Maximizing a Monotone Submodular Function subject to a Matroid Constraint*

Gruia Calinescu† Chandra Chekuri‡ Martin Pál§ Jan Vondrák¶

September 15, 2008

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 [14] for this problem. For certain special cases, e.g. $\max_{|S| \leq k} f(S)$, 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) [28], and also for explicitly posed instances assuming $P \neq NP$ [10].

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 [1], and a *continuous greedy process* that might be of independent interest.

As a special case, our algorithm implies an optimal approximation for the Submodular Welfare Problem in the value oracle model [32]. 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.

Keywords: monotone submodular set function, matroid, social welfare, generalized assignment problem, approximation algorithm.

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

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

*This paper combines results and ideas from two extended abstracts [6] and [32], that had previously appeared in *Proc. of IPCO, 2007* and *Proc. of ACM STOC, 2008* respectively.

†Computer Science Dept., Illinois Institute of Technology, Chicago, IL. calinescu@iit.edu. Research partially supported by NSF grant CCF-0515088.

‡Dept. of Computer Science, University of Illinois, Urbana, IL 61801. chekuri@cs.uiuc.edu. This work was partly done while the author was at Lucent Bell Labs and is partly supported by a NSF grant CCF-0728782.

§Google Inc., 1440 Broadway, New York, NY 10018. mpal@google.com.

¶Dept. of Mathematics, Princeton University, Princeton, NJ 08544. jvondrak@math.princeton.edu.

for all $A, B \subseteq X$. We restrict attention to monotone (by which we mean non-decreasing) functions, that is $f(A) \leq f(B)$ for all $A \subseteq B$ and $f(\emptyset) = 0$. 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 $B \subseteq A$ implies 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). 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. 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, 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 $f : 2^X \rightarrow \mathcal{R}_+$ 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 $w : X \rightarrow \mathcal{R}_+$ 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 $A_1, A_2, \ldots, A_n \subset U$, we obtain several natural submodular functions on the set $X = \{1, 2, \ldots, n\}$. 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 non-negative weight function $w : U \rightarrow \mathcal{R}_+$, $f(S) = w(\cup_{i \in S} A_i)$. We obtain a multi-cover 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 non-negative 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 $w : X \rightarrow \mathcal{R}_+$, 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. 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 very simple matroid constraint that is of much importance in applications [7, 27, 3, 4, 15] is the *partition matroid*: X is partitioned into ℓ sets $X_1, X_2, \ldots, X_\ell$ with associated integers $k_1, k_2, \ldots, k_\ell$, and a set $A \subseteq X$ is independent iff $|A \cap X_i| \leq k_i$. 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 $\mathcal{M}_1 = (X, \mathcal{I}_1), \mathcal{M}_2 = (X, \mathcal{I}_2), \ldots, \mathcal{M}_p = (X, \mathcal{I}_p)$ on the same ground set X. That is, $\mathcal{I} = \{A \subseteq X \mid A = \cap_i A_i \text{ and } A_i \in \mathcal{I}_i,\ 1 \leq i \leq p\}$. A simple example is the family of hypergraph matchings in a p-partite graph ($p = 2$ 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 [19, 21]. Given an independence family $\mathcal{I}$, let $\mathcal{B}$ be the set of maximal independent sets in $\mathcal{I}$, that is $\mathcal{B} = \{A \in \mathcal{I} \mid \text{there is no } A' \in \mathcal{I} \text{ such that } A' \supset A\}$. Then $\mathcal{I}$ is a p-system if for all $Y \subseteq X$, $\frac{\max_{A \in \mathcal{B}, A \subseteq Y} |A|}{\min_{A \in \mathcal{B}, A \subseteq Y} |A|} \leq p$. p-systems properly generalize several simpler and easier to understand special cases including families obtained from the intersection of p matroids. We discuss some other special cases in Section A. The set of matchings in a general graph form a 2-system. Similarly the set of matchings in a hypergraph with edges of cardinality at most p form a p-system.

The Greedy Algorithm. A simple 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-independence constraint [19, 21]. For submodular functions, the greedy algorithm yields a ratio of $1/(p+1)$ [14][1]. These ratios for Greedy 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 [18].

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 [27, 28]. This special case already captures the max-k-cover problem (with $f(S)$ of the coverage type) for which it is shown in [10] that no $(1 - 1/e + \epsilon)$-approximation is possible for any constant $\epsilon > 0$, unless $P = NP$. Thus it is a 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 which gives with high probability a $(1 - 1/e)$-approximation 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.*

Our main tools are the *pipage rounding* technique of Ageev and Sviridenko [1], and a *continuous greedy process*. We give an overview of these techniques in Section 2.

Applications. As a consequence we obtain $(1 - 1/e)$-approximation algorithms for a number of optimization problems. An immediate application is the Submodular Welfare Problem which can be cast as a submodular maximization problem subject to a partition matroid [22]. In this problem, we are given n players and m items. Each player i has a 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 [22], while a $(1 - 1/e + \epsilon)$-approximation in the value oracle model, for any fixed $\epsilon > 0$, would imply $P = NP$ [20]. Improvements over the $1/2$-approximation were achieved only in special cases or using a stronger computation model [8, 12]. Our work implies the following optimal result, which first appeared in [32].

