SIAM J. Comput.
Vol. 40, No. 6, pp. 1740–1766

MAXIMIZING A MONOTONE SUBMODULAR FUNCTION SUBJECT TO A MATROID CONSTRAINT*

GRUIA CALINESCU†, CHANDRA CHEKURI‡, MARTIN PÁL§, AND JAN VONDRÁK¶

Abstract. Let $f : 2^X \to \mathcal{R}_+$ be a monotone submodular set function, and let $(X, \mathcal{I})$ be a matroid. We consider the problem $\max_{S \in \mathcal{I}} f(S)$. It is known that the greedy algorithm yields a 1/2-approximation [M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, *Math. Programming Stud.,* no. 8 (1978), pp. 73–87] for this problem. For certain special cases, e.g., $\max_{|S| \le k} f(S)$, the greedy algorithm yields a $(1 - 1/e)$-approximation. It is known that this is optimal both in the value oracle model (where the only access to f is through a black box returning $f(S)$ for a given set S) [G. L. Nemhauser and L. A. Wolsey, *Math. Oper. Res.*, 3 (1978), pp. 177–188] and for explicitly posed instances assuming $P \neq NP$ [U. Feige, *J. ACM*, 45 (1998), pp. 634–652]. In this paper, we provide a randomized $(1 - 1/e)$-approximation for any monotone submodular function and an arbitrary matroid. The algorithm works in the value oracle model. Our main tools are a variant of the *pipage rounding technique* of Ageev and Sviridenko [*J. Combin. Optim.*, 8 (2004), pp. 307–328], and a *continuous greedy process* that may be of independent interest. As a special case, our algorithm implies an optimal approximation for the submodular welfare problem in the value oracle model [J. Vondrák, *Proceedings of the* 38*th ACM Symposium on Theory of Computing*, 2008, pp. 67–74]. As a second application, we show that the generalized assignment problem (GAP) is also a special case; although the reduction requires $|X|$ to be exponential in the original problem size, we are able to achieve a $(1 - 1/e - o(1))$-approximation for GAP, simplifying previously known algorithms. Additionally, the reduction enables us to obtain approximation algorithms for variants of GAP with more general constraints.

Key words. monotone submodular set function, matroid, social welfare, generalized assignment problem, approximation algorithm

AMS subject classifications. 68W20, 68W25, 52B40

DOI. 10.1137/080733991

1. Introduction.

This paper is motivated by the following optimization problem: We are given a ground set X of n elements and a monotone submodular set function $f : 2^X \to \mathcal{R}_+$. A function f is *submodular* iff

$$f(A \cup B) + f(A \cap B) \le f(A) + f(B)$$

for all $A, B \subseteq X$. We restrict our attention to monotone (by which we mean nondecreasing) functions, that is, $f(A) \le f(B)$ for all $A \subseteq B$, and we assume $f(\emptyset) = 0$. We are also given an *independence family* $\mathcal{I} \subseteq 2^X$, a family of subsets that is downward closed; that is, $A \in \mathcal{I}$ and $B \subseteq A$ imply that $B \in \mathcal{I}$. A set A is *independent* iff $A \in \mathcal{I}$. We are primarily interested in the special case where $\mathcal{I}$ is the collection of independent sets of a given matroid $\mathcal{M} = (X, \mathcal{I})$ (we give a definition of a matroid in section 2.1).

*Received by the editors September 2, 2008; accepted for publication (in revised form) October 15, 2009; published electronically December 22, 2011. This work combines results and ideas from two extended, previously published abstracts [3, 42].
http://www.siam.org/journals/sicomp/40-6/73399.html

†Computer Science Department, Illinois Institute of Technology, Chicago, IL 60616 (calinescu@iit.edu). This author's research was partially supported by NSF grant CCF-0515088.
‡Department of Computer Science, University of Illinois, Urbana, IL 61801 (chekuri@cs.illinois.edu). This author's work was partly done while the author was at Lucent Bell Labs and was partially supported by NSF grant CCF-0728782.
§Google Inc., 1440 Broadway, New York, NY 10018 (mpal@google.com).
¶IBM Almaden Research Center, San Jose, CA 95120 (jvondrak@us.ibm.com). This author's work was done while the author was at Princeton University.

For computational purposes we will assume that f and $\mathcal{I}$ are specified as oracles, although in specific settings of interest, an explicit description is often available.

Submodular maximization subject to a matroid. The problem (or, rather, class of problems) of interest in this paper is the problem of maximizing $f(S)$ over the independent sets $S \in \mathcal{I}$; in other words we wish to find $\max_{S\in\mathcal{I}} f(S)$. We denote by SUB-M the problem where f is monotone submodular and $\mathcal{M} = (X, \mathcal{I})$ is a matroid.

The problem of maximizing a submodular set function subject to independence constraints has been studied extensively—we discuss prior work at the end of the section. A number of interesting and useful combinatorial optimization problems, including NP-hard problems, are special cases. Some notable examples are maximum independent set in a matroid, maximum matroid intersection, and max-k-cover. Below we describe some candidates for f and $\mathcal{I}$ that arise frequently in applications.

Modular functions. A function $f : 2^X \to \mathcal{R}_+$ is modular iff $f(A) + f(B) = f(A \cup B) + f(A \cap B)$ for all A, B. If f is modular, then there is a weight function $w : X \to \mathcal{R}_+$ such that $f(A) = w(A) = \sum_{e\in A} w(e)$. Such functions are also referred to as *additive* or *linear*.

Set systems and coverage. Given a universe U and n subsets $A_1, A_2, \dots, A_n \subset U$, we obtain several natural submodular functions on the set $X = \{1, 2, \dots, n\}$. First, the coverage function f given by $f(S) = |\cup_{i\in S} A_i|$ is submodular. This naturally extends to the weighted coverage function; given a nonnegative weight function $w : U \to \mathcal{R}_+$, $f(S) = w(\cup_{i\in S} A_i)$. We obtain a multicover version as follows. For $x \in U$ let $k(x)$ be an integer. For each $x \in U$ and A_i let $c(A_i, x) = 1$ if $x \in A_i$ and 0 if $x \notin A_i$. Given $S \subseteq X$, let $c'(S, x)$, the coverage of x under S, be defined as $c'(S, x) = \min\{k(x), \sum_{i\in S} c(A_i, x)\}$. The function $f(S) = \sum_{x\in U} c'(S, x)$ is submodular. A related function defined by $f(S) = \sum_{x\in U} \max_{i\in S} w(A_i, x)$ is also submodular, where $w(A_i, x)$ is a nonnegative weight for A_i covering x.

Weighted rank functions of matroids and their sums. The rank function of a matroid $\mathcal{M} = (X, \mathcal{I})$, $r_{\mathcal{M}}(A) = \max\{|S| : S \subseteq A, S \in \mathcal{I}\}$, is submodular. Given $w : X \to \mathcal{R}_+$, the weighted rank function defined by $r_{\mathcal{M},w}(A) = \max\{w(S) : S \subseteq A, S \in \mathcal{I}\}$ is a submodular function [39]. Submodularity is preserved by taking a sum, and hence a sum of weighted rank functions is also submodular. The functions of coverage type mentioned above are captured by this class. However, the class does not include all monotone submodular functions.

Matroid constraint. An independence family of particular interest is one induced by a matroid $\mathcal{M} = (X, \mathcal{I})$. A simple matroid constraint that is of much importance in applications [9, 37, 7, 8, 19] is the *partition matroid*: X is partitioned into ℓ sets $X_1, X_2, \dots, X_\ell$ with associated integers $k_1, k_2, \dots, k_\ell$, and a set $A \subseteq X$ is independent iff $|A \cap X_i| \le k_i$. In fact even the case of $\ell = 1$ (the uniform matroid) is of interest.

Intersection of matroids. A natural generalization of the single matroid case is obtained when we consider intersections of different matroids $\mathcal{M}_1 = (X, \mathcal{I}_1)$, $\mathcal{M}_2 = (X, \mathcal{I}_2), \dots, \mathcal{M}_p = (X, \mathcal{I}_p)$ on the same ground set X. That is, $\mathcal{I} = \{A \subseteq X \mid A \in \mathcal{I}_i, 1 \le i \le p\}$. A simple example is the family of hypergraph matchings in a p-partite hypergraph ($p = 2$ is simply the family of matchings in a bipartite graph).

p-systems. More general independence families parametrized by an integer p can be defined. We follow the definition of [25, 28]. Given an independence family $\mathcal{I}$ and a set $Y \subseteq X$, let $\mathcal{B}(Y)$ be the set of maximal independent sets of $\mathcal{I}$ included in Y, that is, $\mathcal{B}(Y) = \{A \in \mathcal{I} \mid A \subseteq Y$ and there is no $A' \in \mathcal{I}$ such that $A \subset A' \subseteq Y\}$. Then $\mathcal{I}$ is a p-system if, for all $Y \subseteq X$, $\frac{\max_{A\in\mathcal{B}(Y)} |A|}{\min_{A\in\mathcal{B}(Y)} |A|} \le p$. p-systems properly generalize several simpler and easier to understand special cases including families obtained from the

intersection of p matroids. We discuss some of these special cases in Appendix B). The set of matchings in a general graph forms a 2-system. Similarly the set of matchings in a hypergraph with edges of cardinality at most p forms a p-system.

The greedy algorithm. A greedy algorithm is quite natural for the problem $\max\{f(S) : S \in \mathcal{I}\}$. The algorithm incrementally builds a solution (without backtracking) starting with the empty set. In each iteration it adds an element that most improves the current solution (according to f) while maintaining independence of the solution. The greedy algorithm yields a $1/p$-approximation for maximizing a modular function subject to a p-system independence constraint [25, 28]. For submodular functions, the greedy algorithm yields a ratio of $1/(p+1)$ [17].[1] These ratios for the greedy algorithm are tight for all p, even for intersections of p matroids. For large but fixed p, the p-dimensional matching problem is NP-hard to approximate to within an $\Omega(\log p/p)$ factor [24].

For the problem of maximizing a submodular function subject to a matroid constraint (special case of $p = 1$), the greedy algorithm achieves a ratio of $1/2$. When the matroid is uniform, i.e., the problem is $\max\{f(S) : |S| \leq k\}$, the greedy algorithm yields a $(1-1/e)$-approximation and this is optimal in the value oracle model [37, 36]. This special case already captures the max-k-cover problem (with $f(S)$ the coverage function of a set system) for which it is shown in [13] that no $(1 - 1/e + \epsilon)$-approximation is possible for any constant $\epsilon > 0$, unless $P = NP$. Thus it is natural to ask whether a $1-1/e$ approximation is achievable for any matroid or whether there is a gap between the case of uniform matroids and general matroids. We resolve the question in this paper.

THEOREM 1.1. *There is a randomized algorithm giving a* $(1-1/e)$*-approximation (in expectation) to the problem* $\max\{f(S) : S \in \mathcal{I}\}$*, where* $f : 2^X \to \mathcal{R}_+$ *is a monotone submodular function given by a value oracle, and* $\mathcal{M} = (X, \mathcal{I})$ *is a matroid given by a membership oracle.*

The proof of this theorem appears in section 3. Our main tools are the *pipage rounding* technique of Ageev and Sviridenko [1] and a *continuous greedy process* [42]. We give an overview of these techniques in section 2.

Applications. Using Theorem 1.1 we obtain $(1 - 1/e)$-approximation algorithms for a number of optimization problems. An immediate application is the submodular welfare problem [32], which can be cast as a submodular maximization problem subject to a partition matroid. (This reduction already appeared in [17].) In this problem, we are given n players and m items. Each player i has a monotone submodular utility function $w_i : 2^{[m]} \to \mathcal{R}_+$. The goal is to allocate items to the agents to maximize the total utility $\sum_{i=1}^{n} w_i(S_i)$. It was known that the greedy algorithm yields a $1/2$-approximation [17, 32], while a $(1 - 1/e)$-approximation in the value oracle model would be optimal [26, 35]. Improvements over the $1/2$-approximation were achieved only in special cases or using a stronger computation model [11, 16]. Our work implies the following optimal result, which first appeared in [42].

THEOREM 1.2. *There is a randomized algorithm which achieves a* $(1 - 1/e)$*-approximation (in expectation) for the submodular welfare problem in the value oracle model.*

The proof of this theorem appears in section 4.1. Another application of Theorem 1.1 is related to variants of the generalized assignment problem (GAP). In GAP

[1] We give a proof of this result in Appendix B for the sake of completeness. If only an α-approximate oracle ($\alpha \leq 1$) is available for the function evaluation, the ratio obtained is $\alpha/(p + \alpha)$. Several old and recent applications of the greedy algorithm can be explained using this observation.

we are given n bins and m items. Each item i specifies a size s_{ij} and a value (or profit) v_{ij} for each bin j. Each bin has capacity 1, and the goal is to assign a subset of items to bins such that the bin capacities are not violated and the profit of the assignment is maximized. Fleischer et al. [19] gave a $(1 - 1/e)$-approximation for this problem, improving upon a previous 1/2-approximation [6]. We rederive almost the same ratio, casting the problem as a special case of submodular function maximization. Moreover, our techniques allow us to obtain a $(1 - 1/e - o(1))$-approximation for GAP, even under any given matroid constraint on the bins. A simple example is GAP with the added constraint that at most k of the n bins be used for packing items.

