1,\ldots,o_i\}}(o_{i+1}) \leq f_{\{o_1,\ldots,o_{i-1}\}}(o_i)$ for every $i = 1, \ldots, k-1$. In the following definition γ is a parameter to be chosen later.

▶ **Definition 2.** *A consistent guess $D = (E_0, E_1, \mathbf{c}')$ is called* correct *with respect to O if:* (1) $E_1 \subseteq O$; (2) $E_0 \subseteq \bar{O}$; (3) $\{o_1, \ldots, o_\gamma\} \subseteq E_1$; *and* (4) $\mathbf{c}' \leq \mathbf{C}\mathbf{1}_O \leq (1+\delta)\mathbf{c}'$.

Intuitively, requirement (1) states that the chosen elements E_1 are indeed elements of O, (2) states that no element of O is discarded, (3) states that the γ elements of largest marginal value are all chosen, and (4) states that $\mathbf{c}'$ represents (up to a factor of $1+\delta$) how much O actually covers each of the covering constraints.

We are now ready to present our preprocessing algorithm (Algorithm 1), which produces a list $\mathcal{L}$ of consistent guesses that is guaranteed to contain at least one guess that is also correct with respect to O. Lemma 7 summarizes this, its proof appears in [36].

Algorithm 1: Preprocessing.

```
L ← ∅
foreach j_1, ..., j_c ∈ {0, 1, ..., ⌈log_{1+δ} n⌉} do
    Let c' = ((1+δ)^{j_1}, ..., (1+δ)^{j_c})
    foreach E_1 ⊆ N such that |E_1| ≤ γ + (p+c)/(αδ) do
        Let H = (∅, E_1, c')
        Let E_0 = {ℓ ∈ N \ E_1 : f_{E_1}(ℓ) > (γ^{-1}) f(E_1)} ∪ P_H ∪ C_H
        Set D = (E_0, E_1, c')
        If D is consistent according to Definition 1 add it to L.
Output L.
```

▶ **Lemma 7.** *The output $\mathcal{L}$ of Algorithm 1 contains at least one guess D that is correct with respect to some optimal solution O.*

Proof. Fix any optimal solution O. At least one of the vectors $\mathbf{c}'$ enumerated by Algorithm 1 satisfies property (4) in Definition 2 with respect to O. Let us fix an iteration in which such a $\mathbf{c}'$ is enumerated. Define the "large" elements O has with respect to this $\mathbf{c}'$:

$$O_L \triangleq \{\ell \in O : \exists i \text{ s.t. } \mathbf{P}_{i,\ell} \geq \alpha\delta\} \cup \left\{\ell \in O : \exists j \text{ s.t. } \mathbf{C}_{j,\ell} \geq \alpha\delta\mathbf{c}'_j\right\}. \tag{2}$$

Denote by $O_\gamma \triangleq \{o_1, \ldots, o_\gamma\}$ the γ elements of O with the largest marginal (recall the ordering of O satisfies: $f_{\{o_1,\ldots,o_i\}}(o_{i+1}) \leq f_{\{o_1,\ldots,o_{i-1}\}}(o_i)$). Let us fix $E_1 \triangleq O_\gamma \cup O_L$ and choose $H \triangleq (\emptyset, E_1, \mathbf{c}')$. Clearly, $|E_1| \leq \gamma + {}^{(p+c)}/_{(\alpha\delta)}$ since $|O_\gamma| = \gamma$ and $|O_L| \leq {}^{(p+c)}/_{(\alpha\delta)}$. Hence, we can conclude that H is considered by Algorithm 1.

We fix the iteration in which the above H is considered and show that the resulting $D = (E_0, E_1, \mathbf{c}')$ of this iteration is correct and consistent (recall that Algorithm 1 chooses $E_0 = \{\ell \in \mathcal{N} \setminus E_1 : f_{E_1}(\ell) > (\gamma^{-1})f(E_1)\} \cup P_H \cup C_H$). The following two observations suffice to complete the proof:

Observation 1: $\forall \ell \in O \cup (\mathcal{N} \setminus E_0)$: $f_{E_1}(\ell) \leq \gamma^{-1} f(E_1)$.

Observation 2: $O \cap P_H = \emptyset$ and $O \cap C_H = \emptyset$.

Clearly properties (1) and (3) of Definition 2 are satisfied by construction of E_1, H, and subsequently D. Property (2) of Definition 2 requires the above two observations, which together imply that no element of O is added to E_0 by Algorithm 1. Thus, all four properties of Definition 2 are satisfied, and we focus on showing that the above D is consistent according to Definition 1. Property (1) of Definition 1 follows from properties (1) and (2) of Definition 2. Property (2) of Definition 1 follows from the choice of $\mathbf{c}'$. Property (3) of Definition 1 follows from the feasibility of O and property (1) of Definition 2. Lastly, property (4) of Definition 1 follows from the fact that $P_D \subseteq P_H$ and that $P_H \subseteq E_0$, implying that $P_D = \emptyset$ (the same argument applies to C_D). We are left with proving the above two observations.

