Optimal Approximation for the Submodular Welfare Problem in the Value Oracle Model

Jan Vondrák
Department of Mathematics
Princeton University
Princeton, NJ 08544, USA
jvondrak@gmail.com

ABSTRACT

In the Submodular Welfare Problem, m items are to be distributed among n players with utility functions $w_i : 2^{[m]} \to \mathbb{R}_+$. The utility functions are assumed to be monotone and submodular. Assuming that player i receives a set of items S_i, we wish to maximize the total utility $\sum_{i=1}^{n} w_i(S_i)$. In this paper, we work in the *value oracle model* where the only access to the utility functions is through a black box returning $w_i(S)$ for a given set S. Submodular Welfare is in fact a special case of the more general problem of *submodular maximization subject to a matroid constraint*: $\max\{f(S) : S \in \mathcal{I}\}$, where f is monotone submodular and $\mathcal{I}$ is the collection of independent sets in some matroid.

For both problems, a greedy algorithm is known to yield a 1/2-approximation [21, 16]. In special cases where the matroid is uniform ($\mathcal{I} = \{S : |S| \leq k\}$) [20] or the submodular function is of a special type [4, 2], a $(1 - 1/e)$-approximation has been achieved and this is optimal for these problems in the value oracle model [22, 6, 15]. A $(1 - 1/e)$-approximation for the general Submodular Welfare Problem has been known only in a stronger *demand oracle* model [4], where in fact $1 - 1/e$ can be improved [9].

In this paper, we develop a randomized *continuous greedy algorithm* which achieves a $(1 - 1/e)$-approximation for the Submodular Welfare Problem in the value oracle model. We also show that the special case of n equal players is approximation resistant, in the sense that the optimal $(1 - 1/e)$-approximation is achieved by a uniformly random solution. Using the *pipage rounding* technique [1, 2], we obtain a $(1 - 1/e)$-approximation for submodular maximization subject to any matroid constraint. The continuous greedy algorithm has a potential of wider applicability, which we demonstrate on the examples of the Generalized Assignment Problem and the AdWords Assignment Problem.

Categories and Subject Descriptors

F.2 [**Theory of computing**]: Analysis of algorithms and problem complexity

General Terms

Algorithms, Economics

Keywords

Submodular functions, matroids, combinatorial auctions

1. INTRODUCTION

A function $f : 2^X \to \mathbb{R}$ is *monotone* if $f(S) \leq f(T)$ whenever $S \subseteq T$. We say that f is *submodular*, if

$$f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$$

for any S, T. Usually we also assume that $f(\emptyset) = 0$. Submodular functions arise naturally in combinatorial optimization, e.g. as rank functions of matroids, in covering problems, graph cut problems and facility location problems [5, 18, 24]. Unlike minimization of submodular functions which can be done in polynomial time [10, 23], problems involving submodular maximization are typically NP-hard. Research on problems involving maximization of monotone submodular functions dates back to the work of Nemhauser, Wolsey and Fisher in the 1970's [20, 21, 22].

1.1 Combinatorial auctions

Recently, there has been renewed interest in submodular maximization due to applications in the area of *combinatorial auctions*. In a combinatorial auction, n players compete for m items which might have different value for different players, and also depending on the particular combination of items allocated to a given player. In full generality, this is expressed by the notion of *utility function* $w_i : 2^{[m]} \to \mathbb{R}_+$ which assigns a value to each set of items potentially allocated to player i. Since this is an amount of information exponential in m, we have to clarify how the utility functions are accessible to an algorithm. Unless we consider a special class of utility functions with a polynomial-size representation, we typically resort to an *oracle model*. An oracle answers a certain type of queries about a utility function. Two types of oracles have been commonly considered:

- **Value oracle.** The most basic query is: *What is the value of* $w_i(S)$*?* An oracle answering such queries is called a *value oracle*.

- **Demand oracle.** Sometimes, a more powerful oracle is considered, which can answer queries of the following type: *Given an assignment of prices to items* $p : [m] \to \mathbb{R}$*, which set* S *maximizes* $w_i(S) - \sum_{j \in S} p_j$*?* Such an oracle is called a *demand oracle*.

Our goal is to find disjoint sets $S_1, \ldots, S_n$ maximizing the total welfare $\sum_{i=1}^{n} w_i(S_i)$. Regardless of what type of oracle we consider, there are computational issues that make the problem very hard in general. In particular, consider players who are *single-minded* in the sense that each desires one particular set T_i, $w_i(S) = 1$ if $T_i \subseteq S$ and 0 otherwise. Then both value and demand queries are easy to answer; however, the problem is equivalent to set packing which is known to have no $m^{-1/2+\epsilon}$-approximation unless $P = NP$ [13, 27]. Thus, restricted classes of utility functions need to be considered if one intends to obtain non-trivial positive results.

