Demand Queries with Preprocessing[*]

Uriel Feige[†] and Shlomo Jozeph[‡]

May 1, 2014

Abstract

Given a set of items and a submodular set-function f that determines the value of every subset of items, a demand query assigns prices to the items, and the desired answer is a set S of items that maximizes the profit, namely, the value of S minus its price. The use of demand queries is well motivated in the context of combinatorial auctions. However, answering a demand query (even approximately) is NP-hard. We consider the question of whether exponential time preprocessing of f prior to receiving the demand query can help in later answering demand queries in polynomial time. We design a preprocessing algorithm that leads to approximation ratios that are NP-hard to achieve without preprocessing. We also prove that there are limitations to the approximation ratios achievable after preprocessing, unless NP $\subset$ P/poly.

1 Introduction

Given a universe U of n items, a *valuation function* is a set-function f that assigns nonnegative integer values to every subset of items, and satisfies the following three properties:

- *Normalization*: $f(\emptyset) = 0$.
- *Monotonicity*: for every two sets $S \subseteq T \subseteq U$, $f(S) \leq f(T)$.
- *M-bounded*: there is some fixed constant c independent of n such that $f(U)$ (the value assigned to the whole universe, which we denote by M) satisfies $\log M \leq n^c$. Hence every single value of the function f can be represented by a number of bits that is polynomial in n.

Valuation functions are used in order to represent the internal preferences of bidders in combinatorial auctions, where the items of U are for sale. The *maximum welfare*

[*]To appear in ICALP 2014

[†]Department of Computer Science and Applied Mathematics, the Weizmann Institute, Rehovot, Israel. uriel.feige@weizmann.ac.il

[‡]Department of Computer Science and Applied Mathematics, the Weizmann Institute, Rehovot, Israel. shlomo.jozeph@weizmann.ac.il

allocation problem associated with a combinatorial auction is the following: given the valuation functions of the bidders (f_i for bidder i), one needs to give each bidder i one *bundle* $B_i \subseteq U$ of items (a bundle is simply a set, which may also be empty), and these bundles need to be disjoint ($B_i \bigcap B_j = \emptyset$ for $i \neq j$). The objective is to do so while maximizing $\sum_i f_i(B_i)$, which is referred to as the welfare of the allocation.

Ideally, one would like to compute the maximum welfare allocation in time polynomial in n. However, there are two obstacles to overcome.

- The communication bottleneck. An explicit representation of a valuation function (as a table) might take space $2^n \log M$, and hence bidders might not be able to communicate their valuation functions to the seller.
- The computation bottleneck. Even if valuation functions have *succinct representations* of polynomial size (for example, the case of *single minded bidders* where f is determined by a single bundle SB, with $f(S) = 1$ if $SB \subseteq S$, and $f(S) = 0$ otherwise), the problem of computing the maximum welfare allocation is NP-hard, and also at least as hard to approximate as the notorious *set packing* problem.

Remark 1 *Another difficulty associated with combinatorial auctions is how to provide incentives to the bidders to reveal their true valuation function to the seller, but this game-theoretic aspect is beyond the scope of our current work.*

One way of handling the communication bottleneck is by allowing the seller to ask the bidder *queries* regarding the nature of his valuation function f. One wishes to design a polynomial time allocation algorithm in which the seller makes only polynomially many queries to each bidder. One natural class of queries is that of *value queries*: the query is a bundle S and the reply is its value $f(S)$. The class of queries that is the focus of this work is called *demand queries* (see [15] for several types of queries commonly used). For demand queries, one assumes that the utilities for bidders can be separated into two components: the value of the bundle received, and the payment that the bidder pays. Namely, if the bidder received bundle B and pays for it a price P, then the utility derived by bidder is $f(B) - P$. A demand query is a vector $\bar{p}$ of nonnegative integer prices to items ($\bar{p}(j)$ for item j) and its answer is the most preferable bundle for the bidder under these prices, namely, a bundle S that maximizes $f(S) - \sum_{j \in S} \bar{p}(j)$, together with the value $f(S)$. The ability to answer demand queries appears to be a natural requirement from a bidder, as without this ability, if the bidder comes to a market in which the items have prices, he himself would not know what he prefers to buy. Moreover, the assumption that bidders can answer demand queries turns out to be very beneficial for algorithms that approximate the maximum welfare problem. (Demand queries implement a separation oracle for the dual of the *configuration LP*, and solving this configuration LP is a first step in many of the approximation algorithms for maximum welfare.)