We start with proving the first observation. Let $\ell \in O \cup (\mathcal{N} \setminus E_0)$. If $\ell \in \mathcal{N} \setminus E_0$ then the observation follows by the construction of E_0 in Algorithm 1. Otherwise, $\ell \in O$. If $\ell \in O_\gamma$ then we have that $f_{E_1}(\ell) = 0$ since $O_\gamma \subseteq E_1$. Otherwise $\ell \in O \setminus O_\gamma$. Note:

$$f_{E_1}(\ell) \leq f_{O_\gamma}(\ell) \leq \gamma^{-1} f(O_\gamma) \leq \gamma^{-1} f(E_1).$$

The first inequality follows from diminishing returns and $O_\gamma \subseteq E_1$. The third and last inequality follows from the monotonicity of f and $O_\gamma \subseteq E_1$. Let us focus on the second inequality, and denote $O = \{o_1, \ldots, o_k\}$ and the sequence $a_i \triangleq f_{\{o_1,\ldots,o_{i-1}\}}(o_i)$. The sequence of a_is is monotone non-increasing by the ordering of O and the monotonicity of f implies that all a_is are non-negative. Note that $a_1 + \ldots + a_\gamma = f(O_\gamma)$, thus implying that $f_{O_\gamma}(\ell) \leq \gamma^{-1} f(O_\gamma)$ for every $\ell \in \{o_{\gamma+1}, \ldots, o_k\}$ (otherwise $a_1 + \ldots + a_\gamma > f(O_\gamma)$). The second inequality above, *i.e.*, $f_{O_\gamma}(\ell) \leq \gamma^{-1} f(O_\gamma)$, now follows since $\ell \in O \setminus O_\gamma = \{o_{\gamma+1}, \ldots, o_k\}$.

Let us now focus on proving the second observation. Let us assume on the contrary that there is an element ℓ such that $\ell \in O \cap P_H$. Recall that $P_H = \{\ell \in \mathcal{N} \setminus E_1 : \exists i \notin Y_H \text{ s.t. } \mathbf{P}_{i,\ell} \geq \alpha(\mathbf{r}_H)_i\}$ where $Y_H = \{i : (\mathbf{r}_H)_i \leq \delta\}$. This implies that $\ell \in O \setminus E_1$, namely that $\ell \notin O_L$, from which we derive that for *all* packing constraint i we have that $\mathbf{P}_{i,\ell} < \alpha\delta$. Since $\ell \in P_H$ we conclude that there exists a packing constraint i for which $(\mathbf{r}_H)_i \leq \mathbf{P}_{i,\ell}/\alpha$. Combining the last two bounds we conclude that $(\mathbf{r}_H)_i < \delta$, which implies that the i^{th} packing constraint is critical, *i.e.*, $i \in Y_H$. This is a contradiction, and hence $O \cap P_H = \emptyset$. A similar proof applies to C_H and the covering constraints. □ ◀

Randomized Rounding. Before presenting our main rounding algorithm, let us define the residual problem we are required to solve given a consistent guess D. First, the residual objective $g : 2^{\tilde{\mathcal{N}}} \to \mathbb{R}_+$ is defined as: $g(S) \triangleq f(S \cup E_1) - f(E_1)$ for every $S \subseteq \tilde{\mathcal{N}}$. Clearly, g

is submodular, non-negative, and monotone. Second, let us focus on the feasible domain and denote by $\tilde{\mathbf{P}}$ ($\tilde{\mathbf{C}}$) the submatrix of $\mathbf{P}$ ($\mathbf{C}$) obtained by choosing all the columns in $\tilde{\mathcal{N}}$. Hence, given $D = (E_0, E_1, \mathbf{c}')$ the residual problem is:

$$\max\{g(S) + f(E_1) : S \subseteq \tilde{\mathcal{N}}, \tilde{\mathbf{P}}\mathbf{1}_S \leq \mathbf{r}_D, \tilde{\mathbf{C}}\mathbf{1}_S \geq \mathbf{s}_D\}. \tag{3}$$