A class of particular interest is the class of monotone submodular functions. An equivalent definition of this class is as follows: Let $f_S(j) = f(S+j) - f(S)$ denote the *marginal value* of item j with respect to S. (We write $S+j$ instead of $S \cup \{j\}$ to simplify notation.) Then f is monotone if $f_S(j) \geq 0$ and f is submodular if $f_S(j) \geq f_T(j)$ whenever $S \subset T$. This can be interpreted as the property of *diminishing returns* known in economics and arising naturally in certain settings. This leads to what we call the Submodular Welfare Problem.

The Submodular Welfare Problem.

Given m items and n players with monotone submodular utility functions $w_i : 2^{[m]} \to \mathbb{R}_+$, we seek a partition of the items into disjoint sets $S_1, S_2, \ldots, S_n$ in order to maximize $\sum_{i=1}^{n} w_i(S_i)$.

This problem was first studied by Lehmann, Lehmann and Nisan [16]. They showed that a simple on-line greedy algorithm gives a 1/2-approximation for this problem. There has been no further progress over 1/2 in the value oracle model, except for lower order terms and special cases. On the negative side, it was proved that the Submodular Welfare Problem cannot be approximated to a factor better than $1-1/e$, unless $P = NP$ [15]; recently, Mirrokni, Schapira and the author showed that a better than $(1-1/e)$-approximation would require exponentially many value queries, regardless of $P = NP$ [19]. A $(1-1/e)$-approximation was developed by Dobzinski and Schapira [4]; however, their algorithm works in the *demand oracle model*. In fact, it turns out that a factor strictly better than $1-1/e$ can be achieved in the demand oracle model [9]. (The hardness result of [15] is circumvented by the demand oracle model, since demand queries on the hard instances are in fact NP-hard to answer.) In the value oracle model, Dobzinski and Schapira [4] gave an (on-line) $\frac{n}{2n-1}$-approximation, and also a $(1-1/e)$-approximation in the special case where utilities can be written explicitly as coverage functions for some set system.

The welfare maximization problem has been also considered under assumptions on utility functions somewhat weaker than submodularity, namely subadditivity[1] and fractional subadditivity[2]. In the value oracle model, $m^{-1/2+\epsilon}$-approximation for these classes of utility functions is impossible with a polynomial number value queries [19]. In contrast, the demand oracle model allows 1/2-approximation for subadditive utility functions and $(1-1/e)$-approximation for fractionally subadditive functions [7].

[1]f is subadditive if $f(S \cup T) \leq f(S) + f(T)$ for any S, T.

[2]f is fractionally subadditive if $f(S) \leq \sum_T \alpha_T f(T)$ for any $\alpha_T \geq 0$ such that $\forall j \in S; \sum_{T: j \in T} \alpha_T \geq 1$.

1.2 Submodular maximization subject to a matroid constraint

The Submodular Welfare Problem is in fact a special case of the following problem, considered by Fisher, Nemhauser and Wolsey in 1978 [21]. The reduction appears in [17]; we review it briefly in Section 2.

Submodular maximization subject to a matroid constraint.

Given a monotone submodular function $f : 2^X \to \mathbb{R}_+$ (by a value oracle) and a matroid [3] $\mathcal{M} = (X, \mathcal{I})$ (by a membership oracle), find $\max_{S \in \mathcal{I}} f(S)$.

In [20], Nemhauser, Wolsey and Fisher studied the special case of a *uniform matroid*, $\mathcal{I} = \{S \subset X : |S| \leq k\}$. In this case, they showed that a greedy algorithm yields a $(1-1/e)$-approximation. Nemhauser and Wolsey also proved that a better approximation is impossible with a polynomial number of value queries [22]. Note that the Max k-cover problem, to find k sets maximizing $|\bigcup_{j \in K} A_j|$, is a special case of submodular maximization subject to a uniform matroid constraint. Feige proved that $1-1/e$ is optimal even for this explicitly posed problem, unless $P = NP$ [6].

For general matroids, the greedy algorithm delivers a factor of 1/2 [21]. Several subsequent results indicated that it might be possible to improve the factor to $1-1/e$: apart from the case of uniform matroids, $(1-1/e)$-approximation was also obtained for a "maximum coverage problem with group budget constraints" [3, 1], for a related problem of submodular maximization subject to a knapsack constraint [25], and most recently for any matroid constraint and a special class of submodular functions, *sums of weighted rank functions* [2]. A key ingredient in the last result is the technique of *pipage rounding*, introduced by Ageev and Sviridenko [1] and adapted to matroid optimization problems by Calinescu et al. [2]. In [2], this technique converts a fractional LP solution into a feasible integral solution. Nonetheless, since it is unclear how to write a suitable LP for a general submodular function given by a value oracle, this approach seems to apply only to a restricted class of functions.