Despite their attractiveness, demand queries are problematic in the sense that even if a succinct representation of a valuation function is given (which allows efficient replies to value queries), answering demand queries is in general NP-hard (and also hard to approximate). This NP-hardness allows for the situation that for certain classes of valuation functions (such as submodular functions, to be defined shortly), the approximation ratio achieved for the maximum welfare problem if demand queries are allowed is strictly better (unless P=NP) than without demand queries (but with succinct representations of the valuation functions). See Section 1.1.

In the current work, we investigate a certain approach for reconciling the NP-hardness of demand queries with the desirability of being able to answer them. A valuation function with a succinct representation would in general have more than one such representation. Could it be that NP-hardness of answering demand queries is a consequence of the choice of representation, but under a different representation answering demand queries is easy? We model this question using the notion of *preprocessing*.

We envision the following situation. First, an arbitrary valuation function f is given. One is allowed to preprocess f for arbitrary amount of time, even more than exponential. The outcome of this preprocessing phase is a polynomial size advice string $A(f)$. Thereafter, upon being given a vector $\bar{p}$ of prices as a demand query, the demand query is answered in polynomial time based only on $\bar{p}$, $A(f)$, and at most polynomially many value queries. (Typically, the original representation of f is succinct and allows efficient replies to value queries. In these cases this succinct representation can be made part of $A(f)$. However, some functions (in fact, almost all functions) do not have a succinct representation, and for them we assume access to a *value oracle* that can answer value queries.) The focus of our work is that of approximate answers to demand queries. The quality of the answering algorithm is measured by the *approximation ratio*: the ratio between $f(S) - \sum_{j \in S} p_j$ and $f(T) - \sum_{j \in T} p_j$, where S is the optimal solution and T is the solution returned by the algorithm.

Our results focus on the well studied class of *submodular* valuation functions (see Section 1.2 for definitions). Though this is considered to be a relatively simple class, answering demand queries is very difficult even for this class. The following proposition is well known.

Proposition 2 *For every $\epsilon > 0$, given a succinct representation of a submodular valuation function (without preprocessing) and a demand query, returning an answer with approximation ratio at most $n^{1-\epsilon}$ is NP-hard.*

Our main result shows that preprocessing helps.

Theorem 3 *For every submodular valuation function f (even with no succinct representation), there is a polynomial size advice string $A(f)$, such that given any demand query $\bar{p}$, the answer can be approximated in polynomial time within a ratio of $\mathrm{O}(n^{3/4})$, based only on $\bar{p}$, $A(f)$ and value queries.*

However, there is a limit to the effectiveness of preprocessing.

Theorem 4 *For some $\delta > 0$, there are submodular valuation functions with succinct representations, for which regardless of the polynomial size advice string given, demand queries cannot be approximated within a ratio better than $\Omega(n^{\delta})$ in polynomial time, unless NP has polynomial size circuits (namely, unless NP $\subset$ P/poly).*

We also consider in our work a natural subclasses of submodular functions (that we refer to as NH, *negative hyperedges*, see section 3) and show that for this class demand queries can be approximately answered (after preprocessing) within a ratio of $O(n^{1/2})$. An even more restricted but very natural setting is that of MWIS (*maximum weight independent set*) demand queries, which is addressed in Section 2. The negative results of Theorem 4 apply already to a further restriction that we refer to as MIS (*maximum independent set*) demand queries.

1.1 Related Work

Submodular set-functions is a well studied class of functions and surveying all the literature about it is beyond the scope of this paper. We just mention a few facts that help put our work in perspective. There have been studies of trying to approximately learn (in the sense of approximately answering future *value queries*) submodular functions by making polynomially many value queries to the function (which is a weaker notion than preprocessing). Approximation ratios of $\tilde{\Theta}(\sqrt{n})$ are achievable [11]. Our Theorem 4 constructs submodular functions that have succinct representations that allow one to efficiently answer value queries exactly, and still they have no succinct representation that allows one to even approximately answer demand queries (unless NP $\subset$ P/poly). We remark that the ability to answer value queries is very powerful in the context of submodular functions. For example, it allows one to efficiently find the minimum value of the function [4], and to approximate the maximum within a factor of $1/2$ if the function is nonnegative [3].