In order to formulate the multilinear relaxation of (3), consider the following two polytopes: $\mathcal{P} \triangleq \{\mathbf{x} \in [0,1]^{\tilde{\mathcal{N}}} : \tilde{\mathbf{P}}\mathbf{x} \leq \mathbf{r}_D\}$ and $\mathcal{C} \triangleq \{\mathbf{x} \in [0,1]^{\tilde{\mathcal{N}}} : \tilde{\mathbf{P}}\mathbf{x} \geq \mathbf{s}_D\}$. Let $G : [0,1]^{\tilde{\mathcal{N}}} \to \mathbb{R}_+$ be the multilinear extension of g. Thus, the continuous multilinear relaxation of (3) is:

$$\max\left\{f(E_1) + G(\mathbf{x}) : \mathbf{x} \in [0,1]^{\tilde{\mathcal{N}}}, \mathbf{x} \in \mathcal{P} \cap \mathcal{C}\right\}. \tag{4}$$

Our algorithm performs randomized rounding of a fractional solution to the above relaxation (4). However, this is not enough to obtain our main result and an additional post-processing step is required in which additional elements are discarded. Since covering constraints are present, one needs to perform the post-processing step in great care. To this end we denote by L_D the collection of large elements with respect to some critical packing constraint: $L_D \triangleq \{\ell \in \tilde{\mathcal{N}} : \exists i \in Y_D \text{ s.t. } \mathbf{P}_{i,\ell} \geq \beta \mathbf{r}_D\}$ ($\beta \in (0,1)$ is a parameter to be chosen later). Intuitively, we would like to discard elements in L_D since choosing any one of those will incur a violation of a packing constraint. We are now ready to present our rounding algorithm (Algorithm 2).

Algorithm 2: $(f, \mathcal{N}, \mathbf{P}, \mathbf{C})$.

1 Use Algorithm 1 to obtain a list of guesses $\mathcal{L}$.
2 **foreach** $D = (E_0, E_1, \mathbf{c}') \in \mathcal{L}$ **do**
3 Use Theorem 6 to compute an approximate solution $\mathbf{x}^*$ to problem (4).
4 Scale down $\mathbf{x}^*$ to $\bar{\mathbf{x}} = \mathbf{x}^*/(1+\delta)$
5 Let R_D be such that for every $\ell \in \tilde{\mathcal{N}}$ independently: $\Pr[\ell \in R_D] = \bar{\mathbf{x}}_\ell$.
6 Let $R'_D = R_D \setminus L_D$.
7 $S_D \leftarrow E_1 \cup R'_D$.
8 $S_{alg} \leftarrow \operatorname{argmax}\{f(S_D) : D \in \mathcal{L}, \mathbf{P} \cdot \mathbf{1}_{S_D} \leq \mathbf{1}_p, \mathbf{C} \cdot \mathbf{1}_{S_D} \geq (1-\varepsilon)\mathbf{1}_c\}$

We note that Line 6 of Algorithm 2 is the post-processing step where all elements of L_D are discarded. Our analysis of Algorithm 2 shows that in an iteration a correct guess D is examined, with a constant probability, S_D satisfies the packing constraints, violates the covering constraint by only a fraction of ε, and $f(S_D)$ is sufficiently high.

The following lemma gives a lower bound on the value of the fractional solution $\bar{\mathbf{x}}$ computed by Algorithm 2 (for a full proof refer [36]).

▶ **Lemma 8.** *If $D \in \mathcal{L}$ is correct then in the iteration of Algorithm 2 it is examined the resulting $\bar{\mathbf{x}}$ satisfies: $G(\bar{\mathbf{x}}) \geq (1 - 1/e - \delta)f(O) - f(E_1)$.*

Let us now fix an iteration of Algorithm 2 for which D is not only consistent but also correct (the existence of such an iteration is guaranteed by Lemma 7). Intuitively, Algorithm 2 performs a straightforward randomized rounding where each element $\ell \in \tilde{\mathcal{N}}$ is independently chosen with a probability that corresponds to its fractional value in the solution of the multilinear relaxation (4). However, two key ingredients in Algorithm 2 are required in order to achieve an ε violation of the covering constraints and no violation of the packing constraints: (1) *scaling*: prior to the randomized rounding $\mathbf{x}^*$ is scaled down by a factor $(1+\delta)$ (line 4 in Algorithm 2); and (2) *post-processing*: after the randomized rounding all chosen large elements in a critical packing constraint are discarded (line 6 in Algorithm 2).

The first ingredient above (scaling of $\mathbf{x}^*$) allows us to prove using standard concentration bounds that with good probability all non-critical packing constraints are not violated. However, when considering critical packing constraints this does not suffice and the second ingredient above (discarding L_D) is required to show that with good probability even the critical packing constraints are not violated. While discarding L_D is beneficial when considering packing constraints, it might have a destructive effect on both the covering constraints and the value of the objective. To remedy this we argue that with high probability only few elements in L_D are actually discarded, *i.e.*, $|R_D \cap L_D|$ is sufficiently small. Combining the latter fact with the assumption that the current guess D is not only consistent but also correct, according to Definition 2, allows us to prove the following lemma (for a full proof refer to [36]).