Theorem 1.2. *There is a randomized algorithm which achieves with high probability a $(1 - 1/e)$-approximation for the Submodular Welfare Problem, in the value oracle model.*

[1] We give a proof of this result in the appendix 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.

Another application of Theorem 1.1 is related to variants of the Generalized Assignment Problem (GAP). In GAP 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. Recently Fleischer et al. [15] gave a $(1-1/e)$-approximation for this problem, improving upon a previous 1/2-approximation [5]. We rederive the same ratio casting the problem as a special case of submodular function maximization. Moreover, our techniques allow us to obtain a simpler $(1-1/e-o(1))$-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.

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 satisfy the constraint $S \in \mathcal{I}$.*

We note that the approximation ratio for GAP has been improved to $1-1/e+\delta$ for a small $\delta > 0$ in [12] using the same LP as in [15]. However, the algorithm in [15] extends to even more general class of problems, the *Separable Assignment Problem* (SAP). For SAP, it is shown in [15] that it is NP-hard to obtain an approximation ratio of $1-1/e+\epsilon$ for any constant $\epsilon > 0$. Our framework also extends to the Separable Assignment Problem, and hence $1-1/e$ is the best approximation factor one can achieve with this approach. We discuss this further in Section 4.2.

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.

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 non-increasing as a function of A for every fixed i.

Smooth submodular functions. As a continuous analogy, Wolsey [34] defines submodularity for a function $F : [0,1]^X \to \mathcal{R}_+$ as follows:

$$F(x \vee y) + F(x \wedge y) \leq F(x) + F(y) \tag{1}$$

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 $x \leq y$ coordinate-wise. 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*

- *$F \in C_2([0,1]^X)$, 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* $\nabla F = (\frac{\partial F}{\partial y_1}, \ldots, \frac{\partial F}{\partial y_n})$ 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 non-increasing with respect to y_i. It can be seen that this implies (1). Also, it means that a smooth submodular function is concave along any non-negative direction vector; however, it is not necessarily concave in all directions.

Extension by expectation. 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 [6]: 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, 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

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

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

Matroids. A matroid is a pair $\mathcal{M} = (X, \mathcal{I})$ where $\mathcal{I} \subseteq 2^X$ and

1. $\forall B \in \mathcal{I}, A \subset B \Rightarrow A \in \mathcal{I}$.
2. $\forall A, B \in \mathcal{I}; |A| < |B| \Rightarrow \exists x \in B \setminus A; A + x \in \mathcal{I}$.

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

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

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, $0 \leq x \leq y$, $y \in P \Rightarrow x \in P$. 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 [9], an equivalent description is

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

where $r_M(S) = \max\{|I| : I \subseteq S \ \& \ I \in \mathcal{I}\}$ is the rank function of $\mathcal{M}$. 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 $B(\mathcal{M}) = \{y \in P(\mathcal{M}) \mid y(X) = r_{\mathcal{M}}(X)\}$. 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^* where 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 [29] for more details on matroids and polyhedral aspects.

2.2 Our approach

We consider the problem

$$\max\{f(S) : S \in \mathcal{I}\}, \tag{2}$$

where $f : 2^X \rightarrow \mathcal{R}_+$ 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 [6], the discrete problem (2) can be replaced by a linear programming problem,

$$\max\{\sum_i c_i x_i : x \in P\} \tag{3}$$

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 (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 [6] 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) by a linear program. Nonetheless, we have a generic way of replacing a discrete function $f(S)$ by a continuous function: the extension $F(y) = \mathbf{E}[f(\hat{y})]$ described in Section 2.1. Using this extension, we obtain a *non-linear optimization problem*:

$$\max\{F(y) : y \in P(\mathcal{M})\} \tag{4}$$

where $P(\mathcal{M})$ is the matroid polytope of $\mathcal{M}$. If $F(y)$ were a concave function, we would be in a good shape to deal with this continuous problem, using generic non-linear 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. As we discussed in Section 2.1, $F(y)$ is a smooth submodular function, and it particular it is a *harmonic function* which means that it is concave in certain directions while convex in others. This will be actually 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 [6] in the case of sums of weighted rank functions. The first stage has been discovered more recently; it first appeared in [32] 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 to move 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:

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

where $v(y)$ is defined as

$$v(y) = \mathrm{argmax}_{v \in P}(v \cdot \nabla F(y)).$$

Claim. $y(1) \in P$ and $F(y(1)) \geq (1 - 1/e)OPT$.
First of all, the trajectory for $t \in [0, 1]$ is contained in P, since

$$y(t) = \int_0^t v(y(\tau))d\tau$$

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 a 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 non-increasing. 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 $v(y) \in P$ maximizes $v \cdot \nabla F(y)$ over all vectors $v \in P$, we get

$$v(y) \cdot \nabla F(y) \geq v^* \cdot \nabla F(y) \geq F(y+v^*) - F(y) \geq OPT - F(y). \tag{5}$$

Now let us return to our continuous process and analyze $F(y(t))$. By the chain rule and using (5), we get

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

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 $v(y)$ 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 $v(y)$ is a vertex of P and furthermore, since ∇F is a nonnegative vector, that this vertex corresponds to a basis of $\mathcal{M}$. Hence, without loss of generality $v(y)$ is contained in $B(\mathcal{M})$, the basis polytope, and it can be found by the greedy algorithm for maximum-weight basis in a matroid.