THEOREM 1.3. *Let A be an instance of GAP with n bins and m items, and let B be the set of bins. Let $\mathcal{M} = (B, \mathcal{I})$ be a matroid on B. There is a randomized $(1 - 1/e - o(1))$-approximation to find a maximum profit assignment to bins such that the subset $S \subseteq B$ of bins that are used in the assignment satisfies the constraint $S \in \mathcal{I}$.*

The proof of this theorem appears in section 4.2. We note that the approximation ratio for GAP has been improved to $1-1/e+\delta$ for a small $\delta > 0$ in [16] using the same linear programming relaxation as in [19]. However, the algorithm in [19] extends to an even more general class of problems, the *separable assignment problem* (SAP). For SAP, it is shown in [19] that an approximation ratio of $1 - 1/e + \epsilon$ for any constant $\epsilon > 0$ implies $NP \subseteq DTIME(n^{O(\log \log n)})$. Our framework also extends to SAP, and hence $1 - 1/e$ is the best approximation factor one can achieve with this approach. We discuss this further in section 4.2.

Remarks on efficiency. Our algorithm is conceptually quite simple and easy to implement; however, its running time is a different issue. The variant described here would run in time on the order of $\tilde{O}(n^8)$, using $\tilde{O}(n^7)$ oracle queries, mostly due to the number of random samples necessary to achieve high probability bounds (see section 3.1). This can be substantially reduced by a more careful implementation and analysis, which will be discussed in a future paper.

Prior and related work. Maximizing a submodular function subject to independence constraints is related to many important and interesting problems in combinatorial optimization. The seminal work of Edmonds on polyhedral aspects of matroids and matroid intersection led to a variety of investigations on linear function optimization subject to independence constraints. In particular, the greedy algorithm, which is known to be optimal for a single matroid [12], was shown to be $1/p$-approximate for p-systems by Jenkyns [25], which generalized an earlier result [28]. The impetus for considering maximizing submodular functions subject to a matroid constraint appears to have come from an application to a facility location problem in the work of Cornuejols, Fisher, and Nemhauser [9]. In two important papers [37, 17], Fisher, Nemhauser, and Wolsey formally stated and studied the problem SUB-M and generalizations to other independent systems; they proved a number of useful theorems, including the analysis of greedy as well as local search, and also discussed various applications. Unfortunately, the results in these early papers appear to have been forgotten in later work on approximation algorithms, and easy consequences were often rediscovered. We note that the "locally greedy heuristic" for partition matroids in [17] is particularly useful. See [21] for a survey of recent applications. Nemhauser and Wolsey [36] showed that for SUB-M with the uniform matroid, in the model where f is accessed via a value oracle, any algorithm that obtains an approximation better than $(1 - 1/e)$ requires an exponential number of queries. As we already mentioned, Feige [13] showed that, unless $P = NP$, $(1 - 1/e)$ is the best approximation ratio for max-k-cover.

The (historically) first part of our work (the pipage rounding technique) is inspired by that of Ageev and Sviridenko [1], who introduced the pipage rounding technique and gave a $(1 - 1/e)$-approximation for the special case of the max-coverage problem with a partition matroid constraint, among several other results. See [40, 20] for randomized variants of pipage rounding. The starting point for this work was the observation that the pipage rounding technique can be generalized to arbitrary matroids [3]. This work also introduced the notion of a *multilinear extension* of a submodular function, which is crucial in this paper. Another inspiration for this work came from the $(1 - 1/e)$-approximation for GAP given in [19].

The second main technical ingredient of our work (the *continuous greedy process*) first appeared in [42]. Related to it is the notion of a real-valued submodular function and a continuous greedy algorithm considered by Wolsey [44]; the differences are discussed at the end of section 2.3. An important motivating problem for this part of our work is the submodular welfare problem; see section 4.1 for related work on this topic. Maximizing a monotone submodular function subject to knapsack constraints has also received attention; Sviridenko [41] gave a $(1-1/e)$-approximation for a single knapsack constraint based on earlier work [27] on max-coverage. Recently, this has been extended to a constant number of knapsack constraints [29], using the continuous greedy algorithm described in this paper.

We have so far discussed the problem SUB-M and its generalizations in the context of monotone submodular functions. It is also of interest to consider nonmonotone submodular functions that are nonnegative; cut-functions in undirected and directed graphs are prominent special cases. Recent work has obtained several new approximation results in this more general setting with local search and randomization as the key techniques [15, 30, 31]. Further connections between the notion of a multilinear extension and approximability of submodular maximization problems are studied in [43].

2. Overview of techniques.

We start with an overview of our approach to the problem of maximizing a monotone submodular function subject to a matroid constraint. We present background material and intuition for our algorithm, which appears in the next section.

2.1. Preliminaries.

Submodular functions. A function $f : 2^X \to \mathcal{R}$ is submodular if, for all $A, B \subseteq X$,

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B).$$

Given a submodular function $f : 2^X \to \mathcal{R}$ and $A \subset X$, the function f_A defined by $f_A(S) = f(S \cup A) - f(A)$ is also submodular. Further, if f is monotone, f_A is also monotone. For $i \in X$, we abbreviate $S \cup \{i\}$ by $S + i$. By $f_A(i)$, we denote the "marginal value" $f(A + i) - f(A)$. For monotone functions, submodularity is equivalent to $f_A(i)$ being nonincreasing as a function of A for every fixed i.

Smooth submodular functions. As a continuous analogy, Wolsey [45] defines submodularity for a function $F : [0, 1]^X \to \mathcal{R}_+$ as follows:

$$F(x \vee y) + F(x \wedge y) \leq F(x) + F(y), \tag{2.1}$$

where $(x \vee y)_i = \max\{x_i, y_i\}$ and $(x \wedge y)_i = \min\{x_i, y_i\}$. Similarly, a function is monotone if $F(x) \leq F(y)$ whenever $x \leq y$ coordinatewise. In particular, Wolsey works with monotone submodular functions that are piecewise linear and, in addition, concave. In this paper, we use a related property which we call *smooth monotone submodularity.*

DEFINITION 2.1. *A function $F : [0,1]^X \to \mathcal{R}$ is smooth monotone submodular if the following hold:*

- *$F \in C_2([0,1]^X)$; i.e., it has second partial derivatives everywhere.*
- *For each $j \in X$, $\frac{\partial F}{\partial y_j} \geq 0$ everywhere (monotonicity).*
- *For any $i, j \in X$ (possibly equal), $\frac{\partial^2 F}{\partial y_i \partial y_j} \leq 0$ everywhere (submodularity).*

Thus the *gradient* $\nabla F = (\frac{\partial F}{\partial y_1}, \dots, \frac{\partial F}{\partial y_n})$ is a nonnegative vector. The submodularity condition $\frac{\partial^2 F}{\partial y_i \partial y_j} \leq 0$ means that $\frac{\partial F}{\partial y_j}$ is nonincreasing with respect to y_i. It can be seen that this implies (2.1). Also, it means that a smooth submodular function is concave along any nonnegative direction vector; however, it is not necessarily concave in all directions.

Multilinear extension. For a monotone submodular set function $f : 2^X \to \mathcal{R}_+$, a canonical extension to a smooth monotone submodular function can be obtained as follows [3]: For $y \in [0,1]^X$, let $\hat{y}$ denote a random vector in $\{0,1\}^X$ where each coordinate is independently rounded to 1 with probability y_j or 0 otherwise. We identify $\hat{y} \in \{0,1\}^X$ with a set $R \subseteq X$ whose indicator vector is $\hat{y} = \mathbf{1}_R$. Then, we define

$$F(y) = \mathbf{E}[f(\hat{y})] = \sum_{R \subseteq X} f(R) \prod_{i \in R} y_i \prod_{j \notin R} (1 - y_j).$$

This is a multilinear polynomial which satisfies

$$\frac{\partial F}{\partial y_j} = \mathbf{E}[f(\hat{y}) \mid \hat{y}_j = 1] - \mathbf{E}[f(\hat{y}) \mid \hat{y}_j = 0] \geq 0$$

by monotonicity of f. For $i \neq j$, we get

$$\begin{aligned}\frac{\partial^2 F}{\partial y_i \partial y_j} &= \mathbf{E}[f(\hat{y}) \mid \hat{y}_i = 1, \hat{y}_j = 1] - \mathbf{E}[f(\hat{y}) \mid \hat{y}_i = 1, \hat{y}_j = 0] \\ &\quad - \mathbf{E}[f(\hat{y}) \mid \hat{y}_i = 0, \hat{y}_j = 1] + \mathbf{E}[f(\hat{y}) \mid \hat{y}_i = 0, \hat{y}_j = 0] \\ &\leq 0\end{aligned}$$

by submodularity of f. In addition, $\frac{\partial^2 F}{\partial {y_j}^2} = 0$, since F is multilinear.

Example. Consider $X = [n]$ and $f(S) = \min\{|S|, 1\}$. This is a monotone submodular function. The multilinear extension of this function is

$$F(y) = \mathbf{E}[f(\hat{y})] = 1 - \prod_{i=1}^{n} (1 - y_i).$$

This is the unique multilinear function which coincides with $f(S)$ on $\{0,1\}$-vectors. It is also easy to see that its first partial derivatives are nonnegative and its second partial derivatives are nonpositive; i.e., $F(y)$ is a smooth monotone submodular function.

Chernoff bounds. In the value oracle model, we cannot compute the multilinear extension exactly, but we can estimate it (and related quantities) by random sampling. To this end we will use the symmetric version of Theorem A.1.16 in [2].

THEOREM 2.2. *Let X_i, $1 \leq i \leq k$, be mutually independent with all $\mathbf{E}[X_i] = 0$ and all $|X_i| \leq 1$. Set $S = X_1 + \cdots + X_k$. Then $Pr[|S| > a] \leq 2e^{-a^2/2k}$.*

Matroids. A matroid is a pair $\mathcal{M} = (X, \mathcal{I})$, where $\mathcal{I} \subseteq 2^X$ and
1. for all $B \in \mathcal{I}, A \subset B \Rightarrow A \in \mathcal{I}$,
2. for all $A, B \in \mathcal{I}; |A| < |B|$ implies that there exists $x \in B \setminus A; A + x \in \mathcal{I}$.

A matroid is a combinatorial abstraction of the notion of linear independence among vectors. By $r_{\mathcal{M}}$, we denote the *rank function* of $\mathcal{M}$:

$$r_{\mathcal{M}}(S) = \max\{|I| : I \subseteq S, I \in \mathcal{I}\}.$$

The rank function of a matroid is monotone and submodular. It is analogous to the notion of dimension in vector spaces.

Matroid polytopes. We consider polytopes $P \subset \mathcal{R}_+^X$ with the property that for any $x, y \in \mathcal{R}^X$, $0 \leq x \leq y$, $y \in P \Rightarrow x \in P$. We call such a polytope *down-monotone*. A down-monotone polytope of particular importance here is the *matroid polytope*. For a matroid $\mathcal{M} = (X, \mathcal{I})$, the matroid polytope is defined as

$$P(\mathcal{M}) = \text{conv } \{\mathbf{1}_I : I \in \mathcal{I}\}.$$

As shown by Edmonds [12], an equivalent description is

$$P(\mathcal{M}) = \left\{ x \geq 0 : \forall S \subseteq X; \sum_{j \in S} x_j \leq r_{\mathcal{M}}(S) \right\}.$$

From this description, it is clear that $P(\mathcal{M})$ is down-monotone.

A base of $\mathcal{M}$ is a set $S \in \mathcal{I}$ such that $r_{\mathcal{M}}(S) = r_{\mathcal{M}}(X)$. The base polytope of $\mathcal{M}$ is given by $B(\mathcal{M}) = \{y \in P(\mathcal{M}) \mid \sum_{i \in X} y_i = r_{\mathcal{M}}(X)\}$. The extreme points of $B(\mathcal{M})$ are the characteristic vectors of the bases of $\mathcal{M}$. Given the problem $\max\{f(S) : S \in \mathcal{I}\}$, where $\mathcal{M} = (X, \mathcal{I})$ is a matroid, there always exists an optimum solution S^* such that S^* is a base of $\mathcal{M}$. Note that this is false if f is not monotone. Thus, for monotone f, it is equivalent to consider the problem $\max_{S \in \mathcal{B}} f(S)$, where $\mathcal{B}$ is the set of bases of $\mathcal{M}$. See [39] for more details on matroids and polyhedral aspects.