▶ **Lemma 9.** *For any constant $\varepsilon > 0$, choose constants $\alpha = \delta^3$, $\beta = \delta^2/(3b)$, $\gamma = 1/\delta^3$, and $\delta < \min\{1/(15(p+c)), \varepsilon/(2+30(p+c)^2)\}$. With a probability of at least $1/2$ Algorithm 2 outputs a solution S_{alg} satisfying: (1) $\mathbf{P}\mathbf{1}_{S_{alg}} \leq \mathbf{1}_p$; (2) $\mathbf{C}\mathbf{1}_{S_{alg}} \geq (1-\varepsilon)\mathbf{1}_c$; and (3) $f(S_{alg}) \geq (1 - 1/e - \varepsilon)f(O)$.*

The above lemma suffices to prove Theorem 1, as it immediately implies it.

4 Greedy Dynamic Programming

In this section, we present a novel algorithmic approach for submodular maximization that leads to *deterministic* and considerably faster approximation algorithms in several settings. Perhaps the most notable application of our approach is Theorem 4. To the best of our knowledge, it provides the first *deterministic* non-trivial approximation algorithm for maximizing a monotone submodular function subject to packing constraints. To highlight the core idea of our approach, we first present a vanilla version of the greedy dynamic programming approach applied to (PCSM) that gives a constant-factor approximation and satisfies the packing constraints, but violates the covering constraints by a factor of 2 and works in pseudo-polynomial time.

Vanilla Greedy Dynamic Programming. Let us start with a sketch of the algorithm's definition and analysis. For simplicity of presentation, we assume in the current discussion relating to pseudo-polynomial time algorithms that $\mathbf{C} \in \mathbb{N}_+^{c \times n}$ and $\mathbf{P} \in \mathbb{N}_+^{p \times n}$. Let $\mathbf{p} \in \mathbb{N}_+^p$ and $\mathbf{c} \in \mathbb{N}_+^c$ be the packing and covering requirements, respectively. A solution $S \subseteq \mathcal{N}$ is feasible if and only if $\mathbf{C} \cdot \mathbf{1}_S \geq \mathbf{c}$ and $\mathbf{P} \cdot \mathbf{1}_S \leq \mathbf{p}$. We also use the following notations: $c_{\max} = \|\mathbf{c}\|_\infty$, $p_{\max} = \|\mathbf{p}\|_\infty$, and $[s]_0 = \{0, \ldots, s\}$ for every integer s.

We define our dynamic programming as follows: for every $q \in [n]_0$, $\mathbf{c}' \in [n \cdot c_{\max}]_0^c$, and $\mathbf{p}' \in [p_{\max}]_0^p$ a table entry $T[q, \mathbf{c}', \mathbf{p}']$ is defined and it stores an approximate solution S of cardinality q with $\mathbf{C} \cdot \mathbf{1}_S = \mathbf{c}'$ and $\mathbf{P} \cdot \mathbf{1}_S = \mathbf{p}'$. [8] For the base case, we set $T[0, \mathbf{0}_c, \mathbf{0}_p] \leftarrow \varnothing$. For populating $T[q, \mathbf{c}', \mathbf{p}']$ when $q > 0$, we examine every set of the form $T[q-1, \mathbf{c}' - \mathbf{C}_\ell, \mathbf{p}' - \mathbf{P}_\ell] \cup \{\ell\}$, where ℓ satisfies $\ell \in \mathcal{N} \setminus T[q-1, \mathbf{c}' - \mathbf{C}_\ell, \mathbf{p}' - \mathbf{P}_\ell]$, $\mathbf{c}' - \mathbf{C}_\ell \geq 0$, and $\mathbf{p}' - \mathbf{P}_\ell \geq 0$. Out of all these sets, we assign the most valuable one to $T[q, \mathbf{c}', \mathbf{p}']$. Note that this operation stores a *greedy approximate* solution in the table entry $T[q, \mathbf{c}', \mathbf{p}']$. The output of our algorithm is the best of the solutions $T[q, \mathbf{c}', \mathbf{p}']$, for $1 \leq q \leq n$, $\mathbf{c}' \geq \mathbf{c}/2$ and $\mathbf{p}' \leq \mathbf{p}$. See Algorithm 3 for pseudo code.

[8] We introduce a dummy solution $\perp$ for denoting undefined table entries, and initialize the entire table with $\perp$. For the exact details we refer to [36].