1.3 Our results

In this paper, we design a randomized algorithm which provides a $(1-1/e)$-approximation for the Submodular Welfare Problem, using only value queries. The algorithm can be viewed as a *continuous greedy process* which delivers an approximate solution to a *non-linear continuous optimization problem*. Employing the technique of *pipage rounding* [1, 2], we also obtain a $(1-1/e)$-approximation for submodular maximization subject to an arbitrary matroid constraint. This settles the approximation status of these problems in the value oracle model. Interestingly, in the special case of Submodular Welfare with n equal players, the optimal $(1-1/e)$-approximation is in fact obtained by a uniformly random solution.

The continuous greedy algorithm can be also applied to problems where the underlying matroid is exponentially large. Our algorithm provides a $(1-e^{-\alpha})$-approximation, assuming that we have an α-approximation for maximizing certain linear functions over the matroid. This improves the

[3]A matroid is a system of "independent sets", generalizing the notion of linear independence of vectors [24].

previously known factors of $\alpha(1-1/e)$ [11] and $\alpha/(1+\alpha)$ [2, 12] in this framework. In particular, we obtain a $(1-1/e-o(1))$-approximation for the Generalized Assignment Problem (which is suboptimal but more practical than previously known algorithms [11, 9]), and a $(1-e^{-1/2}-o(1))$-approximation for the AdWords Assignment Problem (improving the previously known $(1/3-o(1))$-approximation [12]).

Remark. The notion of a *continuous greedy algorithm* has been used before. Wolsey proposed a continuous greedy algorithm for the problem of maximizing a "real-valued submodular function" over a knapsack polytope already in 1982 [26]. We note that this algorithm is somewhat different from ours. Wolsey's algorithm increases one fractional variable at a time, the one maximizing local gain. In our setting, his algorithm in fact reduces to the standard greedy algorithm which yields only a 1/2-approximation.

Our insight is that a suitable variant of the continuous greedy method lends itself well to non-linear optimization problems with certain analytical properties. This bears some resemblance to gradient descent methods used in continuous optimization. However, ours is not a continuous process converging to a local optimum; rather, it is a process running for a fixed amount of time resulting in a globally approximate solution. We discuss this in Section 3. In Section 4, we present a discrete version of our algorithm which, combined with pipage rounding, gives a $(1-1/e)$-approximation for submodular maximization subject to a matroid constraint. In the special case of the Submodular Welfare Problem, pipage rounding is actually not needed. We present a self-contained $(1-1/e)$-approximation algorithm in Section 5. Finally, we discuss a generalization of our framework in Section 6.

2. PRELIMINARIES

Smooth submodular functions.

A discrete function $f : 2^X \to \mathbb{R}$ is submodular, if $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$ for any $S, T \subseteq X$. As a continuous analogy, Wolsey [26] defines submodularity for a function $F : [0,1]^X \to \mathbb{R}$ as follows:

$$F(x \vee y) + F(x \wedge y) \leq F(x) + F(y) \qquad (1)$$

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 $F : [0,1]^X \to \mathbb{R}$ 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 function $f : 2^X \to \mathbb{R}_+$, a canonical extension to a smooth monotone submodular function can be obtained as follows [2]: 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. 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.

Matroid polytopes.

We consider polytopes $P \subset \mathbb{R}_+^X$ 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 [5], an equivalent description is

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

Here, $r_M(S) = \max\{|I| : I \subseteq S \ \& \ I \in \mathcal{I}\}$ is the *rank function* of matroid $\mathcal{M}$. From this description, it is clear that $P(\mathcal{M})$ is down-monotone.

Pipage rounding.

A very useful tool that we invoke is a rounding technique introduced by Ageev and Sviridenko [1], and adapted for the matroid polytope by Calinescu et al. [2]. We use it here as the following black box.

Lemma 2.2. *There is a polynomial-time randomized algorithm which, given a membership oracle for matroid $\mathcal{M} = (X, \mathcal{I})$, a value oracle for a monotone submodular function $f : 2^X \to \mathbb{R}_+$, and $y \in P(\mathcal{M})$, returns an independent set $S \in \mathcal{I}$ of value $f(S) \geq (1-o(1))\mathbf{E}[f(\hat{y})]$ with high probability.*

The $o(1)$ term can be made polynomially small in $n = |X|$. Since any fractional solution $y \in P(\mathcal{M})$ can be converted to an integral one without significant loss in the objective function, we can consider the continuous problem $\max\{F(y) : y \in P(\mathcal{M})\}$ instead of $\max\{f(S) : S \in \mathcal{I}\}$.