Remark. Wolsey's continuous greedy algorithm [34] can be viewed as a greedy process guided by $v(y) = \mathbf{e}_j$, 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 $F(y) = \mathbf{E}[f(\hat{y})]$, it can be seen that y_j will increase up to its maximal possible value 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 non-linear programs that they called "pipage rounding". Subsequently, Srinivasan [31] and Gandhi et al. [23] 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, 31, 23] show that randomized rounding can be applied in a certain controlled way to guide a solution that respects certain class 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 a 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 basis polytope has a particular structure: it is known for example that the edges of the basis polytope are all of the form $(\mathbf{1}_I, \mathbf{1}_J)$, where J is a basis obtained from I by swapping one element for another [29]. 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 $F_{ij}^y(t) = F(y + t(\mathbf{e}_i - \mathbf{e}_j))$ is convex.
We prove this by considering the properties of $F(y) = \mathbf{E}[f(\hat{y})]$. By differentiating $F_{ij}^y(t)$ twice, we obtain

$$\frac{d^2 F_{ij}^y}{dt^2} = \sum_{\alpha,\beta} \frac{\partial^2 F}{\partial y_\alpha \partial y_\beta} (\mathbf{e}_i - \mathbf{e}_j)_\alpha (\mathbf{e}_i - \mathbf{e}_j)_\beta = \frac{\partial^2 F}{\partial {y_i}^2} - 2\frac{\partial^2 F}{\partial y_i \partial y_j} + \frac{\partial^2 F}{\partial {y_j}^2}.$$

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, $\frac{d^2 F_{ij}^y}{dt^2} \geq 0$ and the function F_{ij}^y is convex.

The convexity of $F_{ij}^y(t) = F(y + t(\mathbf{e}_i - \mathbf{e}_j))$ 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. A suitable choice of probabilities will ensure that we never lose in expectation. We defer further details to Section 3.2.

3 The algorithm for submodular maximization subject to a matroid contraint

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 $v(y) = \mathrm{argmax}_{v \in P(\mathcal{M})}(v \cdot \nabla F(y))$. 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 $\delta = 1/n^2$ where $n = |X|$. Start with $t = 0$ and $y(0) = \mathbf{0}$.
2. Let $R(t)$ contain each j independently with probability $y_j(t)$.
   For each $j \in X$, estimate
   $$\omega_j(t) = \mathbf{E}[f_{R(t)}(j)].$$
   by taking the average of n^5 independent samples.
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 non-negative weights, we can assume that each $I(t)$ is a basis 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 Eq. (5)).

Lemma 3.1. *Let $OPT = \max_{S \in \mathcal{I}} f(S)$. 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, $OPT \leq \mathbf{E}[f(R) + \sum_{j \in O} f_R(j)] = F(y) + \sum_{j \in O} \mathbf{E}[f_R(j)] \leq F(y) + \max_{I \in \mathcal{I}} \sum_{j \in I} \mathbf{E}[f_R(j)]$. □

Given this lemma, we prove the main result of this section.

Lemma 3.2. *The fractional solution y found by the Continuous Greedy Algorithm satisfies with high probability*

$$F(y) = \mathbf{E}[f(\hat{y})] \geq \left(1 - \frac{1}{e} - o(1)\right) \cdot OPT.$$

Proof. We start with $F(y(0)) = 0$. Our goal is to estimate how much $F(y(t))$ increases during one step 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 $\Delta_j(t) = y_j(t+\delta) - y_j(t)$. I.e., $\Delta(t) = y(t + \delta) - y(t) = \delta \cdot \mathbf{1}_{I(t)}$ and $D(t)$ is a random subset of $I(t)$ where each element appears independently with probability δ. It can be seen easily 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 $1 - (1 - y_j(t))(1 - \Delta_j(t))$.

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 $D(t)$ is 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

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

Recall that $I(t)$ is an independent set maximizing $\sum_{j \in I} \omega_j(t)$ where $\omega_j(t)$ are our estimates of $\mathbf{E}[f_{R(t)}(j)]$. By standard Chernoff bounds, the probability that the error in any estimate is more than OPT/n^2 is exponentially small in n (note that $OPT \geq \max_{R,j} f_R(j)$). Hence, w.h.p. we incur an error of at most OPT/n in our computation of the maximum-weight independent set. Then we can write

$$\begin{aligned} F(y(t+\delta)) - F(y(t)) &\geq \delta(1-n\delta) \left(\max_{I \in \mathcal{I}} \sum_{j \in I} \mathbf{E}[f_{R(t)}(j)] - OPT/n \right) \\ &\geq \delta(1-1/n)(OPT - F(y(t)) - OPT/n) \\ &\geq \delta(\tilde{OPT} - F(y(t))) \end{aligned}$$

using Lemma 3.1, $\delta = 1/n^2$ and setting $\tilde{OPT} = (1-2/n)OPT$. 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, $F(y(1)) \geq (1-1/e)\tilde{OPT} \geq (1-1/e-o(1))OPT$. □