Algorithm 3: Vanilla Greedy Dynamic Program.

create a table $T\colon [n]_0 \times [n \cdot c_{\max}]_0^c \times [p_{\max}]_0^p \to 2^{\mathcal{N}}$ initialized with entries $\perp$
$T[0, \mathbf{0}_c, \mathbf{0}_p] \leftarrow \varnothing$
for $q = 0$ **to** n **do**
  foreach $\mathbf{c}' \in [n \cdot c_{\max}]_0^c$ *and* $\mathbf{p}' \in [p_{\max}]_0^p$ **do**
    foreach $\ell \in \mathcal{N} \setminus T[q, \mathbf{c}', \mathbf{p}']$ **do**
      $\mathbf{c}'' \leftarrow \mathbf{c}' + \mathbf{C}_\ell$, $\mathbf{p}'' \leftarrow \mathbf{p}' + \mathbf{P}_\ell$
      $T[q+1, \mathbf{c}'', \mathbf{p}''] \leftarrow \arg\max\{f(T[q+1, \mathbf{c}'', \mathbf{p}'']), f(T[q, \mathbf{c}', \mathbf{p}'] \cup \{\ell\})\}$
Output $\operatorname{argmax}_{q, \mathbf{c}' \geq \mathbf{c}/2, \mathbf{p}' \leq \mathbf{p}} f(T[q, \mathbf{c}', \mathbf{p}'])$.

Let us now sketch the analysis of the above algorithm. Let O be an optimal set solution. We consider an arbitrary permutation of O, say $\{o^1, o^2, \ldots, o^k\}$. Let $O^i = \{o^1, \ldots, o^i\}$ be the set of the first i elements in this permutation and let $O^0 = \emptyset$. We introduce the function $g\colon O \to \mathbb{R}_+$ for denoting the *marginal* value of the elements in O. More precisely, let $g(o^i) = f_{O^{i-1}}(o^i)$. Note that $f(O) = \sum_{\ell \in O} g(\ell)$. Let for any subset $S \subseteq O$, $g(S) = \sum_{\ell \in S} g(\ell)$. We then inductively construct an order $o_1, \ldots, o_k$ of O with the intention of upper bounding for every prefix $O_q = \{o_1, \ldots, o_q\}$ the value $g(O_q)$ in terms of the value $f(S_q)$ of the table entry $S_q := T[q, \mathbf{C1}_{O_q}, \mathbf{P1}_{O_q}]$ *corresponding to* O_q. The construction of the sequence $o_1, \ldots, o_k$ divides $[k]$ into m phases where m is a positive integer parameter. A (possibly empty) phase $i \in [m]$ is characterized by the following property. Consider a prefix O_q and its corresponding table entry S_q. If q is in phase i then there exists an element $o_{q+1} \in O \setminus O_q$ such that adding o_{q+1} to S_q increases f by at least an amount of $(1 - i/m)g(o_{q+1})$. We set $O_{q+1} = O_q \cup \{o_{q+1}\}$. Thus, in earlier phases we make more progress in the corresponding dynamic programming solution S_q relative to $g(O_q)$ than in later phases. Additionally, we can prove a complementing inequality. At the end of phase $i \in [m]$ all elements in $O \setminus O_q$ increase f by no more than $(1 - i/m)g(o_{q+1})$. We prove that this implies that $f(S_q)$ is at least $i/m \cdot g(O \setminus O_q)$ and thus large relatively to the *complement* of O_q. We set up a factor-revealing linear program that constructs the worst distribution of the marginal values over the phases that satisfy the above inequalities. For the purpose of analysis, by scaling, we assume that $f(O) = \sum_{o \in O} g(o) = 1$. The following lemma formalizes the above sketch. It is also the basis for the *factor-revealing* LP below (for its proof refer to [36]).

▶ **Lemma 10.** *Let $m \geq 1$ be an integral parameter. We can pick for each $i \in [m]$ a set $O_i = \{o_1^i, o_2^i, \ldots, o_{q_i}^i\} \subseteq O$ (possibly empty) such that the following holds. For $i \neq j$, we have that $O_i \cap O_j = \varnothing$. Let $L_i = \sum_{j=1}^i q_j$, $Q_i = \cup_{j=1}^i O_j$, $\mathbf{c}_i = \mathbf{C1}_{Q_i}$, $\mathbf{p}_i = \mathbf{P1}_{Q_i}$ and let $A_i := T[L_i, \mathbf{c}_i, \mathbf{p}_i]$ be the corresponding DP cell. Then $\mathbf{C1}_{A_m} \geq \mathbf{c}/2$ and the following inequalities hold.*

1. $f(A_0) = g(O_0)$ *where* $A_0 = O_0 = \varnothing$,
2. $f(A_i) \geq f(A_{i-1}) + (1 - i/m)g(O_i)$ $\forall i \in [m]$ *and*
3. $f(A_i) \geq \frac{i}{m}\left(1 - \sum_{j \leq i} g(O_j)\right)$ $\forall i \in \{0\} \cup [m]$.