Remark. In previous work [2], the pipage rounding technique was used in conjunction with linear programming to approximate $\max\{f(S) : S \in \mathcal{I}\}$ for a special class of submodular functions. However, it is not clear how to write a linear program for a submodular function without any special structure. Therefore, we abandon this approach and attack the non-linear optimization problem $\max\{F(y) : y \in P(\mathcal{M})\}$ directly.

The Submodular Welfare Problem.

It is known that the Submodular Welfare Problem is a special case of submodular maximization subject to a matroid constraint [17]. To fix notation, let us review the reduction here. Let the set of n players be P, the set of m items Q, and for each $i \in P$, let the respective utility function be $w_i : 2^Q \to \mathbb{R}_+$. We define a new ground set $X = P \times Q$, with a function $f : 2^X \to \mathbb{R}_+$ defined as follows: Every set $S \subseteq X$ can be written uniquely as $S = \bigcup_{i\in P}(\{i\}\times S_i)$. Then let

$$f(S) = \sum_{i\in P} w_i(S_i).$$

Assuming that each w_i is a monotone submodular function, it is easy to verify that f is also monotone submodular. The interpretation of this construction is that we make $|P|$ copies of each item, one for each player. However, in reality we can only allocate one copy of each item. Therefore, let us define a partition matroid $\mathcal{M} = (X, \mathcal{I})$ as follows:

$$\mathcal{I} = \{S \subseteq X \mid \forall j; |S \cap (P \times \{j\})| \leq 1\}.$$

Then the Submodular Welfare Problem is equivalent to max $\{f(S) : S \in \mathcal{I}\}$. Due to Lemma 2.2, we know that instead of this discrete problem, it suffices to consider the non-linear optimization problem $\max\{F(y) : y \in P(\mathcal{M})\}$. We remark that in this special case, we do not need the full strength of Lemma 2.2; instead, $y \in P(\mathcal{M})$ can be converted into an integral solution by simple randomized rounding. We return to this issue in Section 5.

3. THE CONTINUOUS GREEDY PROCESS

In this section, we present the analytic picture behind our algorithm. This section is not formally needed for our main result. Our point of view is that the analytic intuition explains why $1 - 1/e$ arises naturally in problems involving maximization of monotone submodular functions. We consider any down-monotone polytope P and a smooth monotone submodular function F. For concreteness, the reader may think of the matroid polytope and the function $F(y) = \mathbf{E}[f(\hat{y})]$ defined in the previous section. 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 equal to 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, $F(y+v^*) = F(x^* \vee y) \geq F(x^*) = OPT$. Note that $y + v^*$ is not necessarily in P but this is not an issue. 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 the mean value theorem, there is some $c \in [0, 1]$ such that $F(y + v^*) - F(y) = \frac{dF}{d\xi}\big|_{\xi=c} \leq \frac{dF}{d\xi}\big|_{\xi=0} = v^* \cdot \nabla F(y)$. Since $v^* \in P$, and $v(y) \in P$ maximizes $v \cdot \nabla F(y)$, we get

$$v(y)\cdot\nabla F(y) \geq v^* \cdot \nabla F(y) \geq F(y+v^*) - F(y) \geq OPT - F(y). \tag{2}$$

Now let us return to our continuous process and analyze $F(y(t))$. By the chain rule and using (2), 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 $F(y(t)) \geq (1 - e^{-t})OPT$.

4. SUBMODULAR MAXIMIZATION SUBJECT TO A MATROID CONSTRAINT

Let us assume now that the polytope in question is a matroid polytope $P(\mathcal{M})$ and the smooth submodular function is $F(y) = \mathbf{E}[f(\hat{y})]$. We consider the question how to find a fractional solution $y \in P(\mathcal{M})$ of value $F(y) \geq (1-1/e)OPT$. The continuous greedy process 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. We show how to handle this issue for the particular choice of $F(y) = \mathbf{E}[f(\hat{y})]$.

- 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.

In analogy with Equation (2), we use a suitable bound on the optimum. Note that $\frac{\partial F}{\partial y_j} = \mathbf{E}[f(\hat{y}) \mid \hat{y}_j = 1] - \mathbf{E}[f(\hat{y}) \mid \hat{y}_j = 0]$; we replace this by $\mathbf{E}[f_{\hat{y}}(j)]$ which works as well and will be more convenient.