2.2. Our approach. We consider the problem

$$\max\{f(S) : S \in \mathcal{I}\}, \tag{2.2}$$

where $f : 2^X \to \mathcal{R}_+$ is a monotone submodular function and $\mathcal{M} = (X, \mathcal{I})$ is a matroid. Apart from the greedy algorithm, which yields a 1/2-approximation for this problem, previous approaches have relied on *linear programming*. In special cases, such as the case of sums of weighted rank functions [3], the discrete problem (2.2) can be replaced by a linear programming problem,

$$\max \left\{ \sum_i c_i x_i : x \in P \right\}, \tag{2.3}$$

where P is a convex polytope. This linear program relies on the structure of a specific variant of the problem and typically can be solved exactly or to an arbitrary precision. Then, an optimal solution of (2.3) can be rounded to an integral solution $S \in \mathcal{I}$ while losing a certain fraction of the objective value. In the case where $f(S)$ is a sum of weighted rank functions of matroids, we showed in [3] that using the technique of *pipage rounding* we can recover an integral solution of value at least $(1 - 1/e)OPT$.

For a monotone submodular function $f(S)$ given by a value oracle, it is not obvious how to replace (2.2) by a linear program. Nonetheless, we have a generic way of replacing a discrete function $f(S)$ by a continuous function: the multilinear extension $F(y) = \mathbf{E}[f(\hat{y})]$ described in section 2.1. We note that although we cannot evaluate $F(y)$ exactly, we can approximate it by random sampling. For the purposes of this section, we can assume that we can evaluate $F(y)$ to an arbitrary precision. Using this extension, we obtain a *multilinear optimization problem*:

$$\max\{F(y) : y \in P(\mathcal{M})\}, \tag{2.4}$$

where $P(\mathcal{M})$ is the matroid polytope of $\mathcal{M}$. If $F(y)$ were a concave function, we would be in good shape to deal with this continuous problem, using convex optimization techniques (although it is still not clear how we would then convert a fractional solution to a discrete one). However, $F(y)$ is not a concave function (consider the example $F(y) = 1 - \prod_{i=1}^{n}(1 - y_i)$ from section 2.1). As we discussed in section 2.1, $F(y)$ is a smooth submodular function. Typically, it is concave in certain directions while convex in others. This will actually be useful both for treating the continuous problem and rounding its fractional solution.

Our solution.

1. We use a *continuous greedy process* to approximate $\max\{F(y) : y \in P(\mathcal{M})\}$ within a factor of $1 - 1/e$.
2. We use the *pipage rounding technique* to convert a fractional solution $y \in P(\mathcal{M})$ to a discrete solution S of value $f(S) \geq F(y) \geq (1 - 1/e)OPT$.

We remark that the second stage of the solution is identical to the one that we used in [3] in the case of sums of weighted rank functions. The first stage has been discovered more recently; it first appeared in [42] in the context of the submodular welfare problem. Next, we describe these two stages of our solution at a conceptual level before giving detailed proofs in section 3.

2.3. The continuous greedy process. We consider any down-monotone polytope P and a smooth monotone submodular function F. For concreteness, the reader may think of the matroid polytope $P(\mathcal{M})$ and the function $F(y) = \mathbf{E}[f(\hat{y})]$ defined in section 2.1. Our aim is to define a process that runs continuously, depending only on local properties of F, and produces a point $y \in P$ approximating the optimum $OPT = \max\{F(y) : y \in P\}$. We propose moving in the direction of a vector constrained by P which maximizes the local gain.

The continuous greedy process. We view the process as a particle starting at $y(0) = 0$ and following a certain flow over a unit time interval:

$$\frac{dy}{dt} = v_{max}(y),$$

where $v_{max}(y)$ is defined as $v_{max}(y) = \operatorname{argmax}_{v \in P}(v \cdot \nabla F(y))$.

Claim. $y(1) \in P$ and $F(y(1)) \geq (1 - 1/e)OPT$.

First, the trajectory for $t \in [0, 1]$ is contained in P, since

$$y(t) = \int_0^t v_{max}(y(\tau))d\tau$$

is a convex linear combination of vectors in P. To prove the approximation guarantee, fix a point y and suppose that $x^* \in P$ is the true optimum, $OPT = F(x^*)$. The essence of our analysis is that *the rate of increase in $F(y)$ is at least as much as the deficit*

$OPT - F(y)$. This kind of behavior always leads to a factor of $1 - 1/e$, as we show below.

Consider the direction $v^* = (x^* \vee y) - y = (x^* - y) \vee 0$. This is a nonnegative vector; since $v^* \leq x^* \in P$ and P is down-monotone, we also have $v^* \in P$. By monotonicity, $F(y + v^*) = F(x^* \vee y) \geq F(x^*) = OPT$. Consider the ray of direction v^* starting at y, and the function $F(y + \xi v^*)$, $\xi \geq 0$. The directional derivative of F along this ray is $\frac{dF}{d\xi} = v^* \cdot \nabla F$. Since F is smooth submodular and v^* is nonnegative, $F(y + \xi v^*)$ is concave in ξ and $\frac{dF}{d\xi}$ is nonincreasing. By concavity, $F(y + v^*) - F(y) \leq \frac{dF}{d\xi}\big|_{\xi=0} = v^* \cdot \nabla F(y)$. Since $v^* \in P$, and $v_{max}(y) \in P$ maximizes $v \cdot \nabla F(y)$ over all vectors $v \in P$, we get

$$v_{max}(y) \cdot \nabla F(y) \geq v^* \cdot \nabla F(y) \geq F(y + v^*) - F(y) \geq OPT - F(y). \tag{2.5}$$

Now let us return to our continuous process and analyze $F(y(t))$. By the chain rule and using (2.5), we get

$$\frac{dF}{dt} = \sum_j \frac{\partial F}{\partial y_j} \frac{dy_j}{dt} = v_{max}(y(t)) \cdot \nabla F(y(t)) \geq OPT - F(y(t)).$$

This means that $F(y(t))$ dominates the solution of the differential equation $\frac{d\phi}{dt} = OPT - \phi(t)$, $\phi(0) = 0$, which is $\phi(t) = (1 - e^{-t})OPT$. This proves $F(y(t)) \geq (1 - e^{-t})OPT$.

The direction $v_{max}(y)$ at each point is determined by maximizing a *linear function* $v \cdot \nabla F(y)$ over $v \in P$. In the case of a matroid polytope $P(\mathcal{M})$, this problem can be solved very efficiently. We can assume that $v_{max}(y)$ is a vertex of P and furthermore, since ∇F is a nonnegative vector, that this vertex corresponds to a base of $\mathcal{M}$. Hence, without loss of generality $v_{max}(y)$ is the indicator vector of a base and can be found by the greedy algorithm for maximum-weight base in a matroid. The final point $y(1)$ is a convex linear combination of such vectors and hence lies in the base polytope $B(\mathcal{M})$.

Remark. Wolsey's continuous greedy algorithm [45] can be viewed as a greedy process guided by $v_{max}(y) = \mathbf{e}_j$, where $\frac{\partial F}{\partial y_j}$ is the maximum partial derivative out of those where y_j can be still increased. In other words, only one coordinate is being increased at a time. In our setting, with $F(y) = \mathbf{E}[f(\hat{y})]$ and starting at $y(0) = 0$, it can be seen that y_j will increase up to its maximal possible value (which turns out to be 1) and then a new coordinate will be selected. This is equivalent to the classical greedy algorithm, which gives a 1/2-approximation.

2.4. Pipage rounding. Ageev and Sviridenko [1] developed an elegant technique for rounding solutions of linear and nonlinear programs that they called "pipage rounding." Subsequently, Srinivasan [40] and Gandhi et al. [20] interpreted some applications of pipage rounding as a deterministic variant of dependent randomized rounding. In a typical scenario, randomly rounding a fractional solution of a linear program does not preserve the feasibility of constraints, in particular equality constraints. Nevertheless, the techniques of [1, 40, 20] show that randomized rounding can be applied in a certain controlled way to guide a solution that respects certain classes of constraints. In this paper we show that the rounding framework applies quite naturally to our problem. Further, our analysis also reveals the important role of submodularity in this context.

In our setting, we have an (approximate) fractional solution of the problem $\max\{F(y) : y \in P(\mathcal{M})\}$, where $F(y) = \mathbf{E}[f(\hat{y})]$. We wish to round a fractional solution $y^* \in P(\mathcal{M})$ to a discrete solution $S \in \mathcal{I}$. In other words we want to go from a point inside $P(\mathcal{M})$ to a vertex of the polytope. As we argued above, we can assume that $y^* \in B(\mathcal{M})$; i.e., it is a convex linear combination of bases of $\mathcal{M}$.

The base polytope has a particular structure: it is known, for example, that the edges of the base polytope are all of the form $(\mathbf{1}_I, \mathbf{1}_J)$, where J is a base obtained from I by swapping one element for another [39]. This implies (and we will argue about this more precisely in section 3.2) that it is possible to move from any point $y \in B(\mathcal{M})$ to a vertex in a sequence of moves, where the direction in each move is given by a vector $\mathbf{e}_i - \mathbf{e}_j$, where $\mathbf{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ is a canonical basis vector. Moreover, in each step we have a choice of either $\mathbf{e}_i - \mathbf{e}_j$ or $\mathbf{e}_j - \mathbf{e}_i$. The crucial property of $F(y)$ that makes this procedure work is the following.

Claim. For any $y \in [0,1]^X$ and $i, j \in X$, the function $F^y_{ij}(t) = F(y + t(\mathbf{e}_i - \mathbf{e}_j))$ is convex.

We prove this by using the smooth submodularity of $F(y) = \mathbf{E}[f(\hat{y})]$. We have $F^y_{ij}(t) = F(y + t\mathbf{d})$, where $\mathbf{d} = \mathbf{e}_i - \mathbf{e}_j$. The coordinates of $\mathbf{d}$ are $d_i = 1$ and $d_j = -1$, and the rest are 0. By differentiating $F^y_{ij}(t)$ twice with respect to t, we obtain

$$\frac{d^2 F^y_{ij}}{dt^2} = \sum_{\alpha,\beta} \frac{\partial^2 F}{\partial y_\alpha \partial y_\beta} d_\alpha d_\beta = \frac{\partial^2 F}{\partial {y_i}^2} - 2\frac{\partial^2 F}{\partial y_i \partial y_j} + \frac{\partial^2 F}{\partial {y_j}^2}.$$

As we discussed in section 2.1, $\frac{\partial^2 F}{\partial {y_i}^2} = \frac{\partial^2 F}{\partial {y_j}^2} = 0$, while $\frac{\partial^2 F}{\partial y_i \partial y_j} \leq 0$. Therefore, $\frac{d^2 F^y_{ij}}{dt^2} \geq 0$ and the function $F^y_{ij}(t)$ is convex.

The convexity of $F^y_{ij}(t) = F(y + t(\mathbf{e}_i - \mathbf{e}_j))$ allows us in each step to choose one of two possible directions, $\mathbf{e}_i - \mathbf{e}_j$ or $\mathbf{e}_j - \mathbf{e}_i$, so that the value of $F(y)$ does not decrease. Eventually, we reach a vertex of the polytope and hence a discrete solution such that $f(S) \geq F(y^*) \geq (1 - 1/e)OPT$. We will in fact present a randomized variant of this technique, where we choose a random direction in each step, and we prove a guarantee in expectation. We defer further details to section 3.2.

3. The algorithm for submodular maximization subject to a matroid constraint.

Here we describe in detail the two main components of our algorithm. On a high level, the algorithm works as follows.

The algorithm.

Given: matroid $\mathcal{M} = (X, \mathcal{I})$ (membership oracle), monotone submodular $f : 2^X \to \mathcal{R}_+$ (value oracle).

1. Run **ContinuousGreedy($f, \mathcal{M}$)** to obtain a fractional solution $y^* \in B(\mathcal{M})$.
2. Run **PipageRound($\mathcal{M}, y^*$)** to obtain a discrete solution $S \in \mathcal{I}$.

3.1. The continuous greedy algorithm.

The first stage of our algorithm handles the continuous optimization problem $\max\{F(y) : y \in P(\mathcal{M})\}$. The continuous greedy process (section 2.3) provides a guide on how to design our algorithm. It remains to deal with two issues.

- To obtain a finite algorithm, we need to discretize the time scale. This introduces some technical issues regarding the granularity of our discretization and the error incurred.
- In each step, we need to find $v_{max}(y) = \text{argmax}_{v \in P(\mathcal{M})}(v \cdot \nabla F(y))$. Apart from estimating ∇F (which can be done by random sampling), observe that this amounts to a *linear optimization* problem over $P(\mathcal{M})$. This means finding a maximum-weight independent set in a matroid, a task which can be solved easily.

ALGORITHM **ContinuousGreedy**$(f, \mathcal{M})$:

1. Let $\delta = \frac{1}{9d^2}$, where $d = r_{\mathcal{M}}(X)$ (the rank of the matroid). Let $n = |X|$. Start with $t = 0$ and $y(0) = \mathbf{0}$.
2. Let $R(t)$ contain each j independently with probability $y_j(t)$. For each $j \in X$, let $\omega_j(t)$ be an estimate of $\mathbf{E}[f_{R(t)}(j)]$, obtained by taking the average of $\frac{10}{\delta^2}(1 + \ln n)$ independent samples of $R(t)$.
3. Let $I(t)$ be a maximum-weight independent set in $\mathcal{M}$, according to the weights $\omega_j(t)$. We can find this by the greedy algorithm. Let
$$y(t + \delta) = y(t) + \delta \cdot \mathbf{1}_{I(t)}.$$
4. Increment $t := t + \delta$; if $t < 1$, go back to step 2. Otherwise, return $y(1)$.