Remark. By a more careful analysis, we can eliminate the error term and achieve a clean approximation factor of $1-1/e$. We can argue as follows: 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 $R(t) \cup \tilde{D}(t) \subseteq R(t+\delta)$. (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 probability does not exceed 2δ.) $\tilde{D}(t)$ is a random subset of $I(t)$ and the size of $I(t)$ is $d = \mathrm{rank}(\mathcal{M})$. We can repeat the analysis with $\tilde{D}(t)$ and we get

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

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

Denote $\omega^*(t) = \max_j \mathbf{E}[f_{R(t)}(j)]$. By Lemma 3.1 and submodularity, we know that at any time, $\omega^*(t) \geq \frac{1}{d}(OPT - F(y(t)))$ where $d = \mathrm{rank}(\mathcal{M}) = |I(t)|$. Also, if $j^*(t)$ is the element achieving $\omega^*(t)$, we know that $y_{j^*(t)}(t)$ cannot be zero all the time. Even focusing only on the increments corresponding to $j^*(t)$, at most half of them can be to variables where $y_{j^*(t)}(t) < \frac{1}{2n}$, otherwise the total contribution to these variables would be more than $1/2$ and their sum less than $1/2$ - a contradiction. Let's 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 estimate the marginal values $\mathbf{E}[f_{R(t)}(j)]$ within $\delta\, OPT$, so that our

computation of the maximum-weight basis $I(t)$ is accurate within $d\delta\, OPT$. In good steps, the improved analysis gives

$$\begin{aligned} F(y(t+\delta)) - F(y(t)) &\geq \delta(1-2d\delta)\left(\sum_{j\in I(t)} \mathbf{E}[f_{R(t)}(j)] + y_{j^*}(t)\mathbf{E}[f_{R(t)}(j^*)]\right) \\ &\geq \delta(1-2d\delta)\left(1+\frac{1}{2dn}\right)(OPT - F(y(t)) - d\delta OPT) \\ &\geq \delta(1-3d\delta)\left(1+\frac{1}{2dn}\right)(OPT - F(y(t))). \end{aligned}$$

By taking $\delta << \frac{1}{d^2 n}$, we make this expression bigger than $\delta(1+3d\delta)(OPT - F(y(t)))$. Then, we can conclude that in good steps, we have

$$OPT - F(y(t+\delta)) \leq (1-\delta(1+3d\delta))(OPT - F(y(t))),$$

while in bad steps we get by the standard analysis

$$OPT - F(y(t+\delta)) \leq (1-\delta(1-3d\delta))(OPT - F(y(t))).$$

Overall, we get

$$OPT - F(y(1)) \leq (1-\delta+3d\delta^2))^{\frac{1}{2\delta}}(1-\delta-3d\delta^2)^{\frac{1}{2\delta}} OPT \leq (1-\delta)^{1/\delta} OPT$$

which means $F(y(1)) \geq (1-(1-\delta)^{1/\delta})OPT \geq (1-1/e)OPT$. Finally, observe that even $\delta = 1/d^2$ is sufficient to obtain $F(y(1)) \geq (1-1/e-O(1/d))OPT$.

3.2 The pipage rounding algorithm

We start with a point y^* in the basis 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.

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 easily from the submodularity of the rank function $r_{\mathcal{M}}$.

Proposition 3.3. *If A and 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 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 $y_{ij}(t) = y + t(\mathbf{e}_i - \mathbf{e}_j)$. i.e. 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 cannot decrease for both $t > 0$ and $t < 0$. Therefore, it is possible to modify the fractional variables y_i, y_j by increasing one of them and decreasing the other. We do this until we hit another constraint of the matroid polytope. The subroutine **HitConstraint** performs this task. It will be useful to make the procedure oblivious, independent of the function $F(y)$, and hence we will randomize this step to achieve a good value in expectation.

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 Prop. 3.3).

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 $\{\delta \leftarrow y_j,\ A \leftarrow \{j\}\}$;
$y_i \leftarrow y_i + \delta,\ y_j \leftarrow y_j - \delta$;
Return (y, A).

Algorithm **PipageRound**(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.

Lemma 3.4. *Given $y \in B(\mathcal{M})$,* **PipageRound***(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 don't 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 any case, we get a new tight set T at the end of the inner loop, due to Proposition 3.3.

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.

The non-trivial step in the algorithm is the minimization of $r_{\mathcal{M}}(S) - y(S)$ 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 [16, 30].

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 choose randomly between $F(y^+) = F^y_{ij}(\epsilon^+)$ and $F(y^-) = F^y_{ij}(\epsilon^-)$ where $\epsilon^- < 0 < \epsilon^+$. Let y' denote the (random) point obtained after one step. By convexity of F^y_{ij}, using $p\epsilon^- + (1-p)\epsilon^+ = 0$, 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 discrete 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 many applications.

Definition 3.5. *For a ground set partitioned into $X = X_1 \cup X_2 \cup \ldots \cup X_k$, 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, we have $P(\mathcal{M}) = \{y \in \mathcal{R}^X_+ : \forall i; \sum_{j \in X_i} y_j \leq 1\}$. 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_j (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 justifies that this works.