LEMMA 4.1. *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, $OPT \leq \mathbf{E}[f(R) + \sum_{j \in O} f_R(j)] = F(y) + \sum_{j \in O} \mathbf{E}[f_R(j)] \leq F(y) + \max_{I \in \mathcal{I}} \sum_{j \in I} \mathbf{E}[f_R(j)]$. $\square$

The Continuous Greedy Algorithm.
Given: matroid $\mathcal{M} = (X, \mathcal{I})$, monotone submodular function $f : 2^X \to \mathbb{R}_+$.

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, $y(1) = \delta \sum_t \mathbf{1}_{I(t)} \in P(\mathcal{M})$. In the second stage of the algorithm, we take the fractional solution $y(1)$ and apply *pipage rounding* to it. Considering Lemma 2.2, it suffices to prove the following.

LEMMA 4.2. *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 4.1, $\delta = 1/n^2$ 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, $F(y(1)) \geq (1-1/e)\tilde{OPT} \geq (1-1/e-o(1))OPT$. $\square$

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 $R(t+\delta) = R(t) \cup \tilde{D}(t)$. Hence the analysis goes through with $\tilde{D}(t)$ and we get $F(y(t+\delta)) - F(y(t)) \geq \sum_j \Pr[\tilde{D}(t) = \{j\}]\mathbf{E}[f_{R(t)}(j)] \geq \delta(1-n\delta)\sum_{j \in I(t)} \mathbf{E}[f_{R(t)}(j)]/(1 - y_j(t))$. Observe that this is equivalent to our previous analysis when $y_j(t) = 0$, but we get a small gain as the fractional variables increase.

Denote $\omega^*(t) = \max_j \omega_j(t)$. The element achieving this is always part of $I(t)$. By Lemma 4.1 and submodularity, we know that at any time, $\omega^*(t) \geq \frac{1}{n}(\tilde{OPT} - F(y(t)))$, where $\tilde{OPT} = (1 - o(1))OPT$ depends on the accuracy of our sampling estimates. 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 $j^*(t)$ (summing up to 1 overall), at most half of them can occur when $y_{j^*(t)}(t) < \frac{1}{2n}$. Let's call these steps "bad", and the steps where $y_{j^*(t)}(t) \geq \frac{1}{2n}$ "good". In a good step, we have $\omega^*(t)/(1 - y_{j^*}(t)) \geq \omega^*(t) + \frac{1}{2n}\omega^*(t) \geq \omega^*(t) + \frac{1}{2n^2}(\tilde{OPT} - F(y(t)))$ instead of $\omega^*(t)$ in the original analysis. In good steps, the improved analysis gives

$$F(y(t+\delta)) - F(y(t)) \geq \delta(1-n\delta)(1 + \frac{1}{2n^2})(\tilde{OPT} - F(y(t))).$$

By taking $\delta = o(\frac{1}{n^3})$, we get that $F(y(t+\delta)) - F(y(t)) \geq \delta(1+n\delta)(\tilde{OPT} - F(y(t)))$. Then, we can conclude that in good steps, we have

$$\tilde{OPT} - F(y(t+\delta)) \leq (1-\delta-n\delta^2))(\tilde{OPT} - F(y(t))),$$

while in bad steps

$$\tilde{OPT} - F(y(t+\delta)) \leq (1-\delta+n\delta^2))(\tilde{OPT} - F(y(t))).$$

Overall, we get

$$\begin{aligned} F(y(1)) &\geq (1-(1-\delta+n\delta^2)^{\frac{1}{2\delta}}(1-\delta-n\delta^2)^{\frac{1}{2\delta}})\tilde{OPT} \\ &\geq (1-(1-\delta)^{1/\delta})\tilde{OPT} \geq (1-1/e+\Omega(\delta))\tilde{OPT}. \end{aligned}$$

We can also make our sampling estimates accurate enough so that $\tilde{OPT} = (1-o(\delta))OPT$. We conclude that $F(y(1)) \geq (1-1/e+\Omega(\delta))OPT$.

Pipage rounding. Finally, we use pipage rounding to convert y into an integral solution. Using Lemma 2.2, we obtain an independent set S of value $f(S) \geq (1-1/e)OPT$ w.h.p.

5. SUBMODULAR WELFARE

In this section, we return to the Submodular Welfare Problem. We deal with a partition matroid $\mathcal{M}$ here which allows us to simplify our algorithm and avoid the technique of pipage rounding. For a set of players P and a set of items Q, the new ground set is $X = P \times Q$, hence it is natural to associate variables y_{ij} with player-item pairs. The variable expresses the extent to which item j is allocated to player i. In each step, we estimate the expected marginal value ω_{ij} of element (i,j), i.e. the expected marginal profit that player i derives from adding item j. With respect to these weights, we find a maximum independent set in $\mathcal{M}$; this means selecting a *preferred player* for each item j, who derives the maximum marginal profit from j. Then we increase the respective variable for each item and its preferred player.