The fractional solution found by the continuous greedy algorithm is a convex combination of independent sets, $y(1) = \delta \sum_t \mathbf{1}_{I(t)} \in P(\mathcal{M})$. Since the independent sets $I(t)$ are obtained by maximization with nonnegative weights, we can assume that each $I(t)$ is a base of $\mathcal{M}$. Therefore, $y(1) \in B(\mathcal{M})$.

The crucial inequality that allows us to analyze the performance of this stage is the following lemma (analogous to (2.5)). In the following, we denote by $OPT = \max_{S \in \mathcal{I}} f(S)$ the optimum to our problem.

LEMMA 3.1. *Consider any $y \in [0,1]^X$, and let R denote a random set corresponding to $\hat{y}$, with elements sampled independently according to y_j. Then*

$$OPT \leq F(y) + \max_{I \in \mathcal{I}} \sum_{j \in I} \mathbf{E}[f_R(j)].$$

Proof. Fix an optimal solution $O \in \mathcal{I}$. By submodularity, we have $OPT = f(O) \leq f(R) + \sum_{j \in O} f_R(j)$ for any set R. By taking the expectation over a random R as above, $OPT \leq \mathbf{E}[f(R) + \sum_{j \in O} f_R(j)] = F(y) + \sum_{j \in O} \mathbf{E}[f_R(j)] \leq F(y) + \max_{I \in \mathcal{I}} \sum_{j \in I} \mathbf{E}[f_R(j)]$. ◻

In each step, our algorithm tries to find a set $I \in \mathcal{I}$ maximizing $\sum_{j \in I} \mathbf{E}[f_R(j)]$. However, instead of the true expectations $\mathbf{E}[f_R(j)]$, we use the estimates $\omega_j(t)$. The following lemma characterizes how much we lose by using these estimates. In the next lemma and the one after that, we use the term "with high probability" to mean with probability at least $(1 - 1/\text{poly}(n))$.

LEMMA 3.2. *With high probability, the algorithm for* every *t finds a set $I(t)$ such that*

$$\sum_{j \in I(t)} \mathbf{E}[f_{R(t)}(j)] \geq (1 - 2d\delta) OPT - F(y(t)).$$

Proof. Recall that $I(t)$ is an independent set maximizing $\sum_{j \in I} \omega_j(t)$, where $\omega_j(t)$ are our estimates of $\mathbf{E}[f_{R(t)}(j)]$. Let us call an estimate *bad* if $|\omega_j(t) - \mathbf{E}[f_{R(t)}(j)]| > \delta \cdot OPT$. We prove that with high probability there is no bad estimate throughout the algorithm.

Consider an estimate $\omega_j(t)$ of $\mathbf{E}[f_{R(t)}(j)]$, obtained using $\frac{10}{\delta^2}(1 + \ln n)$ independent samples R_i of $R(t)$. We apply Theorem 2.2 with $X_i = (f_{R_i}(j) - \mathbf{E}[f_{R(t)}(j)])/OPT$, $k = \frac{10}{\delta^2}(1 + \ln n)$, and $a = \delta k$. We have $|X_i| \leq 1$, because $OPT \geq \max_j f(j) \geq \max_{R,j} f_R(j)$. The estimate is bad if and only if $|\sum X_i| > a$, and Theorem 2.2 implies that $\Pr[|\sum X_i| > a] \leq 2e^{-a^2/2k} = 2e^{-\delta^2 k/2} \leq 1/(18n^5)$. In each time step, we compute n such estimates, and the total number of steps is $1/\delta = 9d^2$. By the union bound, the probability that there is any bad estimate is at most $\frac{d^2}{2n^4}$.

Therefore, with probability at least $(1-\frac{1}{2n^2})$, all estimates are good; i.e., we have $|\omega_j(t) - \mathbf{E}[f_{R(t)}(j)]| \leq \delta \cdot OPT$ for all j, t.

Let $M = \max_{I \in \mathcal{I}} \sum_{j \in I} \mathbf{E}[f_{R(t)}(j)]$, and let I^* be a set achieving this maximum. By Lemma 3.1, we know that $M \geq OPT - F(y(t))$. We have $\sum_{j \in I^*} \omega_j \geq M - d\delta \cdot OPT$, and since our algorithm finds a set $I(t) \in \mathcal{I}$ maximizing $\sum_{j \in I} \omega_j$, we also have $\sum_{j \in I(t)} \omega_j \geq M - d\delta \cdot OPT$. Finally,

$$\sum_{j \in I(t)} \mathbf{E}[f_{R(t)}(j)] \geq \sum_{j \in I(t)} \omega_j - d\delta \cdot OPT \geq M - 2d\delta \cdot OPT \geq (1-2d\delta)OPT - F(y(t)). \qquad \square$$

Now we prove the main result of this section.

LEMMA 3.3. *With high probability, the fractional solution y found by the continuous greedy algorithm satisfies*

$$F(y) \geq \left(1 - \frac{1}{e} - \frac{1}{3d}\right) \cdot OPT.$$

Proof. We start with $F(y(0)) = 0$. Our goal is to estimate how much $F(y(t))$ increases during one iteration of the algorithm. Consider a random set $R(t)$ corresponding to $\hat{y}(t)$ and an independently random set $D(t)$ that contains each item j independently with probability $\Delta_j(t) = y_j(t+\delta) - y_j(t)$. In other words $\Delta(t) = y(t+\delta) - y(t) = \delta \cdot \mathbf{1}_{I(t)}$ and $D(t)$ is a random subset of $I(t)$ where each element appears independently with probability δ. It can easily be seen that $F(y(t+\delta)) = \mathbf{E}[f(R(t+\delta))] \geq \mathbf{E}[f(R(t) \cup D(t))]$. This follows from monotonicity, because $R(t+\delta)$ contains items independently with probabilities $y_j(t) + \Delta_j(t)$, while $R(t) \cup D(t)$ contains items independently with (smaller) probabilities $1 - (1 - y_j(t))(1 - \Delta_j(t))$. The two distributions can be coupled so that $R(t) \cup D(t)$ is a subset of $R(t+\delta)$, and hence the inequality follows.

Now we are ready to estimate how much $F(y)$ gains at time t. It is important that the probability that any item appears in $D(t)$ be very small, so we can focus on the contributions from sets $D(t)$ that turn out to be singletons. From the discussion above, we obtain

$$\begin{aligned} F(y(t+\delta)) - F(y(t)) &\geq \mathbf{E}[f(R(t) \cup D(t)) - f(R(t))] \geq \sum_j \Pr[D(t) = \{j\}]\ \mathbf{E}[f_{R(t)}(j)] \\ &= \sum_{j \in I(t)} \delta(1-\delta)^{|I(t)|-1} \mathbf{E}[f_{R(t)}(j)] \geq \delta(1 - d\delta) \sum_{j \in I(t)} \mathbf{E}[f_{R(t)}(j)], \end{aligned}$$

using $|I(t)| = d$ in the last inequality. Applying Lemma 3.2, with high probability, for each t, we get

$$F(y(t+\delta)) - F(y(t)) \geq \delta(1 - d\delta)\left((1 - 2d\delta)OPT - F(y(t))\right) \geq \delta(\tilde{OPT} - F(y(t))),$$

where we set $\tilde{OPT} = (1 - 3d\delta)OPT$. From here, $\tilde{OPT} - F(y(t+\delta)) \leq (1-\delta)(\tilde{OPT} - F(y(t)))$ and, by induction, $\tilde{OPT} - F(y(k\delta)) \leq (1-\delta)^k\ \tilde{OPT}$. For $k = 1/\delta$, we get

$$\tilde{OPT} - F(y(1)) \leq (1-\delta)^{1/\delta}\ \tilde{OPT} \leq \frac{1}{e} \tilde{OPT}.$$

Therefore, $F(y(1)) \geq (1 - 1/e)\tilde{OPT} \geq (1 - 1/e - 3d\delta)OPT = (1 - 1/e - 1/(3d))OPT$. Note that the probability of success is the same as that in Lemma 3.2. $\square$

Remark. The error term $1/(3d)$ can be made ϵ/d for any fixed $\epsilon > 0$, but in fact it can be eliminated altogether. As we prove in Appendix A, when the discretization step δ is chosen small enough, the continuous greedy algorithm achieves a clean approximation factor of $1 - 1/e$.

Here we sketch an alternative way to eliminate the error term, using partial enumeration [22, 27]. We argue as follows: For each $x \in X$, we run the algorithm above with F_x instead of F, where $F_x(y) = \mathbf{E}[f_{\{x\}}(\hat{y})]$, and $\mathcal{M}_x$ instead of $\mathcal{M}$, where $\mathcal{M}_x = \mathcal{M}/x$ is the matroid with element x contracted (A is independent in $\mathcal{M}_x$ iff $A + x$ is independent in $\mathcal{M}$). Since the optimum solution has d elements, by submodularity there is an element x such that $f(\{x\}) \geq OPT/d$. We assume that this is the element chosen by the algorithm. The optimum solution in the reduced instance has value $OPT - f(\{x\})$, and the continuous greedy algorithm returns a point $y \in B(\mathcal{M}_x)$ such that $F_x(y) \geq (1 - \frac{1}{e} - \frac{1}{3d})(OPT - f(\{x\}))$. Hence, $y' = y + \mathbf{e}_x$ is feasible in $B(\mathcal{M})$ and

$$F(y') = f(\{x\}) + F_x(y) \geq \left(1 - \frac{1}{e} - \frac{1}{3d}\right) OPT + \frac{1}{e} f(\{x\}) \geq \left(1 - \frac{1}{e} + \frac{1}{ed} - \frac{1}{3d}\right) OPT.$$

So in fact we gain a $\Theta(\frac{1}{d} OPT)$ term. Our bounds holds with probability at least $1 - 1/\mathrm{poly}(n)$, and hence we also get a clean factor $1 - 1/e$ in expectation.

3.2. The pipage rounding algorithm. We start with a point y^* in the base polytope $B(\mathcal{M})$. The pipage rounding technique aims to convert y^* into an integral solution, corresponding to a vertex of $B(\mathcal{M})$. Given $y \in [0,1]^n$ we say that i is fractional in y if $0 < y_i < 1$. Our goal is to gradually eliminate all fractional variables.

In the following, we use $y(A) = \sum_{i \in A} y_i$. For $y \in P(\mathcal{M})$, a set $A \subseteq X$ is *tight* if $y(A) = r_{\mathcal{M}}(A)$. The following well-known proposition follows from the submodularity of the rank function $r_{\mathcal{M}}$.

PROPOSITION 3.4. *If $y \in P(\mathcal{M})$ and A, B are two tight sets with respect to y, then $A \cap B$ and $A \cup B$ are also tight with respect to y.*

Proof. Using $y(A) = r_{\mathcal{M}}(A)$, $y(B) = r_{\mathcal{M}}(B)$, $y(A \cap B) \leq r_{\mathcal{M}}(A \cap B)$, $y(A \cup B) \leq r_{\mathcal{M}}(A \cup B)$, the submodularity of $r_{\mathcal{M}}$, and the linearity of y, we get

$$\begin{aligned} y(A) + y(B) &= r_{\mathcal{M}}(A) + r_{\mathcal{M}}(B) \geq r_{\mathcal{M}}(A \cap B) + r_{\mathcal{M}}(A \cup B) \\ &\geq y(A \cap B) + y(A \cup B) = y(A) + y(B). \end{aligned}$$

Therefore, all the inequalities above must be tight. ∎

We are interested in tight sets that contain a fractional variable. Observe that a tight set with a fractional variable has at least two fractional variables. Given a tight set T with fractional variables i, j, we let $y_{ij}(t) = y + t(\mathbf{e}_i - \mathbf{e}_j)$. In other words, we add t to y_i, subtract t from y_j, and leave the other values unchanged. We define a function of one variable F_{ij}^y, where $F_{ij}^y(t) = F(y_{ij}(t))$. As we argued in section 2.4, $F_{ij}^y(t)$ is convex, and hence it must be nondecreasing either for $t > 0$ or $t < 0$. Therefore we can modify the fractional variables y_i, y_j by increasing one of them and decreasing the other until we hit another constraint.

Instead of checking which direction is more profitable (as in [3]), we make the choice random. We move from y to a random point y', choosing the probabilities so that $\mathbf{E}[y' \mid y] = y$; due to convexity, this preserves the guarantee on $F(y)$ *in expectation.* (We analyze this step more formally in the proof of Lemma 3.5.) This has the advantage that the rounding technique is *oblivious*—it does not need to access the objective function. For instance, we can use it even in settings where we have only approximate-oracle access to $f(S)$.