Below we describe a *factor-revealing* LP that captures the above-described multi-phase analysis for the greedy DP algorithm. The idea is to introduce variables for the quantities in the inequalities in the previous lemma and determining the minimum ratio that can be

guaranteed by these inequalities.

$$\begin{aligned}
&\min\ a_m && \text{s.t.} && && \text{(LP)}\\
&\quad a_1 \geq \left(1-\tfrac{1}{m}\right) o_1; && && && (5)\\
&\quad a_i \geq a_{i-1} + \left(1-\tfrac{i}{m}\right) o_i && \forall i \in [m]\setminus\{1\}; && (6)\\
&\quad a_i \geq \tfrac{i}{m}\left(1-\textstyle\sum_{j\leq i} o_j\right) && \forall i \in [m]; && (7)\\
&\quad a_i \geq 0,\ o_i \geq 0 && \forall i \in [m]. && (8)
\end{aligned}$$

The variable o_i corresponds to the marginal value $g(O_i)$ for the set O_i in our analysis. Variables a_i correspond to the quantities $f(A_i)$ for the approximate solution A_i for each phase $i = 1, 2, \ldots, m$. We add all the inequalities we proved in Lemma 10 as the constraints for this LP. Note that since $f(O) = 1$, the minimum possible value of a_m will correspond to a lower bound on the approximation ratio of our algorithm.

The following is the dual for the above LP.

$$\begin{aligned}
&\max\ \textstyle\sum_{i=1}^{m} \frac{i}{m} y_i && \text{s.t.} && && \text{(DP)}\\
&\quad x_i + y_i - x_{i+1} \leq 0 && && \forall i \in [m-1]; && (9)\\
&\quad x_m + y_m \leq 1; && && && (10)\\
&\quad \textstyle\sum_{j\geq i} \frac{j}{m} y_j - (1-\frac{i}{m}) x_i \leq 0 && \forall i \in [m]; && (11)\\
&\quad x_i \geq 0,\ y_i \geq 0 && \forall i \in [m]. && (12)
\end{aligned}$$

This linear program gives for every m a lower bound on the approximation ratio. Analytically, we can show that if m tends to infinity the optimum value of the LP converges to $1/e$. This leads to the following lemma (for its proof refer to [36]).

▶ **Lemma 11.** *Assuming p and c are constants, the vanilla greedy dynamic programming algorithm for* (PCSM) *runs in pseudo-polynomial time $O(n^2 p_{\max} c_{\max})$ and outputs a solution $S \subseteq \mathcal{N}$ that satisfies:* (1) $f(S) \geq (1/e) \cdot f(O)$, (2) $\mathbf{P1}_S \leq \mathbf{p}$ *and* $\mathbf{C1}_S \geq 1/2 \cdot \mathbf{c}$.

Applications and Extensions of Greedy Dynamic Programming Approach

We briefly explain the applications of the approach to the various specific settings and the required tailored algorithmic extensions to the vanilla version of the algorithm.

Scaling, guessing and post-processing for packing constraints. An immediate consequence of Lemma 11 is a *deterministic* $(1/e)$-approximation for the case of constantly many packing constraints that runs in pseudo-polynomial time. We can apply standard scaling techniques to achieve truly polynomial time. This may, however, introduce a violation of the constraints within a factor of $(1 + \varepsilon)$. To avoid this violation, we can apply a pre-processing and post-processing by Kulik et al. [32] to achieve Theorem 4.

Forbidden sets for a single packing and a single covering constraints. In this setting we are able to ensure a $(1 - \varepsilon)$-violation of the covering constraints by using the concept of *forbidden sets*. Intuitively, we exclude the elements of these set from being included to the dynamic programming table in order to be able to complete the table entries to solutions with only small violation.

Fix some $\varepsilon > 0$. By guessing we assume that we know the set G of all, at most $1/\varepsilon$ elements ℓ from the optimum solution with $\mathbf{P}_\ell > \varepsilon \cdot \mathbf{p}$. We can guess G using brute force in $n^{O(1/\varepsilon)}$ time. This allows us to remove all elements with $\mathbf{P}_\ell \geq \varepsilon \cdot \mathbf{p}$ from the instance. Let $\mathcal{N}'$ be the rest of the elements. (For consistency reasons, we use bold-face vector notation here also for dimension one.)