The partition matroid polytope can be written as

$$P(\mathcal{M}) = \{y \geq 0 : \forall j; \sum_{i=1}^{n} y_{ij} \leq 1\}.$$

The continuous greedy algorithm finds a point $y \in P(\mathcal{M})$ of value $F(y) \geq (1-1/e)OPT$. Here,

$$F(y) = \mathbf{E}[f(\hat{y})] = \sum_{i=1}^{n} \mathbf{E}[f(R_i)]$$

where R_i contains each item j independently with probability y_{ij}. Formally, the sets R_i should also be sampled independently for different players, but this does not affect the expectation. We can modify the sampling model and instead allocate each item independently to exactly one player, item j to player i with probability y_{ij}. This is possible because $\sum_{i=1}^{n} y_{ij} \leq 1$, and yields a feasible allocation of expected value $F(y)$. We obtain the following self-contained algorithm.

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.

Lemma 4.2 and the discussion above imply our main result.

THEOREM 5.1. *The Continuous Greedy Algorithm gives a $(1-1/e-o(1))$-approximation (in expectation) for the Submodular Welfare Problem in the value oracle model.*

Remark 1. Our choices of δ and the number of random samples are pessimistic and can be improved in this special case. Again, the $o(1)$ error term can be eliminated by a more careful analysis. We believe that our algorithm is quite practical, unlike the $(1-1/e)$-approximation using demand queries [4], or its improvement [9], which both require the ellipsoid method.

Remark 2. In the special case where all n players have the same utility function, we do not need to run any nontrivial algorithm at all. It can be seen from the continuous greedy process (Section 3) that in such a case the greedy trajectory is equal to $y_{ij}(t) = \frac{1}{n}t$ for all i, j (WLOG - out of all optimal directions, we pick the symmetric one, $v = (\frac{1}{n}, \ldots, \frac{1}{n})$). Hence, the fractional solution that we obtain is $y_{ij} = \frac{1}{n}$ for all i, j and we know that it satisfies $F(y) \geq (1-1/e)OPT$. Thus, a $(1-1/e)$-approximation is obtained by allocating each item uniformly at random. It is interesting to note that the hardness results [15, 19] hold even in this special case. Therefore, the problem of n equal submodular utility functions is *approximation resistant* in the sense that it is impossible to beat a blind random solution.

5.1 Counterexample to a possible simplification of the algorithm

Here we consider a possible way to simplify our continuous greedy algorithm. Instead of applying pipage rounding or randomized rounding to the fractional solution $y(1) = \delta \sum_t \mathbf{1}_{I(t)}$, it seems reasonable to compare all the independent sets $I(t)$ in this linear combination and return the most valuable one. However, the non-linearity of the objective function defeats this intuitive approach. Quite surprisingly, the value of each $I(t)$ can be an arbitrarily small fraction of OPT.

Example. Consider an instance arising from the Submodular Welfare Problem, where we have n players and n items. The utility of each player is equal to $w(S) = \min\{|S|, 1\}$; i.e., each player wants at least one item. Obviously, the optimal solution assigns one item to each player, which yields $OPT = n$.

The continuous greedy algorithm (see Section 5) builds a fractional solution y. Given a partial solution after a certain number of steps, we consider the random sets R_i defined by y and the expected marginal values given by

$$\omega_{ij} = \mathbf{E}[w(R_i + j) - w(R_i)] = \Pr[R_i = \emptyset] = \prod_{j=1}^{n}(1 - y_{ij}).$$

A preferred player is chosen for each item by selecting the largest ω_{ij} and incrementing the respective variable y_{ij}. The algorithm may run in such a way that in each step, $y_{ij}(t) = \beta_i(t)$ for all j, and the preferred player for all items is the same. This is true at the beginning ($y_{ij}(0) = 0$), and assuming inductively that $y_{ij}(t) = \beta_i$ for all j, we have $\omega_{ij} = (1 - \beta_i)^n$. The choice of a preferred player for each item is given by minimizing β_i, and we can assume that the algorithm selects the same player i^* for each item. In the next step, we will have $y_{ij}(t + \delta) = \beta_i + \delta$ if $i = i^*$, and β_i otherwise. Hence again, $y_{ij}(t + \delta)$ depends only on i.

Eventually, the algorithm finds the fractional solution $y_{ij} = 1/n$ for all i, j. By randomized rounding (or pipage rounding), we obtain expected value $n(1 - (1 - 1/n)^n)$. However, the solution found by the continuous greedy algorithm in each step consists of all items being assigned to the same player, which yields value 1.