In order to find the first constraint that would be violated in a direction $\mathbf{e}_i - \mathbf{e}_j$, we need to minimize the function $r_{\mathcal{M}}(S) - y(S)$ over $\mathcal{A} = \{S \subseteq X : i \in S, j \notin S\}$. Since $r_{\mathcal{M}}(S) - y(S)$ is a submodular function, minimization can be implemented in polynomial time [23, 18, 38]. In fact, in this particular case, we do not need the full power of submodular minimization; there is a simpler algorithm due to Cunningham [10].

After hitting a new constraint, we obtain a new tight set A'. Then we either produce a new integral variable (in which case we restart with $T = X$), or we continue with $T \cap A'$ which is a smaller set and again tight (due to Proposition 3.4). A formal description of the algorithm is given below.

Subroutine **HitConstraint**(y, i, j):
  Denote $\mathcal{A} = \{A \subseteq X : i \in A, j \notin A\}$;
  Find $\delta = \min_{A \in \mathcal{A}}(r_{\mathcal{M}}(A) - y(A))$ and $A \in \mathcal{A}$ achieving this;
  If $y_j < \delta$, then $\{y_i \leftarrow y_i + y_j,\ y_j \leftarrow 0,\ A' \leftarrow \{j\}\}$
  Else $\{y_i \leftarrow y_i + \delta,\ y_j \leftarrow y_j - \delta,\ A' \leftarrow A\}$;
  Return (y, A').

ALGORITHM **PipageRound**$(\mathcal{M}, y)$:
  While (y is not integral) do
    $T \leftarrow X$;
    While (T contains fractional variables) do
      Pick $i, j \in T$ fractional;
      $(y^+, A^+) \leftarrow$ **HitConstraint**(y, i, j);
      $(y^-, A^-) \leftarrow$ **HitConstraint**(y, j, i);
      If $(y^+ = y^- = y)$, then
        $T \leftarrow T \cap A^+$
      Else
        $p \leftarrow ||y^+ - y|| / ||y^+ - y^-||$;
        With probability p, $\{y \leftarrow y^-, T \leftarrow T \cap A^-\}$
                Else $\{y \leftarrow y^+, T \leftarrow T \cap A^+\}$;
    EndWhile
  EndWhile
  Output y.

PipageRound$(\mathcal{M}, y)$ is the main procedure. The subroutine **HitConstraint**(y, i, j) finds the nearest constraint in the direction $\mathbf{e}_i - \mathbf{e}_j$ and returns the point of intersection as well as a new tight set (which could be a singleton if a variable has become integral).

We remark on a technical point in the procedure **PipageRound**. When considering the current tight set T with fractional variables i and j, it may be the case that there are strictly smaller tight sets containing i or j; in this case we cannot change y_i and y_j and need to find a smaller tight set containing two fractional variables. The procedure **HitConstraint**(y, i, j) returns (y^+, A^+), where $y^+ = y$ and A^+ is a smaller tight set containing i, and similarly **HitConstraint**(y, j, i) returns (y^-, A^-), where $y^- = y$ and A^- is a smaller tight set containing j. **PipageRound** picks a new tight $T = T \cap A^+$ ($T = T \cap A^-$ would work as well). The new tight set is smaller, and hence eventually the algorithm finds two fractional variables that it can change.

LEMMA 3.5. *Given $y^* \in B(\mathcal{M})$,* **PipageRound**$(\mathcal{M}, y^*)$ *outputs in polynomial time an integral solution $S \in \mathcal{I}$ of value $\mathbf{E}[f(S)] \geq F(y^*)$.*

Proof. The algorithm does not alter a variable y_i once $y_i \in \{0,1\}$. An invariant maintained by the algorithm is that $y \in B(\mathcal{M})$ and T is a tight set. To verify that $y \in B(\mathcal{M})$, observe that $y(X) = r_{\mathcal{M}}(X)$ remains unchanged throughout the algorithm; we need to check that $y(S) \leq r_{\mathcal{M}}(S)$ remains satisfied for all sets S. Consider the subroutine **HitConstraint**. The only sets whose value $y(S)$ increases are those containing i and not containing j, i.e., $S \in \mathcal{A}$. We increase y_i by at most $\delta = \min_{S \in \mathcal{A}}(r_{\mathcal{M}}(S) - y(S))$; therefore $y(S) \leq r_{\mathcal{M}}(S)$ is still satisfied for all sets. We also make sure that we do not violate nonnegativity, by checking whether $y_j < \delta$. In case $y_j < \delta$, the procedure makes y_j zero and returns $A = \{j\}$.

Concerning the tightness of T, we initialize $T \leftarrow X$, which is tight because $y \in B(\mathcal{M})$. After calling **HitConstraint**, we obtain sets A^+, A^-, which are tight for y^+ and y^-, respectively. The new set T is obtained by taking an intersection with one of these sets; in either case, we get a new tight set T at the end of the inner loop, due to Proposition 3.4. Note that each of the sets A^+, A^- returned by **HitConstraint** contains exactly one of the elements i, j. Therefore, the size of T decreases after the execution of the inner loop. As long as we do not make any new variable integral, one of the fractional variables y_i, y_j is still in the new tight set T and so we can in fact find a pair of fractional variables in T. However, due to the decreasing size of T, we cannot repeat this more than $n-1$ times. At some point, we must make a new variable integral, and then we restart the inner loop with $T = X$. The outer loop can also iterate at most n times, since the number of integral variables increases after each outer loop.

Finally, we need to show that the expected value of the final solution is $\mathbf{E}[f(S)] \geq F(y^*)$. In each step, we move from y to a random point $y' \in \{y^+, y^-\}$, y^- with probability p or y^+ with probability $1-p$. We have $F(y^+) = F^y_{ij}(\epsilon^+)$ and $F(y^-) = F^y_{ij}(\epsilon^-)$, where $\epsilon^- < 0 < \epsilon^+$. The probabilities are chosen so that $p\epsilon^- + (1-p)\epsilon^+ = 0$, i.e., $\mathbf{E}[y' \mid y] = y$. By Jensen's inequality, using the convexity of F^y_{ij}, we obtain

$$\mathbf{E}[F(y')] = pF^y_{ij}(\epsilon^-) + (1-p)F^y_{ij}(\epsilon^+) \geq F^y_{ij}(0) = F(y).$$

By induction on the number of steps of the rounding procedure, we obtain $\mathbf{E}[f(S)] \geq F(y^*)$, where $y^* \in B(\mathcal{M})$ is the initial point and S is the final integral solution. ∎

3.3. Simplification for partition matroids. In the special case of a *simple partition matroid*, we can simplify the algorithm by essentially skipping the pipage rounding stage. This type of matroid appears in several applications.

DEFINITION 3.6. *For a ground set partitioned into $X = X_1 \cup X_2 \cup \cdots \cup X_k$, the simple partition matroid is $\mathcal{M} = (X, \mathcal{I})$, where*

$$\mathcal{I} = \{S \subseteq X : \forall i; |S \cap X_i| \leq 1\}.$$

By the definition of the matroid polytope, $P(\mathcal{M}) = \{y \in \mathcal{R}_+^X : \forall i; \sum_{j \in X_i} y_j \leq 1\}$. Therefore, it is natural to interpret the variables y_j as probabilities and round a fractional solution by choosing exactly one random element in each X_i, with probabilities according to y (we can assume that $y \in B(\mathcal{M})$ and hence $\sum_{j \in X_i} y_j = 1$ for each X_i). The following lemma confirms that this works.

LEMMA 3.7. *Let $X = X_1 \cup \cdots \cup X_k$, let $f : 2^X \to \mathcal{R}_+$ be a monotone submodular function, and let $y \in B(\mathcal{M})$, where $\mathcal{M}$ is a simple partition matroid on $X = X_1 \cup \cdots \cup X_k$. Let T be a random set where we sample independently from each X_i exactly one random element, element j with probability y_j. Then*

$$\mathbf{E}[f(T)] \geq F(y).$$

Proof. Let T be sampled as above, and let $\hat{y} = \mathbf{1}_R$. The difference between R and T is that in R each element appears independently with probability y_j (and therefore R is not necessarily an independent set in $\mathcal{M}$). In T, we sample exactly one element from each X_i, with the same probabilities. We claim that $\mathbf{E}[f(T)] \geq \mathbf{E}[f(R)]$.

We proceed by induction on k. First, assume that $k = 1$ and $X = X_1$. Then,

$$\begin{aligned}
\mathbf{E}[f(R)] &= \sum_{R \subseteq X_1} \Pr[R] f(R) \\
&\leq \sum_{R \subseteq X_1} \Pr[R] \sum_{j \in R} f(\{j\}) \\
&= \sum_{j \in X_1} \Pr[j \in R] f(\{j\}) = \sum_{j \in X_1} y_j f(\{j\}) = \mathbf{E}[f(T)].
\end{aligned}$$

Now let $k > 1$. We denote $T' = T \cap (X_1 \cup \cdots \cup X_{k-1})$, $T'' = T \cap X_k$, and $R' = R \cap (X_1 \cup \cdots \cup X_{k-1})$, $R'' = R \cap X_k$. Then,

$$\mathbf{E}[f(T)] = \mathbf{E}[f(T') + f_{T'}(T'')] \geq \mathbf{E}[f(T') + f_{T'}(R'')] = \mathbf{E}[f(T' \cup R'')],$$

using the base case for $f_{T'}$ on X_k. Then,

$$\mathbf{E}[f(T' \cup R'')] = \mathbf{E}[f(R'') + f_{R''}(T')] \geq \mathbf{E}[f(R'') + f_{R''}(R')] = \mathbf{E}[f(R)],$$

using the induction hypothesis for $f_{R''}$ on $X_1 \cup \cdots \cup X_{k-1}$. $\square$

Therefore, we can replace pipage rounding for simple partition matroids by the following simple procedure. (This procedure can also be seen as performing all the steps of pipage rounding at once.)

Simple rounding.

- Given $y \in \mathcal{R}_+^X$ such that for all $i, \sum_{j \in X_i} y_j = 1$, sample independently from each X_i exactly one element: element $j \in X_i$ with probability y_j. Return the set T of the sampled elements.

4. Applications.

4.1. Submodular welfare maximization. Here we describe an application of our framework to the *submodular welfare problem*. First, we review the general framework and known results.

Social welfare maximization. *Given a set X of m items, and n players, each having a monotone utility function $w_i : 2^X \to \mathcal{R}^+$, the goal is to partition X into disjoint subsets $S_1, \ldots, S_n$ in order to maximize the social welfare $\sum_{i=1}^n w_i(S_i)$.*

This problem arises in combinatorial auctions and has been studied intensively in recent years [32, 26, 11, 14, 16, 35]. Before studying its complexity status, one needs to specify how the algorithm accesses the input to the problem. Usually, an algorithm is allowed to ask certain queries about the players' utility functions. This leads to different *oracle models* of computation. The two types of oracles most commonly considered are the following:

- *Value oracle:* This returns the value of $w_i(S)$ for a given player i and $S \subseteq X$.
- *Demand oracle:* This returns a set S maximizing $w_i(S) - \sum_{j \in S} p_j$ for a given player i and an assignment of prices p_j.

Previous work. In general, social welfare maximization contains set packing as a special case (the case of *single-minded bidders*), and hence an $m^{1/2-\epsilon}$-approximation for any fixed $\epsilon > 0$, even in the demand oracle model, would imply $P = NP$ [32].

Positive results have been achieved only under strong assumptions on the utility functions. A particular case of interest is when the utility functions are submodular—this is what we call the submodular welfare problem.

This problem (under a different name) was considered by Fisher, Nemhauser, and Wolsey [17] as a motivating example for submodular maximization subject to a matroid constraint. They showed that the greedy algorithm achieves a $1/2$-approximation in the value oracle model, in fact even for the on-line variant, where items are arriving one by one—this follows from the "locally greedy heuristic" in [17] and was rediscovered in [32]. On the hardness side, it has been shown that a $(1-1/e+\epsilon)$-approximation for any fixed $\epsilon > 0$ would imply $P = NP$ [26]. This hardness result holds in particular for submodular utilities induced by explicitly given set coverage systems. A $(1-1/e)$-approximation has been found in this case [11], and also in the case of general submodular utilities, but using the stronger *demand oracle model*. In this model, the $(1-1/e)$-approximation has been subsequently generalized to all fractionally subadditive functions [14] and the ratio for submodular functions has been improved to $1-1/e+\delta$ for some fixed $\delta > 0$ [16]. This does not contradict the hardness result of [26], which holds only in the *value oracle model*. Moreover, recently it has been proved that in the value oracle model a $(1-1/e+\epsilon)$-approximation for the submodular welfare problem would require exponentially many value queries [35]. It was an open problem whether a $(1-1/e)$-approximation can be achieved for the submodular welfare problem using only value queries.

Our result. Since the submodular welfare problem can be seen as a special case of SUB-M [17], our work implies a randomized $(1-1/e)$-approximation, thus resolving the status of the problem in the value oracle model (this result first appeared in [42]). For completeness, we describe the reduction to SUB-M.