There is much work on the maximum welfare problem, and we mention some of it. The focus on submodular valuation functions was initiated by [15]. The configuration LP and demand queries were introduced by [5]. With only value queries, the maximum welfare problem with submodular valuation functions can be approximated within a ratio of $1 - 1/e$ [6], and doing better is NP-hard, given succinct representations of the valuation functions [13]. However, with demand queries an approximation ratio better than $1 - 1/e$ is achievable [10].

Preprocessing is a natural and well studied notion in several different contexts, some of which are beyond the scope of our paper (e.g., quickly answering database queries, quickly breaking cryptographic schemes). The direction most relevant to the current work is that of preprocessing of NP-hard problems. This direction received much attention in the context of coding theory [2] and lattice problems [9, 14]. In our negative results (Theorem 4) we build on earlier work of the authors [8] that considered preprocessing for constraint satisfaction problems, and introduced the notion of *universal factor graphs* as a method for establishing limitations on what preprocessing can achieve.

1.2 Preliminaries

Submodular set-functions can be defined in several equivalent ways. We shall use the following definition.

Definition 5 *A set-function* $f : 2^{[n]} \to \mathbb{R}$ *is* submodular *if it has* decreasing marginal values*, that is, for* $x \notin A \supseteq B$, $f(A \bigcup \{x\}) - f(A) \leq f(B \bigcup \{x\}) - f(B)$.

Submodular functions need not be monotone. However, from now on we shall always assume that functions are normalized, namely, $f(\emptyset) = 0$. Let us mention some easily verifiable properties of submodular functions. Every linear function is submodular. The sum of two submodular functions (and hence also the difference between a submodular function and a linear function) is submodular. Submodular functions are *subadditive*: for a submodular function f and pairwise disjoint $\{S_i\}$, $f(\bigcup S_i) \leq \sum f(S_i)$.

The submodular functions considered in our work will typically either be a submodular valuation function (thus being integer valued, monotone, and M-bounded) or the difference between a submodular valuation function and a nonnegative integer linear function (the price vector). In the latter case, the resulting submodular function will still be M-bounded, but not necessarily monotone. A rounding down aspect used in Section 4 might result in functions having noninteger values, but these values have simple representations.

Definition 6 *Given a submodular function $f : 2^U \to \mathbb{R}$, an* optimal set *with respect to f is a set $S \subseteq U$ that maximizes $f(S)$.*

Definition 7 *Given a set-function $f : 2^U \to \mathbb{R}$, the* hypergraph representation *of f is the unique function $w_f : 2^U \to \mathbb{R}$ such that for every $A \subseteq U$,*

$$f(A) = \sum_{B \subseteq A} w_f(B)$$

Note that an explicit formula for $w_f(A)$ is $w_f(A) = f(A) - \sum_{B \subset A} w_f(B)$ (where $\subset$ denotes strict containment). This allows one to determine w_f by an inductive process, starting with $w_f(\emptyset) = f(\emptyset) = 0$ and progressing to larger sets.

Definition 8 *The* linear part *L_f of a set function f is defined as*

$$L_f(A) = \sum_{\{x\} \subseteq A} w_f(\{x\})$$

The high order part *H_f of a set function f is defined as*

$$H_f(A) = f(A) - L_f(A)$$

The linear part of f is completely determined by the value of the hypergraph representation on individual items, and hence on vertices of the corresponding hypergraph, whereas the high order part is determined by the value of the hypergraph representation on larger sets of items, and hence on hyperedges of the corresponding hypergraph.

A submodular valuation function f can be decomposed into its linear part L_f and into its high order part H_f, with $f = L_f + H_f$. A demand query is a vector $\bar{p}$ of nonnegative integer prices for the items, with the interpretation that the price of a bundle is the sum of the prices of the items that it contains. Hence $\bar{p}$ is a representation of a linear set-function $\bar{p}(S) = \sum_{i \in S} p(i)$. The desired answer for the demand query is thus an optimal set with respect to a new function $g = f - \bar{p}$. Observe that g differs from f only in its linear part. Hence equivalently, we may write $g = H_f + (L_f - \bar{p})$. Let us denote (the vector of coefficients of the linear function) $L_f - \bar{p}$ by $\bar{q}$. Observe that we may assume that the vector $\bar{q}$ is nonnegative. This can be explained as follows. g is a submodular function, and submodular functions have decreasing marginal costs. When trying to maximize the value of g we can always ignore any item x with $g(\{x\}) \leq 0$, because that item cannot possibly add positive value if included in the solution. Hence without loss of generality we may assume that such an item would not be part of the answer to the demand query. Hence all coordinates in which $\bar{q}$ is negative can be rounded up to 0 without affecting the answer to the demand query.