6. OTHER APPLICATIONS

Finally, let us discuss a generalization of our framework. Let us consider a setting where we cannot optimize linear functions over P exactly, but only α-approximately ($\alpha < 1$). Let us consider the continuous setting (Section 3). Assume that in each step, we are able to find a vector $v(y) \in P$ such that $v(y) \cdot \nabla F(y) \geq \alpha \max_{v \in P} v \cdot \nabla F(y) \geq \alpha(OPT - F(y))$. 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 4. This has interesting applications.

The Separable Assignment Problem.

An instance of the Separable Assignment Problem (SAP) consists of m items and n bins. Each bin i has an associated collection of *feasible sets* $\mathcal{F}_i$ which is down-closed ($A \in \mathcal{F}_i, B \subseteq A \Rightarrow B \in \mathcal{F}_i$). Each item j has a value v_{ij}, depending on the bin i where it's placed. The goal is to choose disjoint feasible sets $S_i \in \mathcal{F}_i$ so as to maximize $\sum_{i=1}^{n} \sum_{j \in S_i} v_{ij}$.

Reduction to a matroid constraint. Let us review the reduction from [2]. We define $X = \{(i, S) \mid 1 \leq i \leq n, S \in \mathcal{F}_i\}$ and a function $f : 2^X \rightarrow \mathbb{R}_+$,

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

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 (i, S) for each i. Such a set $\mathcal{S}$ corresponds to an assignment of set S to bin i for each $(i, S) \in \mathcal{S}$. 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 continuous greedy algorithm for SAP. The ground set of $\mathcal{M}$ is exponentially large here, so we cannot use the algorithm of Section 4 as a black box. First of all, the number of steps in the continuous greedy algorithm depends on the discretization parameter δ. It can be seen that it is sufficient here to choose δ polynomially small in the rank of $\mathcal{M}$, which is n. The algorithm works with variables corresponding to the ground set X; let us denote them by $x_{i,S}$ where $S \in \mathcal{F}_i$. Note that in each step, only n variables are incremented (one for each bin i) and hence the number of nonzero variables remains polynomial. Based on these variables, we can generate a random set $\mathcal{R} \subset X$ in each step. However, we cannot estimate all marginal values $\omega_{i,S} = \mathbf{E}[f_{\mathcal{R}}(i, S)]$ since these are exponentially many. What we do is the following.

For each element j, we estimate $\omega_{ij} = \mathbf{E}[f_{\mathcal{R}}(i, j)]$, where $f_{\mathcal{R}}(j) = f(\mathcal{R} + (i, j)) - f(\mathcal{R})$, the marginal profit of adding item j to bin i, compared to its assignment in $\mathcal{R}$. Then $\omega_{i,S} = \sum_{j \in S} \omega_{ij}$ for any set S. Finding a maximum-weight independent set $I \in \mathcal{I}$ means finding the optimal set S_i for each bin i, given the weights $\omega_{i,S}$. This is what we call the *single-bin subproblem.* We use the item weights ω_{ij} and try to find a set for each bin maximizing $\sum_{j \in S} \omega_{ij}$. If we can solve this problem α-approximately ($\alpha < 1$), we can also find an α-approximate maximum-weight independent set I. Consequently, we obtain a $(1 - e^{-\alpha})$-approximation for the Separable Assignment Problem. This beats both the factor $\alpha(1 - 1/e)$ obtained by using the Configuration LP [11] and the factor $\alpha/(1 + \alpha)$ obtained by a simple greedy algorithm [2, 12].

The Generalized Assignment Problem.

Special cases of the Separable Assignment Problem are obtained by considering different types of collections of feasible sets $\mathcal{F}_i$. When each $\mathcal{F}_i$ is given by a knapsack problem, $\mathcal{F}_i = \{S : \sum_{j \in S} s_{ij} \leq 1\}$, 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 $(1 - o(1))$-approximation for the Generalized Assignment Problem. We remark that a $(1 - 1/e + \epsilon)$-approximation can be achieved for some very small $\epsilon > 0$, using an exponentially large "Configuration LP" and the ellipsoid method [9]; in comparison, our new algorithm is much more practical.

The AdWords Assignment Problem.

A related problem defined in [11] is the AdWords Assignment Problem (AAP). Here, bins correspond to rectangular areas associated with keywords where certain ads can be displayed. Each bidder has a rectangular ad of given dimensions that might be displayed in multiple areas. Bidder j is willing to pay v_{ij} to have his ad displayed in area i, but overall his spending is limited by a budget B_j.