Fix an order of $\mathcal{N}'$ in which the elements are sorted in a non-increasing order of $\mathbf{C}_\ell/\mathbf{P}_\ell$ values, breaking ties arbitrarily. Let $\mathcal{N}_i$ be the set of the first i elements in this order. For any $\mathbf{p}' \leq \mathbf{p}$, let $F_{\mathbf{p}'}$ be the smallest set $\mathcal{N}_i$ with $\mathbf{P}\mathbf{1}_{\mathcal{N}_i} \geq \mathbf{p} - \mathbf{p}'$. Note that the profit of $F_{\mathbf{p}'}$ is at least the profit of any subset of $\mathcal{N}'$ with packing value at most $\mathbf{p} - \mathbf{p}'$ and that the packing value of $F_{\mathbf{p}'}$ is no larger than $(1+\varepsilon)\mathbf{p} - \mathbf{p}'$. Also note that for any $0 \leq \mathbf{p}' \leq \mathbf{p}'' \leq \mathbf{p}$, it holds that $F_{\mathbf{p}''} \subseteq F_{\mathbf{p}'}$.

Now we explain the modified Greedy-DP that incorporates the guessing and the forbidden sets ideas. Let G be the set of the guessed big elements as described above. For the base case, we set $T[\mathbf{C}\mathbf{1}_G, \mathbf{P}\mathbf{1}_G] = G$ and $T[\mathbf{c}', \mathbf{p}'] = \perp$ for all table entries with $\mathbf{c}' \neq \mathbf{C}\mathbf{1}_G$ or $\mathbf{p}' \neq \mathbf{P}\mathbf{1}_G$.

In order to compute $T[\mathbf{c}', \mathbf{p}']$, we look at every set of the form $T[\mathbf{c}' - \mathbf{C}_\ell, \mathbf{p}' - \mathbf{P}_\ell] \cup \{\ell\}$, where $\ell \in \mathcal{N} \setminus (T[\mathbf{c}' - \mathbf{C}_\ell, \mathbf{p}' - \mathbf{P}_\ell] \cup F_{\mathbf{p}'})$, $\mathbf{c}' - \mathbf{C}_\ell \geq 0$, and $\mathbf{p}' - \mathbf{P}_\ell \geq 0$. Notice that we forbid elements belonging to $F_{\mathbf{p}'}$ to be included in any table entry of the form $T[\mathbf{c}', \mathbf{p}']$. Now out of all these sets, we assign the most valuable set to $T[\mathbf{c}', \mathbf{p}']$. The output of our algorithm is the best of the solutions $T[\mathbf{c}', \mathbf{p}'] \cup F_{\mathbf{p}'}$, such that $\mathbf{c}' + \mathbf{C}\mathbf{1}_{F_{\mathbf{p}'}} \geq \mathbf{c}$.

By means of a more sophisticated factor-revealing LP, we obtain Theorem 5. Finally, if the packing constraint is actually a cardinality constraint we can assume that $\varepsilon < 1/\mathbf{p}$. Hence, there will be no violation of the cardinality constraint and also guessing can be avoided.

5 Extensions: Matroid Independence and Multi-Objective

Refer to [36] for the extensions that deal with a matroid independence constraint and with multiple objectives.

References

1 A. A. Ageev and M. I. Sviridenko. An 0.828 Approximation Algorithm for the Uncapacitated Facility Location Problem. *Discrete Appl. Math.*, 93(2-3):149–156, July 1999.

2 Shabbir Ahmed and Alper Atamtürk. Maximizing a class of submodular utility functions. *Mathematical Programming*, 128(1):149–169, June 2011.

3 Per Austrin, Siavosh Benabbas, and Konstantinos Georgiou. Better Balance by Being Biased: A 0.8776-Approximation for Max Bisection. *ACM Trans. Algorithms*, 13(1):2:1–2:27, 2016.

4 Francis Bach. *Learning with Submodular Functions: A Convex Optimization Perspective*. Now Publishers Inc., Hanover, MA, USA, 2013.

5 L. Bordeaux, Y. Hamadi, and P. Kohli. *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 2014.

6 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular Maximization with Cardinality Constraints. In *Proc. 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, pages 1433–1452, 2014.

7 Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a Monotone Submodular Function Subject to a Matroid Constraint. *SIAM J. Comput.*, 40(6):1740–1766, December 2011.

8 Chandra Chekuri and Sanjeev Khanna. A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem. *SIAM Journal on Computing*, 35(3):713–728, 2005.

9 Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent Randomized Rounding via Exchange Properties of Combinatorial Structures. In *Proc. 51th Annual IEEE Symposium on Foundations of Computer Science (FOCS'10)*, pages 575–584, 2010.