The reduction. For a given set of items A and number of players n, we define a ground set $X = [n] \times A$. The elements of X can be viewed as clones of items, one clone of each item for each player. For each player i, we define a mapping $\pi_i : 2^X \to 2^A$,

$$\pi_i(S) = \{j \in A \mid (i,j) \in S\},$$

which simply takes all the clones in S corresponding to player i. Given utility functions $w_1, \ldots, w_n : 2^A \to \mathcal{R}^+$, we define a function $f : 2^X \to \mathcal{R}^+$,

$$f(S) = \sum_{i=1}^{n} w_i(\pi_i(S)).$$

It can be seen that if w_i is submodular, then $w_i \circ \pi_i$ is submodular, and hence f is submodular as well.

The problem of partitioning $A = S_1 \cup S_2 \cup \cdots \cup S_n$ so that $\sum_{i=1}^{n} w_i(S_i)$ is maximized is equivalent to finding $S = \bigcup_{i=1}^{n}(\{i\} \times S_i) \subseteq X$, containing at most one clone of each item, so that $f(S)$ is maximized. Sets of this type form a partition matroid:

$$\mathcal{I} = \{S \subseteq X \mid \forall j; |S \cap X_j| \leq 1\},$$

where $X_j = [n] \times \{j\}$. Note that the rank of this matroid is $m = |A|$, the number of items. Therefore, the welfare maximization problem is equivalent to maximizing $f(S)$ subject to $S \in \mathcal{I}$.

Our algorithm yields a $(1-1/e)$-approximation for the submodular welfare problem, which is optimal, as we mentioned above. Moreover, we can simplify the algorithm by skipping the pipage rounding stage. This is possible because we are dealing with a simple partition matroid, as we discussed in section 3.3. The fractional variables are associated with elements of $[n] \times A$, i.e., player-item pairs, and it is natural to denote them by y_{ij}. Each set $X_j = [n] \times \{j\}$ consists of all the clones of item j. By Lemma 3.7, the fractional solution obtained by the continuous greedy algorithm can be rounded by taking one random clone of each item, i.e., allocating each item to a random player, with probabilities y_{ij}. Reinterpreting our continuous greedy algorithm in this setting, we obtain the following. Note that $|X| = mn$ and the rank of the matroid is $m = |A|$.

The continuous greedy algorithm for the submodular welfare problem.

1. Let $\delta = \frac{1}{9m^2}$ where m is the number of items. Start with $t = 0$ and $y_{ij}(0) = 0$ for all i, j.
2. Let $R_i(t)$ be a random set containing each item j independently with probability $y_{ij}(t)$. For all i, j, estimate the expected marginal profit of player i from item j,
$$\omega_{ij}(t) = \mathbf{E}[w_i(R_i(t) + j) - w_i(R_i(t))],$$
by taking the average of $\frac{10}{\delta^2}(1 + \ln(mn))$ independent samples.
3. For each j, let $i_j(t) = \operatorname{argmax}_i \omega_{ij}(t)$ be the *preferred player* for item j (breaking possible ties arbitrarily). Set $y_{ij}(t+\delta) = y_{ij}(t) + \delta$ for the preferred player $i = i_j(t)$ and $y_{ij}(t + \delta) = y_{ij}(t)$ otherwise.
4. Increment $t := t + \delta$; if $t < 1$, go back to step 2.
5. Allocate each item j independently, with probability $y_{ij}(1)$ to player i.

4.2. The generalized and separable assignment problems. Here we consider an application of our techniques to the generalized assignment problem (GAP). An instance of GAP consists of n bins and m items. Each item i has two nonnegative numbers for each bin j: a value v_{ij} and a size s_{ij}. We seek an assignment of items to bins such that the total size of items in each bin is at most one[2] and the total value of all items is maximized. In [19], a $(1 - 1/e)$-approximation algorithm for GAP is presented. The algorithm is based on solving an exponential sized linear program. The separation oracle for the dual of this linear program is the knapsack problem, and one obtains an arbitrarily close approximation to it by using an FPTAS for knapsack. In [16], it is shown that the worst-case integrality gap is in fact $1 - 1/e + \delta$ for some constant $\delta > 0$, thus resulting in an improved approximation. It is also known that unless $P = NP$ there is no 10/11 approximation for GAP [5].

Our techniques yield a $(1-1/e-o(1))$-approximation algorithm for GAP. Previous algorithms that achieve a comparable ratio [19, 16] rely on solving a large linear program. Although the ratio guaranteed by our algorithm is slightly worse than the $(1 - 1/e + \delta)$-approximation of [16], we believe that the algorithm is conceptually simple and with additional work can be made efficient in practice. Our algorithm also generalizes to other assignment problems for which the factor $1 - 1/e$ is known to be optimal.

The separable assignment problem. An instance of the separable assignment problem (SAP) consists of m items and n bins. Each bin j has an associated collection of *feasible sets* $\mathcal{F}_j$ which is down-closed ($A \in \mathcal{F}_j, B \subseteq A \Rightarrow B \in \mathcal{F}_j$). Each item i has a value v_{ij}, depending on the bin j where it is placed. The goal is to choose disjoint

[2]Sometimes GAP is defined with bins having specified capacities. Since the item sizes are allowed to vary with the bin, one can scale them to reduce to an instance in which each bin capacity is 1.

feasible sets $S_j \in \mathcal{F}_j$ so as to maximize $\sum_{j=1}^{n} \sum_{i \in S_j} v_{ij}$. For computational purposes we assume that each $\mathcal{F}_j$ is accessed via a membership oracle.

Reduction to a matroid constraint. SAP can be cast as a special case of SUB-M as follows: We define the ground set X to be $\{(j, S) \mid 1 \leq j \leq n, S \in \mathcal{F}_j\}$. The interpretation of X is that for each bin j we create a separate copy of the items and these can be packed only in j. Note that this allows us to pack an original item in multiple bins. We define a function $f : 2^X \to \mathcal{R}_+$,

$$f(\mathcal{S}) = \sum_i \max_j \{v_{ij} : \exists (j, S) \in \mathcal{S}, i \in S\}.$$

It is easy to check that f is monotone and submodular. We observe that, for an item i that is potentially packed in multiple bins (via copies in X), f gives only the profit for the most profitable copy. We maximize this function subject to a matroid constraint $\mathcal{M} = (X, \mathcal{I})$, where $\mathcal{S} \in \mathcal{I}$ iff $\mathcal{S}$ contains at most one pair (j, S_j) for each j. Such a set $\mathcal{S}$ corresponds to an assignment of set S_j to bin j for each $(j, S_j) \in \mathcal{S}$. $\mathcal{M}$ is a simple partition matroid constraint on X.

The approximate greedy process. Before we describe our algorithm for SAP, we need to discuss a generalization of our framework. Let us consider a setting where we cannot optimize linear functions over $P(\mathcal{M})$ exactly but only α-approximately ($\alpha < 1$). Let us consider the continuous setting (section 2.3). Assume that in each step we are able to find a vector $v_\alpha(y) \in P(\mathcal{M})$ such that $v_\alpha(y) \cdot \nabla F(y) \geq \alpha \max_{v \in P(\mathcal{M})} (v \cdot \nabla F(y))$. By (2.5), $v_\alpha(y) \cdot \nabla F(y) \geq \alpha(OPT - F(y))$. This leads to a differential inequality

$$\frac{dF}{dt} \geq \alpha(OPT - F(y(t)))$$

whose solution is $F(y(t)) \geq (1 - e^{-\alpha t})OPT$. At time $t = 1$, we obtain a $(1 - e^{-\alpha})$-approximation. The rest of the analysis follows as in section 3.1.

Implementing the algorithm for SAP. The ground set of $\mathcal{M}$ is exponentially large here, so we cannot use the algorithm of section 3 as a black box. However, the rank of $\mathcal{M}$ is only n (the number of bins), which is useful. This means the step size δ in the continuous greedy algorithm is polynomially small in n. The algorithm works with variables corresponding to the ground set X; let us denote them by $y_{j,S}$, where $S \in \mathcal{F}_j$. Note that in each step only n variables are incremented (one for each bin j), and hence the number of nonzero variables remains polynomial. Based on these variables, we can generate a random set $\mathcal{R} \subseteq X$ in each step. However, we cannot estimate all marginal values $\omega_{j,S} = \mathbf{E}[f_{\mathcal{R}}(j, S)]$ since these are exponentially many.

What we do is the following: Observe that

$$\omega_{j,S} = \mathbf{E}[f_{\mathcal{R}}(j, S)] = \sum_{i \in S} \mathbf{E}[f_{\mathcal{R}}(j, i)],$$

where

$$f_{\mathcal{R}}(j, i) = \max\{0, v_{ij} - \max\{v_{ij'} : \exists (j', S) \in \mathcal{R}; i \in S\}\}$$

is the marginal profit of adding item i to bin j, compared to its assignment in $\mathcal{R}$. For each item i, we estimate $\omega_{ij} = \mathbf{E}[f_{\mathcal{R}}(j, i)]$. We choose the number of samples to be $(mn)^5$ so that the error per item is bounded by $OPT/(mn)^2$ with high probability. We have $\omega_{j,S} = \sum_{i \in S} \omega_{ij}$ for any set S; our estimate of $\omega_{j,S}$ is accurate within

$OPT/(mn^2)$. Finding a maximum-weight independent set $I \in \mathcal{I}$ means finding the optimal set S_j for each bin j, given the weights ω_{ij}. This is what we call the *single-bin subproblem.* We use the item weights ω_{ij} and try to find a set $S \in \mathcal{F}_j$ maximizing $\sum_{i \in S} \omega_{ij}$. If we can solve this problem α-approximately ($\alpha < 1$), we can also find an α-approximate maximum-weight independent set I. Our sampling estimates add an $o(1)$ error to this computation. As we argued above, we obtain a $(1 - e^{-\alpha} - o(1))$-approximation for SAP. We summarize the algorithm here.

The continuous greedy algorithm for SAP/GAP.

1. Let $\delta = \frac{1}{9n^2}$. Start with $t = 0$ and $y_{j,S}(0) = 0$ for all j, S.
2. Let $\mathcal{R}(t)$ be a random collection of pairs (j, S), each pair (j, S) appearing independently with probability $y_{j,S}(t)$. For all i, j, estimate the expected marginal profit of bin j from item i,
$$\omega_{ij}(t) = \mathbf{E}_{\mathcal{R}(t)}[\max\{0, v_{ij} - \max\{v_{ij'} : \exists (j', S) \in \mathcal{R}(t); i \in S\}\}],$$
by taking the average of $(mn)^5$ independent samples.
3. For each j, find an α-approximate solution $S_j^*(t)$ to $\max\{\sum_{i \in S} \omega_{ij}(t) : S \in \mathcal{F}_j\}$. Set $y_{j,S}(t+\delta) = y_{j,S}(t) + \delta$ for the set $S = S_j^*(t)$ and $y_{j,S}(t+\delta) = y_{j,S}(t)$ otherwise.
4. Increment $t := t + \delta$; if $t < 1$, go back to Step 2.
5. For each bin j independently, choose a random set $S_j := S$ with probability $y_{j,S}(1)$. For each item occurring in multiple sets S_j, keep only the occurrence of maximum value. Allocate the items to bins accordingly.

THEOREM 4.1. *If we have an α-approximation algorithm for* $\max\{\sum_{i \in S} \omega_{ij} : S \in \mathcal{F}_j\}$, *for any bin j and any assignment of values ω_{ij}, then the continuous greedy algorithm delivers a $(1 - e^{-\alpha} - o(1))$-approximation algorithm for the corresponding SAP with families of feasible sets $\mathcal{F}_j$.*

This beats both the factor $\alpha(1 - 1/e)$ obtained by using the "configuration" linear program [19] and the factor $\alpha/(1 + \alpha)$ obtained by a simple greedy algorithm [3, 21].

GAP. Special cases of SAP are obtained by considering different types of collections of feasible sets $\mathcal{F}_j$. When each $\mathcal{F}_j$ is given by a knapsack problem, $\mathcal{F}_j = \{S : \sum_{i \in S} s_{ij} \leq 1\}$, we obtain GAP. Since there is an FPTAS for the knapsack problem, we have $\alpha = 1 - o(1)$ and we obtain a $(1 - 1/e - o(1))$-approximation for GAP.[3]

Matroid constraint on the bins. Consider GAP with the additional restriction that items be assigned to at most k of the given m bins. We can capture this additional constraint by altering the matroid $\mathcal{M}$ as follows: Previously, a set $\mathcal{S}$ was defined to be independent iff $|\mathcal{S} \cap \{(j, S) \mid S \in \mathcal{F}_j\}| \leq 1$ for each bin j. Now a set $\mathcal{S}$ is independent iff $|\mathcal{S} \cap \{(j, S) \mid S \in \mathcal{F}_j\}| \leq 1$ for each bin j *and* $|\mathcal{S}| \leq k$. It is easy to check that this is also a matroid constraint. More generally let $B = \{1, \ldots, m\}$ be the set of bins and $\mathcal{M}' = (B, \mathcal{I})$ be a given matroid on B. A generalization of GAP is obtained by asking for a maximum profit feasible assignment of items to a subset of bins $B' \subseteq B$, where B' is required to be an independent set in $\mathcal{I}$. This constraint can also be incorporated into the reduction by letting $\mathcal{M}$ be the matroid where $\mathcal{S}$ is independent iff (i) $|\mathcal{S} \cap \{(j, S) \mid S \in \mathcal{F}_j\}| \leq 1$ for each bin j, and (ii) $B_{\mathcal{S}} \in \mathcal{I}$, where $B_{\mathcal{S}} = \{j \mid \exists (j, S) \in \mathcal{S}\}$. Once again it is easy to check that $\mathcal{M}$ is a matroid. The algorithm can be implemented in a way similar to the algorithm for SAP. Pipage

[3]It may be possible to eliminate the $o(1)$ term and obtain a clean $(1 - 1/e)$-approximation by partial enumeration; however, we believe that the algorithm is simpler and more "practical" as it stands.

rounding can also be implemented by first dealing with each bin separately as in subsection 3.3 before considering the matroid $\mathcal{M}'$ as in subsection 3.2.