A reduction to submodular maximization subject to a matroid constraint is given in [12]. We have a ground set $X = \{(i, S) : S \in \mathcal{F}_i\}$ where $\mathcal{F}_i$ is the collection of feasible sets of ads that fit in area i. The matroid constraint is that we choose at most one set S for each area i. The sets are not necessarily disjoint. The objective function is

$$f(\mathcal{S}) = \sum_j \min \left\{ \sum_{(i,S) \in \mathcal{S} : j \in S} v_{ij}, B_j \right\}.$$

Again, this function is monotone and submodular. Given a random assignment $\mathcal{R}$, the expected marginal value of an element (i, S) can be written as $\mathbf{E}[f_{\mathcal{R}}(i, S)] = \sum_{j \in S} \omega_{ij}$, where $\omega_{ij} = \mathbf{E}[V_j(\mathcal{R} + (i, j)) - V_j(\mathcal{R})]$, and $V_j(\mathcal{R})$ is the amount spent by bidder j in $\mathcal{R}$. We can estimate the values ω_{ij} by random sampling and then find a $(1/2 - o(1))$-approximation to $\max_{S \in \mathcal{F}_i} \mathbf{E}[f_{\mathcal{R}}(i, S)]$ by using the rectangle packing algorithm of [14] with weights ω_{ij}. Consequently, we can approximate the maximum-weight independent set within a factor of $1/2 - o(1)$ in each step and obtain a $(1 - e^{-1/2} - o(1))$-approximation for AAP. This beats the $\frac{1}{2}(1 - 1/e)$-approximation given in [11] and also the $(\frac{1}{3} - o(1))$-approximation in [12].

Acknowledgements.
The author is grateful to all those who have participated in various inspiring discussions on this topic: Chandra Chekuri, Uri Feige, Mohammad Mahdian, Vahab Mirrokni and Benny Sudakov. Special thanks are due to Chandra Chekuri for introducing me to submodular maximization subject to a matroid constraint and pointing out some important connections.

7. REFERENCES

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

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

[3] C. Chekuri and A. Kumar. Maximum coverage problem with group budget constraints and applications, *Proc. of APPROX* 2004, *Lecture Notes in Computer Science* 3122, 72–83.

[4] S. Dobzinski and M. Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders, *Proc. of* 17th *SODA* (2006), 1064–1073.

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

[6] U. Feige. A threshold of $\ln n$ for approximating Set Cover, *Journal of the ACM* 45 (1998), 634–652.

[7] U. Feige. Maximizing social welfare when utility functions are subadditive, *Proc. of* 38th *STOC* (2006), 41–50.

[8] U. Feige, V. Mirrokni and J. Vondrák. Maximizing non-monotone submodular functions, *Proc. of 48th FOCS* (2007), 461–471.

[9] U. Feige and J. Vondrák. Approximation algorithms for combinatorial allocation problems: Improving the factor of $1 - 1/e$, *Proc. of 47th FOCS* (2006), 667–676.

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

[11] L. Fleischer, M. X. Goemans, V. Mirrokni and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems, *Proc. of* 17th *ACM-SIAM SODA* (2006), 611–620.

[12] P. Goundan and A. Schulz. Revisiting the greedy approach to submodular set function maximization, *manuscript* (2007).

[13] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$, *Acta Mathematica* 182 (1999), 105–142.

[14] K. Jansen and G. Zhang. On rectangle packing: maximizing benefits, *Proc. of* 15th *ACM-SIAM SODA* (2004), 204–213.

[15] S. Khot, R. Lipton, E. Markakis and A. Mehta. Inapproximability results for combinatorial auctions with submodular utility functions, *Proc. of WINE 2005*, *Lecture Notes in Computer Science* 3828, 92–101.

[16] B. Lehmann, D. J. Lehmann and N. Nisan. Combinatorial auctions with decreasing marginal utilities, *ACM Conference on El. Commerce* 2001, 18–28.

[17] B. Lehmann, D. J. Lehmann and N. Nisan. Combinatorial auctions with decreasing marginal utilities (journal version), *Games and Economic Behavior* 55 (2006), 270–296.

[18] L. Lovász. Submodular functions and convexity. A. Bachem et al., editors, *Mathematical Programmming: The State of the Art*, 235–257.

[19] V. Mirrokni, M. Schapira and J. Vondrák. Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions, *manuscript* (2007).

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

[21] M. L. Fisher, G. L. Nemhauser and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions II, *Math. Programming Study* 8 (1978), 73–87.

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

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

[24] A. Schrijver. Combinatorial optimization - polyhedra and efficiency, *Springer-Verlag Berlin Heidelberg*, 2003.

[25] M. Sviridenko. A note on maximizing a submodular set function subject to knapsack constraint, *Operations Research Letters* 32 (2004), 41–43.

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

[27] D. Zuckerman. Linear degree extractors and inapproximability, *Theory of Computing* 3 (2007), 103–128.