As a consequence of the above discussion, the preprocessing done by our algorithms will depend only on H_f and not on L_f. Then, given a demand query $\bar{p}$, we translate it as above to the corresponding $\bar{q} = L_f - \bar{p}$ (rounded up to 0), and attempt to find the set S maximizing $g(S) = H_f(S) + \bar{q}(S)$. Under this view, it is convenient to think of $\bar{q}$ rather than $\bar{p}$ as the query, and then L_f can effectively be ignored. Moreover, since H_f is not necessarily monotone, our positive results regarding preprocessing do not require f to be monotone – they hold with no change even if f is not monotone.

2 Approximately Answering MWIS Demand Queries

Given a graph $G(V, E)$ with n vertices, a *maximum weight independent set* (MWIS) query is an n dimensional vector $\bar{q}$ of nonnegative integers, where for every $1 \leq i \leq n$ entry $\bar{q}(i)$ is interpreted as the weight given to vertex i. Given G and $\bar{q}$, the goal is to output a maximum weight independent set with respect to these weights. The special case in which $\bar{q} \in \{0, 1\}^n$ is referred to as a *maximum independent set* (MIS) query. MWIS queries can easily be seen to be a special case of demand queries with respect to submodular valuation function. Let W be an upper bound on the possible weight a vertex might be assigned in a query $\bar{q}$. Then G can be thought of as a hypergraph representation of the submodular valuation function f_G, with $w_f(e) = -W$ for every edge $e \in E$, and $w_f(i) = nW$ for every vertex $i \in V$. Then the MWIS query $\bar{q}$ is equivalent to a demand query $\bar{p}$ with $\bar{p}(i) = nW - \bar{q}(i)$ for every $i \in V$, because the optimal answer to the demand query will never be a set of vertices that induces any edges (due to their large negative weight). In particular, taking $\bar{q}$ to be the all 1 vector shows that the problem of finding a maximum independent set in G can be formulated as a demand query. This observation coupled with the known $\Omega(n^{1-\epsilon})$ NP-hardness of approximation results for the maximum independent set problem [12, 16] proves Proposition 2.

As a simple introduction to our proof of Theorem 3, we show how preprocessing G helps in improving the approximation ratio for the special case of MWIS queries.

Preprocessing. Given a graph $G = (V, E)$, consider the following collection of sets defined inductively. Let $J_j = \bigcup_{i=1}^{j} I_i$ ($J_0 = \emptyset$), where I_i (for $i \geq 1$) is a maximum independent set in $G_i = (V \setminus J_{i-1}, E_i)$ (where E_i denotes the the set of those edges induced by $V \setminus J_{i-1}$). The sets I_i are referred to as the *advice sets*.

Answering a MWIS query. Given a query vector $\bar{q}$ assigning nonnegative weights to the vertices, return the advice set with highest sum of vertex weights. Namely, the advice set I_i that maximizes $\sum_{j \in I_i} \bar{q}(j)$.

Theorem 9 *The answer to the query is a $\sqrt{2n}$ approximation to the maximum weight independent set in the weighted graph $G' = (V, E, \bar{q})$.*

Proof. The following claim shows that the coloring defined by the sets $\{I_i\}$ has the property that any independent set of G is colored by at most $\sqrt{2n}$ colors.