Lemma 3.6. *Let $X = X_1 \cup \ldots \cup X_k$, let $f : 2^X \to \mathcal{R}_+$ be a monotone submodular function, and $y \in B(\mathcal{M})$ where $\mathcal{M}$ is a simple partition matroid on $X = X_1 \cup \ldots \cup X_k$. 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*

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

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 $T' = T \cap (X_1 \cup \ldots \cup X_{k-1})$, $T'' = T \cap X_k$ and $R' = R \cap (X_1 \cup \ldots \cup X_{k-1})$, $R'' = R \cap X_k$. 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 $X_1 \cup \ldots \cup X_{k-1}$. □

Therefore, we can replace pipage rounding for simple partition matroids by the following simple procedure.

Simple rounding.

- Given $y \in \mathcal{R}_+^X$ such that $\forall i; \sum_{j \in X_i} y_j = 1$, 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 problem and known results.

Social welfare maximization. *Given a set X of m items, and n players, each of which has 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 [22, 20, 8, 11, 12, 26]. Before studying its complexity status, one needs to specify how the algorithm accesses the input of 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 oracle most commonly considered are:

- *Value oracle:* returns the value of $w_i(S)$ for a given player i and $S \subseteq X$.
- *Demand oracle:* 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, the problem is NP-hard to approximate within a factor of $m^{1/2-\epsilon}$ for any $\epsilon > 0$, even in the demand oracle model. 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.

In the value oracle model, the greedy algorithm achieves a 1/2-approximation for all submodular functions [22]. On the hardness side, it has been shown that a $(1-1/e+\epsilon)$-approximation for any fixed $\epsilon > 0$ would imply $P = NP$ [20]. This hardness result holds in particular for submodular functions induced by explicitly given set coverage systems. It is also known that a $(1-1/e+\epsilon)$-approximation would require exponentially many value queries, regardless of $P = NP$ [26]. It was an open problem whether a $(1-1/e)$-approximation can be achieved for all submodular functions.

In the demand oracle model, on the other hand, a $(1-1/e)$-approximation was presented in [8]. This algorithm has been subsequently generalized to all fractionally subadditive functions [11] and improved to $1-1/e+\delta, \delta > 0$, for submodular functions [12].

Our result. The Submodular Welfare Problem can be seen as a special case of SUB-M and our algorithm can be used to give a randomized $(1-1/e)$-approximation, thus resolving the status of the problem in the value oracle model (this result first appeared in [32]). We briefly describe the reduction to SUB-M (see also [22, 17]).

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 $\pi_i : 2^X \rightarrow 2^A$,

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

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

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

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 $A = S_1 \cup S_2 \cup \ldots \cup S_n$ 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 $X_j = [n] \times \{j\}$. 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 Submodular Welfare, 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 the all the clones of item j. By Lemma 3.6, 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.

The Continuous Greedy Algorithm for Submodular Welfare.

1. Let $\delta = 1/(mn)^2$. 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 $(mn)^5$ independent samples.
3. For each j, let $i_j(t) = \text{argmax}_i\, \omega_{ij}(t)$ 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 Assignment Problem

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 non-negative 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 1[2], and the total value of all items is maximized. In [15], a $(1 - 1/e)$-approximation algorithm for GAP is presented. Their algorithm is based on solving an exponential sized linear program. The separation oracle for this LP is the knapsack problem and one obtains an arbitrarily close approximation to it by using an FPTAS for knapsack. In [12], it is shown that the LP 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 [2].

Our techniques yield a simple $(1 - 1/e - o(1))$-approximation algorithm for GAP, which does not rely on the ellipsoid method or any other LP solving algorithms. Although this is known to be a suboptimal approximation factor, the new algorithm is much more practical than the $(1 - 1/e + \delta)$-approximation of [12]. Our algorithm also generalizes to more general 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's placed. The goal is to choose disjoint feasible sets $S_j \in \mathcal{F}_j$ so as to maximize $\sum_{j=1}^{n} \sum_{i \in S_j} v_{ij}$.

Reduction to a matroid constraint. Let us review the reduction from [6]. We define $X = \{(j, S) \mid 1 \leq j \leq n, S \in \mathcal{F}_j\}$ and a function $f : 2^X \to \mathcal{R}_+$,
$$f(\mathcal{S}) = \sum_i \max_j \{v_{ij} : \exists (j, S) \in \mathcal{S}, i \in S\}.$$

[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.

It is clear that f is monotone and submodular. 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) for each i. Such a set $\mathcal{S}$ corresponds to an assignment of set S to bin j for each $(j, S) \in \mathcal{S}$. This is equivalent to SAP: although the bins can be assigned overlapping sets in this formulation, we only count the value of the most valuable assignment for each item.

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 $v(y) \in P(\mathcal{M})$ such that $v(y) \cdot \nabla F(y) \geq \alpha \max_{v \in P(\mathcal{M})}(v \cdot \nabla F(y))$. By Eq. (5), $v(y) \cdot \nabla F(y) \geq \alpha(OPT - F(y))$. 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. First of all, the number of steps in the continuous greedy algorithm depends on the discretization parameter δ. Following the remark at the end of Section 3.1, we choose δ polynomially small in the rank of $\mathcal{M}$, which is n here. 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

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

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 the Separable Assignment Problem. We summarize the algorithm here.

The Continuous Greedy Algorithm for SAP/GAP.

1. Let $\delta = 1/n^2$. 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,
$$\omega_{ij}(t) = \mathbf{E}_{\mathcal{R}(t)}[\max\{v_{ij} - \max\{v_{ij'} : \exists(j', S) \in \mathcal{R}(t); i \in S\}, 0\}]$$
by taking the average of $(mn)^5$ independent samples.