5. Conclusions. The notion of a multilinear extension, $F(y) = \mathbf{E}[f(\hat{y})]$, is crucial in this paper. Since it can be evaluated only probabilistically, our algorithm is intrinsically randomized. It is an interesting open problem whether a $(1 - 1/e)$-approximation can be obtained using a deterministic algorithm in the value oracle model. In some special cases, such as coverage in set systems, $F(y)$ can be evaluated deterministically—in fact, Ageev and Sviridenko [1] use explicit functions for several problems. The pipage rounding that we described is randomized, but one can use a deterministic variant [3]. Therefore, in some special cases, a deterministic $(1 - 1/e)$-approximation can be achieved.

One can also consider the multilinear extension for *nonmonotone* submodular functions. (We still assume that $f(S)$ is nonnegative.) For such functions, even the unconstrained problem $\max_{S\subseteq X} f(S)$ is APX-hard; the max-cut problem is a special case. Feige, Mirrokni, and Vondrák [15] give constant factor approximations for the unconstrained problem. Under a matroid constraint, we can still use the pipage rounding framework, which is applicable even to nonmonotone functions (as already shown in special cases by [1]). However, the continuous greedy algorithm for maximizing $F(x)$ requires monotonicity. Hence, other techniques are required to deal with the continuous optimization problem. For specific cases, one may be able to combine pipage rounding with a linear programming relaxation. For example, in [1], max-cut with fixed sizes of parts and related problems is addressed in this fashion. In recent work, the multilinear extension and pipage rounding have been used directly to derive approximation algorithms for nonmonotone submodular maximization under matroid independence/base constraints [43], improving previous results using local search [30]. The multilinear extension has also been used to design approximations for maximizing monotone and nonmonotone submodular functions subject to knapsack constraints [29, 30]. The hardness results of [43] reveal a more general connection between the approximability of submodular maximization problems and properties of their multilinear relaxations.

At present, it is not clear how to use our framework for $p > 1$ matroid constraints. The special case of $p = 2$ is of much interest since the matroid intersection polytope is integral. Although the continuous greedy algorithm is still applicable, pipage rounding does not generalize. In recent work, [31] obtains a $(1/p - \epsilon)$-approximation for monotone submodular maximization subject to $p \geq 2$ matroids via local search; the running time of the algorithm is exponential in p and $1/\epsilon$.

The original forms of pipage rounding [1] and dependent randomized rounding [40, 20] address the question of rounding a fractional solution to the assignment problem while maintaining the quality of the solution in terms of a certain objective function. A number of applications are given in [1, 40, 20]. This paper shows that submodularity and uncrossing properties of solutions in the matroid polytope and other related structures are the basic ingredients in the applicability of the pipage rounding technique. We hope this insight will lead to more applications in the future.

Appendix A. Clean $(1 - 1/e)$-approximation using smaller steps. Here we present a tighter analysis of the continuous greedy algorithm from section 3.1, showing that a choice of $\delta = 1/(40d^2n)$ is sufficient to achieve a clean approximation factor of $1 - 1/e$.

Recall the proof of Lemma 3.3. In each step, we compared the random set $R(t)$ to $R(t + \delta)$. Instead of $R(t + \delta)$, we actually analyzed $R(t) \cup D(t)$, which is a strictly smaller set. However, rather than $R(t) \cup D(t)$, we can consider $R(t) \cup \tilde{D}(t)$, where $\tilde{D}(t)$ is independent of $R(t)$ and contains each element j with probability $\Delta_j(t)(1 + y_j(t))$. It can be verified that, for any j, $Pr[j \in R(t) \cup \tilde{D}(t)] \leq Pr[j \in R(t + \delta)]$. (We would get equality if we sampled $\tilde{D}(t)$ with probabilities $\Delta_j(t)/(1 - y_j(t))$, but we use a smaller value $\Delta_j(t)(1 + y_j(t))$ so that the probabilities do not exceed 2δ.) $\tilde{D}(t)$ is a random subset of $I(t)$, and the size of $I(t)$ is d. We can repeat the analysis with $\tilde{D}(t)$ and get

$$\begin{aligned} F(y(t+\delta)) - F(y(t)) &\geq \sum_j \Pr[\tilde{D}(t) = \{j\}] \mathbf{E}[f_{R(t)}(j)] \\ &\geq \delta(1 - 2d\delta) \sum_{j \in I(t)} \mathbf{E}[f_{R(t)}(j)](1 + y_j(t)). \end{aligned}$$

Observe that we get a small gain over the previous analysis as the fractional variables $y_j(t)$ increase.

As before, with high probability all marginal values $\mathbf{E}[f_{R(t)}(j)]$ are estimated within an additive error of $\delta\, OPT$, so our computation of the maximum-weight base $I(t)$ is accurate within $2d\delta\, OPT$. We assume in the following that this is the case. Denote $\omega^*(t) = \max_{j \in I(t)} \mathbf{E}[f_{R(t)}(j)]$. By Lemma 3.2 and submodularity, we know that, at any time, $\omega^*(t) \geq \frac{1}{d}((1 - 2d\delta)OPT - F(y(t)))$. Let $j^*(t)$ be the element achieving $\omega^*(t)$. Let us call the steps where $y_{j^*(t)}(t) < \frac{1}{2n}$ "bad" and the steps where $y_{j^*(t)}(t) \geq \frac{1}{2n}$ "good." We argue that at least half of all steps must be good. If more than half of all steps were bad, the sum of all increments to $j^*(t)$ in bad steps would be more than 1/2, but then some variable would receive more than $\frac{1}{2n}$ as $j^*(t)$ in bad steps, which is a contradiction. In good steps, the improved analysis gives

$$\begin{aligned} F(y(t+\delta)) - F(y(t)) &\geq \delta(1 - 2d\delta) \left(\sum_{j \in I(t)} \mathbf{E}[f_{R(t)}(j)] + y_{j^*(t)}(t)\omega^*(t) \right) \\ &\geq \delta(1 - 2d\delta) \left(1 + \frac{1}{2dn}\right) ((1 - 2d\delta)OPT - F(y(t))) \\ &\geq \delta(1 - 8d\delta) \left(1 + \frac{1}{2dn}\right) (OPT - F(y(t))), \end{aligned}$$

assuming that $F(y(t)) \leq \frac{2}{3} OPT$ (otherwise we are done already). By taking $\delta = \frac{1}{40d^2n}$, i.e., $\frac{1}{2dn} = 20d\delta$, we make this expression bigger than $\delta(1 + 10d\delta)(OPT - F(y(t)))$. Then, we can conclude that in good steps we have

$$OPT - F(y(t+\delta)) \leq (1 - \delta(1 + 10d\delta))(OPT - F(y(t))).$$

In bad steps, we ignore the term $y_{j^*(t)}(t)\omega^*(t)$ and obtain

$$OPT - F(y(t+\delta)) \leq (1 - \delta(1 - 8d\delta))(OPT - F(y(t))).$$

Finally, using the fact that at least half of the steps are good, we get

$$OPT - F(y(1)) \leq (1 - \delta + 8d\delta^2)^{\frac{1}{2\delta}} (1 - \delta - 10d\delta^2)^{\frac{1}{2\delta}} OPT \leq (1 - \delta)^{1/\delta} OPT,$$

which means $F(y(1)) \geq (1 - (1 - \delta)^{1/\delta})OPT \geq (1 - 1/e)OPT$.

Appendix B. Analysis of the greedy algorithm for p-systems. We give an analysis of the greedy algorithm for maximizing a submodular function subject to a p-system, that is, $\max_{S \in \mathcal{I}} f(S)$, where $(X, \mathcal{I})$ is a p-system and $f : 2^X \to \mathcal{R}^+$ is a monotone submodular set function. Fisher, Nemhauser, and Wolsey [17] showed that the natural greedy algorithm has a tight approximation ratio of $1/(p+1)$. In [17], a proof is given for a special case when the $\mathcal{I}$ is induced by the intersection of p matroids on X; it is mentioned that the proof extends to an arbitrary p-system using the same outline as that of Jenkyns [25], who showed a bound of $1/p$ if f is modular. As far as we are aware, a formal proof has not appeared in the literature, and hence we give a proof here for the sake of completeness. The proof also adapts to give a bound of $\alpha/(p+\alpha)$ if only an α-approximate oracle for f is available (here $\alpha \leq 1$). We mention that Goundan and Schulz [21] have, in an independent work, adapted the proof from [17] for intersection of p matroids to the approximate setting. However, the proofs in [17, 21] use a linear programming relaxation and differ from that of Jenkyns [25], whose scheme we follow.

We recap the definition of a p-system in more detail below. For a set $Y \subseteq X$, a set J is called a *base* of Y if J is a maximal independent subset of Y; in other words, $J \in \mathcal{I}$ and for each $e \in Y \setminus J$, $J + e \notin \mathcal{I}$. Note that Y may have multiple bases, and, further, a base of Y may not be a base of a superset of Y. $(X, \mathcal{I})$ is said to be a p-system if for each $Y \subseteq X$ the cardinality of the largest base of Y is at most p times the cardinality of the smallest base of Y. In other words, for each $Y \subseteq X$,

$$\frac{\max_{J : J \text{ is a base of } Y} |J|}{\min_{J : J \text{ is a base of } Y} |J|} \leq p.$$

Special cases of p-systems. We mention three examples of independence families that are special cases of p-systems in increasing order of generality.

- Intersection of p matroids.
- p-circuit-bounded families considered in [25].[4] A family $\mathcal{I}$ is a p-circuit-bounded family if, for each $A \in \mathcal{I}$ and $e \in X \setminus A$, $A + e$ has at most p circuits. Here, a circuit is a minimally dependent set.
- p-extendible families defined in [33]. A family $\mathcal{I}$ is p-extendible if the following holds: suppose $A \subset B$, $A, B \in \mathcal{I}$, and $A + e \in \mathcal{I}$; then there is a set $Z \subseteq B \setminus A$ such that $|Z| \leq p$ and $B \setminus Z + e \in \mathcal{I}$.

It can be shown that $\mathcal{I}$ being the intersection of p matroids implies that it is p-circuit-bounded, which in turn implies that it is p-extendible, which in turn implies that it is a p-system. Examples show that these inclusions are proper [34]. An example of a natural 3-system which is not 3-extendible is the following: Given a simple graph $G = (V, E)$, define an independence system $(E, \mathcal{I})$, where a subset of edges $A \subseteq E$ is independent (that is, $A \in \mathcal{I}$) iff (V, A) is planar (the number 3 comes from Euler's formula; see [4] for details). A simple and illustrative example of a 2-extendible system is the following: Take the ground set to be the edges of an undirected graph and the independent sets to be the set of matchings in the graph.

Analysis of the greedy algorithm. The greedy algorithm includes an element of maximum possible marginal value among those elements that keep the set independent, as long as possible.

ALGORITHM **Greedy**:
 $S \leftarrow \emptyset$, $A \leftarrow \emptyset$;
 Repeat
  $A \leftarrow \{e \mid S \cup \{e\} \in \mathcal{I}\}$;

[4] In [25] no name is suggested for these families.

If ($A \neq \emptyset$), then
$e \leftarrow \text{argmax}_{e' \in A} f_S(e')$;
$S \leftarrow S \cup \{e\}$;
Endif
Until ($A = \emptyset$);
Output S;

It is easy to see that the greedy algorithm can be implemented using a value oracle for f and a membership oracle for $\mathcal{I}$. We let $e_1, e_2, \ldots, e_k$ be the elements added to S by the greedy algorithm, and for $i \geq 0$, we let S_i denote the set $\{e_1, e_2, \ldots, e_i\}$.

In some settings, $|X|$ is exponential and the greedy step in picking the element with the largest marginal value can be implemented only in an approximate way. Suppose we are guaranteed only that in each greedy step i the element e_i picked in that step satisfies $f_{S_{i-1}}(e_i) \geq \alpha \max_{e \in A_i} f_{S_{i-1}}(e)$ where A_i is the set of all candidate augmentations of S_{i-1}. Here $\alpha \leq 1$. We prove directly that the approximation ratio of the greedy algorithm is $\alpha/(p + \alpha)$.