Proposition 10 *Given an independent set I, $|\{i | I_i \bigcap I \neq \emptyset\}| < \sqrt{2n}$.*

Proof. Suppose otherwise. Let $s_j = \left| I \setminus \bigcup_{i=1}^{j-1} I_i \right|$. Let $j_1, \cdots, j_{\sqrt{2n}}$ be the last $\sqrt{2n}$ indices for which $I_{j_i} \bigcap I \neq \emptyset$, given in reverse order (that is, advice set I_{j_i} was generated in the preprocessing phase after advice set $I_{j_{i+1}}$). Hence $s_{j_1} \geq 1$, and $s_{j_i} < s_{j_{i+1}}$. Induction establishes that $s_{j_i} \geq i$. Due to the maximality of I_i, $s_i \leq |I_i|$ (otherwise, I_i is not the maximum independent set in G after removing $\bigcup_{i=1}^{j-1} I_i$). Since $I_{j_1}, \cdots, I_{j_{\sqrt{2n}}}$ are disjoint, and $|I_{j_i}| \geq i$, the union of the advice sets would contain more than n vertices, which is a contradiction. ■

Let I be a maximum weight independent set in G'. I is also an independent set in G. From the claim, I intersects at most $\sqrt{2n}$ of the I_i's. Hence for some j the weight of $I' = I \bigcap I_j$ is at least $\frac{1}{\sqrt{2n}}$ of the weight of I. This advice set I_j is an independent set and contains I', so its weight must be at least the weight of I'. Thus, the I_i of highest weight is a $\sqrt{2n}$ approximation to the weight of I. ■

3 Negative Hyperedges Set-functions

Definition 11 *A set-function $f : 2^U \to \mathbb{R}$ is said to be a* negative hyperedges *(NH) function if its hypergraph representation satisfies $w_f(S) \leq 0$ whenever $|S| > 1$.*

Observe that every NH set-function is necessarily submodular. However, a submodular set function need not be NH. For example, the function $f(S) = 1$ for all nonempty S is submodular, and its hypergraph representation is $w_f(S) = (-1)^{|S|+1}$.

As an intermediate step towards proving Theorem 3 and strictly generalizing the notion of MWIS queries considered in Section 2, we consider demand queries with respect to NH functions. We first define our building block for the preprocessing stage.

Definition 12 *Given a submodular function $f : 2^U \to \mathbb{R}$, the* greedy optimal sets for *f is a collection of sets $\{S_i\}$, defined inductively: S_1 is the optimal set with respect to f, and S_{i+1} is the optimal set with respect to f among those sets in $U \setminus \bigcup_{j=1}^{i} S_j$.*

Preprocessing. Given an NH valuation function $f : 2^U \to \mathbb{N}$ that is M-bounded and a nonnegative integer k, define the functions $f_k(A) = 2^k |A| + H_f(A)$. Namely, f_k maintains the high order part of f, and makes the linear part equal to 2^k for every item.

Denote the greedy optimal sets for f_k by $\{S_i^k\}$. These sets, for all integer k in the range $0 \leq k \leq \log M$, will be referred to as the *advice sets*.

Answering a demand query. Recall from Section 1.2, that the demand query can be thought of as a nonnegative integer vector $\bar{q}$, and the goal is to find an optimal set with respect to $H_f + \bar{q}$. Let $\tilde{y} = \text{argmax}_x \{q_x\}$ be the highest valued item in U according to $\bar{q}$. Let $T_k = \left\{x \in U | 2^k \leq q_x < 2^{k+1}\right\}$.

Answer with the highest valued set according to $H_f + \bar{q}$ from $\left\{\{\tilde{y}\}, T_k \bigcap S_i^k\right\}$, where S_i^k ranges over all advice sets.

Theorem 13 *The answer to the query is an* $\mathrm{O}\left(\sqrt{n}\right)$ *approximation to the optimal set.*

Proof. To prove this theorem, we will use the following two lemmas.

Lemma 14 *For* $|U| = n$, *let* $g : 2^U \to \mathbb{R}$ *be a submodular function such that* $g(\{x\}) = c > 0$ *for all* $x \in U$, *and let* $\{S_i\}$ *be the greedy optimal sets for* g. *Then, for every* $A \subset U$ *with* $g(A) > 0$ *there is an* i *such that* $g\left(A \bigcap S_i\right) \geq \frac{g(A)^2}{2cn}$.