3. For each j, find an α-approximate solution $S_j^*(t)$ to the problem $\max\{\sum_{i\in S} \omega_{ij}(t) : S \in \mathcal{F}_j\}$. 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 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 the problem $\max\{\sum_{i\in S} \omega_{ij} : S \in \mathcal{F}_j\}$, 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 Separable Assignment Problem with families of feasible sets $\mathcal{F}_j$.*

This beats both the factor $\alpha(1 - 1/e)$ obtained by using the Configuration LP [15] and the factor $\alpha/(1 + \alpha)$ obtained by a simple greedy algorithm [6, 17].

The Generalized Assignment Problem Special cases of the Separable Assignment Problem 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 the Generalized Assignment Problem (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 the Generalized Assignment Problem.

Matroid constraint on the bins. Consider GAP with the additional restriction that at most k of the given m bins be used in assigning items to. 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 $|\mathcal{S} \cap \{(j, S) \mid S \in \mathcal{F}_j\}| = 1$ for each bin j. Now a set $\mathcal{S}$ is independent iff $|\mathcal{S} \cap \{(j, S) \mid S \in \mathcal{F}_j\}| = 1$ 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 considerable 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) $|\mathcal{S} \cap \{(j, S) \mid S \in \mathcal{F}_j\}| = 1$ for each bin j, and (ii) $B_{\mathcal{S}} \in \mathcal{I}$ where $B_{\mathcal{S}} = \{j \mid (j, S) \in \mathcal{S}\}$. Once again it is easy to check the $\mathcal{M}$ is a matroid. The algorithm can be implemented in a way similar to the algorithm for SAP.

5 Conclusions

Our algorithm uses randomization in an intrinsic way since it works with the extension $F(y) = \mathbf{E}[f(\hat{y})]$. It is an interesting open problem as to 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, one can use an explicit extension function $F'(y)$ that can evaluated deterministically and use this instead of $\mathbf{E}[f(\hat{y})]$ — 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 [1, 6]. Therefore, in some special cases, deterministic $(1 - 1/e)$ approximations can be achieved.

One could consider *non-monotone* submodular functions with the requirement that they are non-negative. However, for such functions, even the unconstrained problem $\max_{S \subseteq N} f(S)$ is NP-hard and APX-hard to approximate; the Max-Cut problem is a special case. Feige, Mirrokni and Vondrák [13] give constant factor approximations for the unconstrained problem. For the matroid constraint problem, the pipage rounding framework is applicable even to non-monotone functions (as already shown in [1]). For non-monotone functions, the problem we need to consider is $\max_{S \in \mathcal{B}} f(S)$ where $\mathcal{B}$ is the set of bases of $\mathcal{M}$. The convexity of F^y_{ij} holds even for non-monotone functions. However, the continuous greedy algorithm for maximizing F requires monotonicity. For specific special cases, one may be able to choose an extension and then apply pipage rounding. For example, in [1], Max-Cut with given sizes on the partitions and related problems are addressed in this fashion. It would be interesting to explore a more general framework for all non-monotone non-negative submodular functions.

Pipage rounding [1] and dependent randomized rounding [31, 23] are based on rounding fractional solutions to the assignment problem into integer solutions while maintaining the quality of a solution that is a function of the variables on the edges of the underlying bipartite graph. A number of applications are given in [1, 31, 23]. This paper shows that submodularity and uncrossing properties of solutions to matroids 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.

Finally, can we improve the $1/(p+1)$ bound given by the greedy algorithm for maximizing a monotone submodular function subject to the intersection of p matroids? 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.

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.

References

[1] A. Ageev and M. Sviridenko. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. *J. of Combinatorial Optimization*, 8:307–328, 2004.

[2] D. Chakrabarty and G. Goel. On the Approximability of Budgeted Allocations and Improved Lower Bounds for Submodular Welfare Maximization and GAP. *Proc. of IEEE FOCS*, 2008. To appear.

[3] C. Chekuri and A. Kumar. Maximum coverage problem with group budget constraints and applications. *Proc. of APPROX*, Springer LNCS, 72–83, 2004.

[4] C. Chekuri and M. Pál. A recursive greedy algorithm for walks in directed graphs. *Proc. of IEEE FOCS*, 245–253, 2005.

[5] C. Chekuri and S. Khanna. A PTAS for the multiple knapsack problem. *SIAM J. on Computing*, 35(3):713–728, 2004.

[6] G. Calinescu, C. Chekuri, M. Pál and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint. *Proc. of* 12^{th} *IPCO*, 182–196, 2007.

[7] G. Cornuejols, M. Fisher and G. Nemhauser. Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. *Management Science*, 23: 789–810, 1977.

[8] Shahar Dobzinski and Michael Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. *Proc. of ACM-SIAM SODA*, 1064–1073, 2006.

[9] J. Edmonds. Matroids, submodular functions and certain polyhedra. Combinatorial Structures and Their Applications, 69–87, 1970.

[10] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[11] U. Feige. On maximizing welfare when utility functions are subadditive. *Proc. of ACM STOC*, 41–50, 2006.

[12] U. Feige and J. Vondrák. Approximation algorithms for allocation problems: Improving the Factor of $1 - 1/e$. *Proc. of IEEE FOCS*, 667–676, 2006.