10 Reuven Cohen, Liran Katzir, and Danny Raz. An Efficient Approximation for the Generalized Assignment Problem. *Inf. Process. Lett.*, 100(4):162–166, November 2006.

11 Gerard Cornuejols, Marshall Fisher, and George L. Nemhauser. On the Uncapacitated Location Problem. In *Studies in Integer Programming*, volume 1 of *Annals of Discrete Mathematics*, pages 163–177. Elsevier, 1977.

12 Gerard Cornuejols, Marshall L. Fisher, and George L. Nemhauser. Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms. *Management Science*, 23(8):789–810, 1977.

13 Shaddin Dughmi, Tim Roughgarden, and Mukund Sundararajan. Revenue Submodularity. *Theory of Computing*, 8(1):95–119, 2012.

14 U. Feige and J. Vondrak. Approximation algorithms for allocation problems: Improving the factor of 1 - 1/e. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 667–676, 2006.

15 Uriel Feige. A Threshold of Ln N for Approximating Set Cover. *J. ACM*, 45(4):634–652, July 1998.

16 Lisa Fleischer, Michel X Goemans, Vahab S Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *Proc. 17th annual ACM-SIAM Symposium on Discrete Algorithm, (SODA'06)*, pages 611–620. SIAM, 2006.

17 Alan Frieze and Mark Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. In Egon Balas and Jens Clausen, editors, *Integer Programming and Combinatorial Optimization*, pages 1–13. Springer Berlin Heidelberg, 1995.

18 J. Gillenwater. *Approximate Inference for Determinantal Point Processes*. PhD thesis, University of Pennsylvania, 2014.

19 Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Near-Optimal MAP Inference for Determinantal Point Processes. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2735–2743. Curran Associates, Inc., 2012.

20 Michel X. Goemans and David P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM*, 42(6):1115–1145, November 1995.

21 Eran Halperin and Uri Zwick. Combinatorial Approximation Algorithms for the Maximum Directed Cut Problem. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 1–7, 2001.

22 Jason Hartline, Vahab Mirrokni, and Mukund Sundararajan. Optimal Marketing Strategies over Social Networks. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 189–198, 2008.

23 Johan Håstad. Clique is hard to approximate within $n^{(1-\varepsilon)}$. In *Acta Mathematica*, pages 627–636, 1996.

24 Johan Håstad. Some Optimal Inapproximability Results. *J. ACM*, 48(4):798–859, July 2001.

25 Xinran He and David Kempe. Stability of Influence Maximization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1256–1265, 2014.

26 Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, 1972.

27 David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the Spread of Influence through a Social Network. *Theory of Computing*, 11(4):105–147, 2015.

28 Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007.

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

30 Alex Kulesza and Ben Taskar. Learning Determinantal Point Processes. In *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011*, pages 419–427, 2011.

31 Alex Kulesza and Ben Taskar. Determinantal Point Processes for Machine Learning. *Foundations and Trends in Machine Learning*, 5(2-3):123–286, 2012.

32 Ariel Kulik, Hadas Shachnai, and Tami Tamir. Approximations for Monotone and Non-monotone Submodular Maximization with Knapsack Constraints. *Mathematics of Operations Research*, 38(4):729–739, 2013. preliminary version appeared in SODA'09.

33 Jon Lee, Vahab S Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proc. 41st Annual ACM Symposium on Theory Of Computing, (STOC'09)*, pages 323–332. ACM, 2009.

34 Hui Lin and Jeff Bilmes. Multi-document Summarization via Budgeted Maximization of Submodular Functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 912–920, 2010.

35 Hui Lin and Jeff Bilmes. A Class of Submodular Functions for Document Summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 510–520, 2011.

36 Eyal Mizrachi, Roy Schwartz, Joachim Spoerhase, and Sumedha Uniyal. A Tight Approximation for Submodular Maximization with Mixed Packing and Covering Constraints. *CoRR*, abs/1804.10947, 2018. `arXiv:1804.10947`.

37 George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.*, 14(1):265–294, 1978.

38 George L Nemhauser and Leonard A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978.

39 Andreas S. Schulz and Nelson A. Uhan. Approximating the least core value and least core of cooperative games with supermodular costs. *Discrete Optimization*, 10(2):163–180, 2013.

40 Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.*, 32(1):41–43, 2004.

41 Jan Vondrák. Symmetry and Approximability of Submodular Maximization Problems. In *Proc. 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS'09)*, pages 651–670. IEEE, 2009.

42 Laurence A. Wolsey. Maximising Real-Valued Submodular Functions: Primal and Dual Heuristics for Location Problems. *Mathematics of Operations Research*, 7(3):410–425, 1982.