Let $\delta_i = f_{S_{i-1}}(e_i) = f(S_i) - f(S_{i-1})$ be the improvement in the solution value when e_i is added. Note that $f(S_k) = \sum_i \delta_i$. Fix some optimum solution O. We show the existence of a partition $O_1, O_2, \ldots, O_k$ of O such that $\delta_i \geq \frac{\alpha}{p} f_{S_k}(O_i)$ for all $i \in \{1, 2, \ldots, k\}$. We allow for some i that $O_i = \emptyset$.

Assuming that we have such a partition of O,

$$f(S_k) = \sum_{i=1}^{k} \delta_i \geq \frac{\alpha}{p} \sum_{i=1}^{k} f_{S_k}(O_i) \geq \frac{\alpha}{p} f_{S_k}(O) \geq \frac{\alpha}{p}(f(O) - f(S_k)),$$

which implies that $f(S_k) \geq \frac{\alpha}{p+\alpha} f(O)$. The penultimate inequality relies on submodularity of f.

We now prove the existence of the desired partition of O. For this we use the following algorithm. Define $T_k = O$. For $i = k, k-1, \ldots, 2$, execute: Let $B_i = \{e \in T_i \mid S_{i-1} + e \in \mathcal{I}\}$. If $|B_i| \leq p$, set $O_i = B_i$; else pick an arbitrary $O_i \subset B_i$ with $|O_i| = p$. Then set $T_{i-1} = T_i \setminus O_i$ before decreasing i. After the for-loop, set $O_1 = T_1$. It is immediate that for $j = 2, \ldots, k$, $|O_j| \leq p$.

We prove by induction on $i = 0, 1, \ldots, k-1$ that $|T_{k-i}| \leq (k-i)p$. For $i = 0$, we reason as follows: When the greedy algorithm stops, S_k is a maximal independent set contained in X, and thus any independent set including $T_k = O$ satisfies $|T_k| \leq p|S_k| = pk$. For the inductive step, let $i > 0$. We have two cases: If $|B_{k-i+1}| > p$, then $|O_{k-i+1}| = p$ and $|T_{k-i}| = |T_{k-i+1}| - |O_{k-i+1}| \leq (k-i+1)p - p = (k-i)p$, using the induction hypothesis. If $|B_{k-i+1}| \leq p$, we have $T_{k-i} = T_{k-i+1} \setminus B_{k-i+1}$. Consider $Y = S_{k-i} \cup T_{k-i}$. By the way B_{k-i+1} is defined, S_{k-i} is a maximal independent set in Y. With T_{k-i} independent and contained in Y, we must have that $|T_{k-i}| \leq p|S_{k-i}| = p(k-i)$.

Thus we also have that $|T_1| = |O_1| \leq p$. Now, by construction, every $e \in O_i$ for every $i \in \{1, 2, \ldots, k\}$ satisfies that $S_{i-1} + e$ is independent. From the choice made by the greedy algorithm it follows that $\delta_i \geq \alpha f_{S_{i-1}}(e)$ for each $e \in O_i$ and hence

$$|O_i|\delta_i \geq \alpha \sum_{e \in O_i} f_{S_{i-1}}(e) \geq \alpha f_{S_{i-1}}(O_i) \geq \alpha f_{S_k}(O_i).$$

We use submodularity of f in both the middle inequality and the last inequality. For all $i \in \{1, 2, \ldots, k\}$, we have $|O_i| \leq p$, and thus $\delta_i \geq \frac{\alpha}{p} f_{S_k}(O_i)$. This finishes the proof.

Remark. The proof is simpler and more intuitive for the special case of p-extendible systems. We define a sequence of sets $O = T_0 \supseteq T_1 \supseteq \cdots \supseteq T_k = \emptyset$ such that, for $1 \leq i \leq k$, (i) $S_i \cup T_i \in \mathcal{I}$ and (ii) $S_i \cap T_i = \emptyset$. In other words, every element in T_i is a candidate for the greedy algorithm in step $i+1$. We partition the optimum solution as $O = O_1 \cup O_2 \cup \cdots \cup O_k$, where $O_i = T_{i-1} \setminus T_i$. This partition is defined so that we can charge the value of the optimum to marginal improvements made by the greedy algorithm.

We start with $T_0 = O$ and show how to construct T_i from T_{i-1}. If $e_i \in T_{i-1}$, we define $O_i = \{e_i\}$ and $T_i = T_{i-1} - e_i$. Otherwise, we let O_i be a smallest subset of T_{i-1} such that $(S_{i-1} \cup T_{i-1}) \setminus O_i + e_i$ is independent; since $\mathcal{I}$ is a p-extendible family, $|O_i| \leq p$. We let $T_i = T_{i-1} \setminus O_i$. It is easy to check that in both cases $S_i \cup T_i \in \mathcal{I}$ and $S_i \cap T_i = \emptyset$. Finally, $T_k = \emptyset$ since $S_k \cup T_k$ is independent, and the algorithm stops only when $\{e \mid S_k + e \in \mathcal{I}\} = \emptyset$. Since any of the elements in O_i (in fact, T_{i-1}) was a candidate for e_i in the greedy algorithm, $\delta_i \geq \alpha f_{S_{i-1}}(e)$ for each $e \in O_i$. Moreover, $|O_i| \leq p$, which implies $\delta_i \geq \frac{\alpha}{p} f_{S_{i-1}}(O_i)$. Note that the proof for p-systems is more subtle since we cannot use a local argument to show that $|O_i| \leq p$ for each i; we need a more global argument.

Acknowledgments. The last author would like to thank Uriel Feige, Mohammad Mahdian, and Vahab Mirrokni for inspiring discussions concerning the submodular welfare problem. We thank Julian Mestre for discussions that clarified the different notions of p-independence families. We thank the anonymous reviewers for their careful reading and suggestions.

REFERENCES

[1] A. Ageev and M. Sviridenko, *Pipage rounding: A new method of constructing algorithms with proven performance guarantee*, J. Combin. Optim., 8 (2004), pp. 307–328.

[2] N. Alon and J. H. Spencer, *The Probabilistic Method*, 2nd ed., Wiley Interscience, New York, 2000.

[3] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, *Maximizing a submodular set function subject to a matroid constraint*, in Proceedings of the 12th IPCO Conference, 2007, pp. 182–196.

[4] G. Calinescu, C. G. Fernandes, H. Karloff, and A. Zelikovski, *A new approximation algorithm for finding heavy planar subgraphs*, Algorithmica, 36 (2003), pp. 179–205.

[5] D. Chakrabarty and G. Goel, *On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and GAP*, in Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, 2008, pp. 687–696.

[6] C. Chekuri and S. Khanna, *A polynomial time approximation scheme for the multiple knapsack problem*, SIAM J. Comput., 35 (2006), pp. 713–728.

[7] C. Chekuri and A. Kumar, *Maximum coverage problem with group budget constraints and applications*, in Approximation, Randomization, and Combinatorial Optimization, Lecture Notes in Comput. Sci. 3122, Springer, Berlin, 2004, pp. 72–83.

[8] C. Chekuri and M. Pál, *A recursive greedy algorithm for walks in directed graphs*, in Proceedings of the 46th Annual Symposium on Foundations of Computer Science, 2005, pp. 245–253.

[9] G. Cornuejols, M. Fisher, and G. Nemhauser, *Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms*, Management Sci., 23 (1977), pp. 789–810.

[10] W. H. Cunningham, *Testing membership in matroid polyhedra*, J. Combin. Theory Ser. B, 36 (1984), pp. 161–188.

[11] S. Dobzinski and M. Schapira, *An improved approximation algorithm for combinatorial auctions with submodular bidders*, in Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2006, pp. 1064–1073.

[12] J. Edmonds, *Submodular functions, matroids, and certain polyhedra*, in Combinatorial Structures and Their Applications, Gordon and Breach, New York, 1970, pp. 69–87.

[13] U. Feige, *A threshold of* $\ln n$ *for approximating set cover*, J. ACM, 45 (1998), pp. 634–652.

[14] U. Feige, *On maximizing welfare when utility functions are subadditive*, in Proceedings of the 38th ACM Symposium on Theory of Computing, 2006, pp. 41–50.

[15] U. Feige, V. Mirrokni, and J. Vondrák, *Maximizing a non-monotone submodular function*, in Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science, 2007, pp. 461–471.

[16] U. Feige and J. Vondrák, *Approximation algorithms for allocation problems: Improving the factor of* $1 - 1/e$, in Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, pp. 667–676.

[17] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey, *An analysis of approximations for maximizing submodular set functions.* II., Math. Programming Stud., no. 8 (1978), pp. 73–87.

[18] L. Fleischer, S. Fujishige, and S. Iwata, *A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions*, J. ACM, 48 (2001), pp. 761–777.

[19] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko, *Tight approximation algorithms for maximum general assignment problems*, in Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2006, pp. 611–620.

[20] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan, *Dependent rounding and its applications to approximation algorithms*, J. ACM, 53 (2006), pp. 324–360.

[21] P. Goundan and A. Schulz, *Revisiting the Greedy Approach to Submodular Set Function Maximization*, manuscript, 2007.

[22] R. L. Graham, *Bounds for certain multiprocessor anomalies*, Bell System Tech. J., 45 (1966), pp. 1563–1581.

[23] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer, Berlin, 1988.

[24] E. Hazan, S. Safra, and O. Schwartz, *On the complexity of approximating k-set packing*, in Approximation, Randomization, and Combinatorial Optimization, Lecture Notes in Comput. Sci. 2764, Springer, Berlin, 2003, pp. 83–97.

[25] T. A. Jenkyns, *The efficacy of the "greedy" algorithm*, in Proceedings of the 7th Southeastern Conference on Combinatorics, Graph Theory and Computing, 1976, pp. 341–350.

[26] S. Khot, R. Lipton, E. Markakis, and A. Mehta, *Inapproximability results for combinatorial auctions with submodular utility functions*, Algorithmica, 52 (2008), pp. 3–18.

[27] S. Khuller, A. Moss, and J. Naor, *The budgeted maximum coverage problem*, Inform. Process. Lett., 70 (1999), pp. 39–45.

[28] B. Korte and D. Hausmann, *An analysis of the greedy heuristic for independence systems*, Ann. Discrete Math., 2 (1978), pp. 65–74.

[29] A. Kulik, H. Shachnai, and T. Tamir, *Maximizing submodular functions subject to multiple linear constraints*, in Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, 2009, pp. 545–554.

[30] J. Lee, V. Mirrokni, V. Nagarajan, and M. Sviridenko, *Non-monotone submodular maximization under matroid and knapsack constraints*, in Proceedings of the 41st ACM Symposium on Theory of Computing, 2009, pp. 323–332.

[31] J. Lee, M. Sviridenko, and J. Vondrák, *Submodular maximization over multiple matroids via generalized exchange properties*, in Approximation, Randomization, and Combinatorial Optimization, Lecture Notes in Comput. Sci. 5687, Springer, Berlin, 2009, pp. 244–257.

[32] B. Lehmann, D. J. Lehmann, and N. Nisan, *Combinatorial auctions with decreasing marginal utilities*, Games Econom. Behav., 55 (2006), pp. 270–296.

[33] J. Mestre, *Greedy in approximation algorithms*, in Proceedings of the 14th European Symposium on Algorithms, Springer, Berlin, 2006, pp. 528–539.

[34] J. Mestre, *private communication*, 2008.

[35] V. Mirrokni, M. Schapira, and J. Vondrák, *Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions*, in Proceedings of the ACM Conference on Electronic Commerce, 2008, pp. 70–77.

[36] G. L. Nemhauser and L. A. Wolsey, *Best algorithms for approximating the maximum of a submodular set function*, Math. Oper. Res., 3 (1978), pp. 177–188.

[37] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, *An analysis of approximations for maximizing submodular set functions.* I., Math. Programming, 14 (1978), pp. 265–294.

[38] A. Schrijver, *A combinatorial algorithm minimizing submodular functions in strongly polynomial time*, J. Combin. Theory Ser. B, 80 (2000), pp. 346–355.

[39] A. Schrijver, *Combinatorial Optimization—Polyhedra and Efficiency*, Springer, Berlin, 2003.

[40] A. Srinivasan, *Distributions on level-sets with applications to approximation algorithms*, in Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, 2001, pp. 588–597.

[41] M. Sviridenko, *A note on maximizing a submodular set function subject to knapsack constraint*, Oper. Res. Lett., 32 (2004), pp. 41–43.

[42] J. Vondrák, *Optimal approximation for the submodular welfare problem in the value oracle model*, in Proceedings of the 40th ACM Symposium on Theory of Computing, 2008, pp. 67–74.

[43] J. Vondrák, *Symmetry and approximability of submodular maximization problems*, in Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science, 2009, pp. 651–670.

[44] L. Wolsey, *An analysis of the greedy algorithm for the submodular set covering problem*, Combinatorica, 2 (1982), pp. 385–393.

[45] L. Wolsey, *Maximizing real-valued submodular functions: Primal and dual heuristics for location problems*, Math. Oper. Res., 7 (1982), pp. 410–425.