[13] U. Feige, V. Mirrokni and J. Vondrák. Maximizing a non-monotone submodular function. *Proc. of IEEE FOCS*, 461–471, 2007.

[14] M. L. Fisher, G. L. Nemhauser and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions - II. *Math. Prog. Study*, 8:73–87, 1978.

[15] L. Fleischer, M.X. Goemans, V.S. Mirrokni and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems. *Proc. of ACM-SIAM SODA*, 611–620, 2006.

[16] L. Fleischer, S. Fujishige and S. Iwata. A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. *Journal of the ACM* 48:4, 761–777, 2001.

[17] P. Goundan and A. Schulz. Revisiting the Greedy Approach to Submodular Set Function Maximization. Manuscript, July 2007.

[18] E. Hazan, S. Safra and O. Schwartz. On the complexity of approximating k-set packing. *Proc. of APPROX*, Springer LNCS, 83–97, 2003.

[19] T. A. Jenkyns. The efficiency of the "greedy" algorithm. *Proc. of 7th South Eastern Conference on Combinatorics, Graph Theory and Computing*, 341–350, 1976.

[20] S. Khot, R. Lipton, E. Markakis and A. Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. *Algorithmica*, 52(1):3–18, 2008.

[21] B. Korte and D. Hausmann. An analysis of the greedy heuristic for independence systems. *Annals of Discrete Math.*, 2:65–74, 1978.

[22] B. Lehmann, D. J. Lehmann and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior* 55:270–296, 2006.

[23] R. Gandhi, S. Khuller, S. Parthasarathy and A. Srinivasan. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM*, 53(3):324–360, 2006.

[24] J. Mestre. Greedy in Approximation Algorithms. *Proc. of ESA*, 2006.

[25] J. Mestre. Personal communication. March, 2008.

[26] V. Mirrokni, M. Schapira and J. Vondrák. Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions. *Proc. of ACM EC*, 2008.

[27] G. L. Nemhauser, L. A. Wolsey and M. L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Math. Prog.*, 14:265–294, 1978.

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

[29] A. Schrijver. Combinatorial optimization - polyhedra and efficiency. Springer, 2003.

[30] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B* 80, 346–355, 2000.

[31] A. Srinivasan. Distributions on level-sets with applications to approximation algorithms. *Proc. of IEEE FOCS*, 588–597, 2001.

[32] J. Vondrák. Optimal approximation for the Submodular Welfare Problem in the value oracle model. *Proc. of ACM STOC*, 67–74, 2008.

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

[34] L. Wolsey. Maximizing real-valued submodular functions: Primal and dual heuristics for location Problems. *Math. of Operations Research*, 7:410–425, 1982.

A Analysis of Greedy for p-independent families

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 $f : 2^X \rightarrow \mathcal{R}^+$ is a monotone submodular set function. Fisher, Nemhauser and Wolsey [14] showed that the natural greedy algorithm has a tight approximation ratio of $1/(p+1)$. In [14], 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 [19] 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 easily 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 [17] have in independent work adapted the proof from [14] for intersection of p matroids to the approximate setting. However, both the proofs use an LP relaxation and differ from that of Jenkyns [19] whose scheme we follow.

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

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

Special cases of p-systems: We mention three special cases 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 [19][3]. A family $\mathcal{I}$ is a p-circuit-bounded family if for each $A \in \mathcal{I}$ and $e \in X - A$, $A + e$ has at most p circuits. Here, circuit is a minimally dependent set.
- p-extendible families defined in [24]. 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 $Z \subseteq B - A$ such that $|Z| \leq p$ and $B - Z + e \in \mathcal{I}$.

It can be shown that $\mathcal{I}$ is the intersecton 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 [25]. We mention that although p-systems are more general than p-extendible families, the latter seem to appear more naturally in applications.

Analysis of Greedy: We first define the greedy algorithm formally.

Algorithm **Greedy:**
$S \leftarrow \emptyset$, $A \leftarrow \emptyset$
repeat
  $A \leftarrow \{e \mid S \cup \{e\} \in \mathcal{I}\}$
  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 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 Greedy and for $i \geq 0$, we let S_i denote the set $\{e_1, e_2, \ldots, e_i\}$. Let $\delta_i = f(S_i) - f(S_{i-1})$ be the improvement in the solution value when e_i is added. Note that $f(S_k) = \sum_i \delta_i$. Since f is submodular, we observe that $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_k$. We now prove that Greedy yields a $1/(p+1)$ approximation. Fix some optimum solution O. We show the existence of a partition $O_1, O_2, \ldots, O_k$ of O with the following two properties:

[3] In [19] no name is suggested for these families.

- for $1 \leq i \leq k$, $p_1 + p_2 + \ldots + p_i \leq i \cdot p$ where $p_i = |O_i|$, and
- for $1 \leq i \leq k$, $p_i \delta_i \geq f_{S_k}(O_i)$.

Assuming that we have a partition of O with the above properties, we prove the desired bound on the performance of Greedy. First we need a simple claim.

Claim A.1. $p \sum_i \delta_i \geq \sum_i p_i \delta_i$.

Since $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_k$, and $\sum_{1 \leq j \leq i} p_i \leq i \cdot p$ for each i, the maximum attainable value for $\sum_i p_i \delta_i$ is to set $p_i = p$ for each i. See [19] for a more formal proof of a similar claim.

We thus have,

$$p \sum_i \delta_i \geq \sum_i p_i \delta_i \geq \sum_i f_{S_k}(O_i) \geq f_{S_k}(O) \geq f(O) - f(S_k).$$

We use submodularity in the third inequality above. Since $\sum_i \delta_i = f(S_k)$, the above inequality implies that $(p+1)f(S_k) \geq f(O)$ and hence $f(S_k) \geq f(O)/(p+1)$.

We now prove the existence of the desired partition of O. Define sets $A_0, \ldots, A_k$ as follows. $A_i = \{e \in O - S_i \mid S_i + e \in \mathcal{I}\}$. Note that $A_0 = O$, $A_i \subseteq A_{i-1}$ for $1 \leq i \leq k$, and $A_k = \emptyset$; the last fact is true since the Greedy algorithm stops only when no more elements can be added to its current set. We define $O_i = A_{i-1} \setminus A_i$. It follows that $O_1, O_2, \ldots, O_k$ is a partition of O. We claim that S_i is a basis for $S_i \cup O_i$. To see this, let $e \in O_i$; Either $e \in S_i$ or $e \in A_{i-1}$ and $S_i \cup \{e\} \notin \mathcal{I}$. In fact this argument shows that S_i is a basis for the set $X_i = O_1 \cup O_2 \cup \ldots \cup O_i \cup S_i$. Since $\mathcal{I}$ is a p-system, we have $|O_1 \cup O_2 \cup \ldots \cup O_i|/|S_i| \leq p$; note that $O_1 \cup O_2 \cup \ldots \cup O_i$ is independent since it is a subset of O. Since $|S_i| = j$ and $|O_1 \cup \ldots \cup O_i| = |O_1| + \ldots + |O_i|$, we get the inequality that $p_1 + \ldots + p_i \leq i \cdot p$. Now we argue that $\delta_i \geq f_{S_k}(O_i)/p_i$. Since $O_i \subseteq A_{i-1}$, each $e \in O_i$ was available as a candidate for Greedy when augmenting S_{i-1}. From the choice of Greedy it follows that $\delta_i \geq f_{S_{i-1}}(e)$ for each $e \in O_i$ and hence

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

We use submodularity of f in the last two inequalities. This finishes the proof.

Remark. The proof is simpler and more intuitive for the special case of p-extendible systems. We do not need Claim A.1. Instead we inductively define a sequence of sets $O = T_0 \supseteq T_1 \supseteq \ldots \supseteq T_k = S_k$ such that for $1 \leq i \leq k$, $S_i \subseteq T_i$ and $T_i \in \mathcal{I}$. We let $O_i = T_{i-1} - T_i$ which implies that $O_1, \ldots, O_k$ partition $O - S_k$. The set T_i is defined from T_{i-1} as the follows: since $S_{i-1} \subseteq T_{i-1}$ and $\mathcal{I}$ is a p-extendible family, there is a set O_i of at most p elements in $T_{i-1} - S_{i-1}$ such that $T_i = T_{i-1} - O_i + e_i$ is independent. Since any of the elements in O_i was a candidate for e_i in the greedy step, $\delta_i \geq \frac{1}{p} f_{S_{i-1}}(O_i)$. This leads to the desired bound.

Greedy with an approximate oracle for f: In some settings, the greedy step in picking the element with the largest marginal value can only be implemented in an approximate way. Suppose we are only guaranteed 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 $\alpha \leq 1$. The above analysis can be adapted easily to show that the bound on the performance now becomes $\alpha/(p+\alpha)$. We sketch the argument. Claim A.1 relies on the fact that $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_k$. This holds true for an exact oracle for f but may not hold true for an

approximate oracle in certain situations. However, one can modify the greedy algorithm slightly to have this property as follows; suppose $\delta_i > \delta_{i-1}$ at some stage. Let j be the smallest index such that $\delta_j > \delta_i$. We can rewind the greedy algorithm to iteration $j+1$ and pick e_i in that iteration instead of e_{j+1} thas was previously picked by the approximate oracle. This can only improve the solution. We can ensure polynomial running time by not rewinding unless the improvement is substantial. In the second part of the argument for an exact oracle, we had the inequality $p_i\delta_i \geq f_{S_{i-1}}(O_i)$. With an α-approximate oracle this changes to $p_i\delta_i \geq \alpha f_{S_{i-1}}(O_i)$. Therefore,

$$p\sum_i \delta_i \geq \sum_i p_i\delta_i \geq \sum_i \alpha f_{S_k}(O_i) \geq \alpha f_{S_k}(O) \geq \alpha(f(O) - f(S_k)),$$

which implies that $f(S_k) \geq \frac{\alpha}{p+\alpha} f(O)$.

Remark. For p-extendible systems, the Greedy algorithm (without any need for backtracking as above for p-systems) gives a bound of $\frac{\alpha}{p+\alpha}$ with an α-approximate oracle. The only change in the proof for p-extendible systems with an exact oracle (see the previous remark) is to replace the inequality $p\delta_i \geq f_{S_{i-1}}(O_i)$ by $p\delta_i \geq \alpha f_{S_{i-1}}(O_i)$.