Proof. Subadditivity of g implies that $g(A) \leq \sum g(A \cap S_i)$. Suppose for the sake of contradiction that $g\left(A \bigcap S_i\right) < \frac{g(A)^2}{2cn}$ for all i. Then there must be at least $\frac{2cn}{g(A)}$ greedy sets. Let $s_j = g\left(A \setminus \bigcup_{i=1}^{j-1} S_i\right)$. Note that $s_j - s_{j+1} < \frac{g(A)^2}{2cn}$ (otherwise, $g\left(A \bigcap S_j\right) \geq \frac{g(A)^2}{2cn}$ by the subadditivity of g). Observe further that $g(S_j) \geq s_j$ (otherwise, S_j is not the optimal after removing $\bigcup_{i=1}^{j-1} S_i$). Since g gives a value of at most c to each element, $|S_j| \geq \frac{g(S_j)}{c} \geq \frac{s_j}{c}$.

Using these facts, we lower bound the number of items in $\biguplus_{i=1}^{2cn/g(A)} S_i$. Observe that $|S_1| \geq \frac{s_1}{c} = \frac{g(A)}{c}$. Since $s_j - s_{j+1} < \frac{g(A)^2}{2cn}$, we have that $s_j \geq g(A)\left(1 - \frac{(j-1)g(A)}{2cn}\right)$, and $|S_j| > \frac{g(A)}{c}\left(1 - \frac{(j-1)g(A)}{2cn}\right)$. The sum of the $\frac{2cn}{g(A)}$ terms is thus larger than $n = |U|$, a contradiction. ■

Lemma 15 *Given an NH function* $g : 2^U \to \mathbb{N}$, *let* $\tilde{g}$ *be the function obtained from* g *by keeping the high order part of* g, *and rounding down each value in the linear part to the closest power of 2. Then* $\max_{S \subseteq U}[\tilde{g}(S)] \geq \frac{1}{4}\max_{T \subseteq U}[g(T)]$.

Proof. Let T be the optimal set for g. Select $S \subseteq T$ by including each item of T in S independently with probability $1/2$. We show that in expectation (all expectations taken over choice of S), $E[\tilde{g}(S)] \geq \frac{1}{4}[g(T)]$, and hence there must be an S satisfying the lemma.

Observe that $E[L_{\tilde{g}}(S)] \geq \frac{1}{2}E[L_g(S)] = \frac{1}{4}L_g(T)$, where the inequality is because rounding down loses at most a factor of 2, and the equality is because each item is included with probability 1/2. Observe also that $E[H_g(S)] \geq \frac{1}{4}H_g(T)$, because every hyperedge (of size at least 2) in the hyperedge representation of g is negative, and is included into S (if included in T) with probability at most $1/4$. Hence $E[\tilde{g}(S)] = E[L_{\tilde{g}}(S)] + E[H_{\tilde{g}}(S)] = E[L_{\tilde{g}}(S)] + E[H_g(S)] \geq \frac{1}{4}(L_g(T) + H_g(T)) = \frac{g(T)}{4}$ ■

A consequence of Lemma 15 is that given a query $\bar{q}$, we may round down each entry of $\bar{q}$ to the nearest power of 2, and lose at most a factor of 4 in the value of the optimal set. Hence we assume from now on that all entries in $\bar{q}$ are powers of 2. In particular, we can update the definition of T_k to $T_k = \left\{x \in U | q_x = 2^k\right\}$.

We use $\tilde{f}$ to denote $H_f + \bar{q}$. Recall that $\tilde{y}$ is the highest valued item. Let $\tilde{w} = q_{\tilde{y}}$. If $\tilde{w}$ is a $4\sqrt{n}$ approximation for the value of the optimal set for $\tilde{f}$, then by returning $\{\tilde{y}\}$ the theorem is proved.

Otherwise, let $A \subseteq U$ be the optimal set for $\tilde{f}$, with $\tilde{f}(A) \geq 4\tilde{w}\sqrt{n}$. Let r be such that $\tilde{w} = 2^r$. Hence, $r \geq k$ for any non-empty T_k. Using the subadditivity of $\tilde{f}$, there is a $k \leq r$ such that $\tilde{f}\left(A \bigcap T_k\right) \geq \tilde{f}(A)/2^{(r-k)/2+2}$ (otherwise, $\tilde{f}(A) \leq \sum_{k \leq r} \tilde{f}\left(A \bigcap T_k\right) < \tilde{f}(A) \sum_{k \leq r} 2^{(k-r)/2-2} < \tilde{f}(A)$). For this k Lemma 14 guarantees an i such that $\tilde{f}\left(A \bigcap T_k \bigcap S_i^k\right) \geq \tilde{f}\left(A \bigcap T_k\right)^2 / 2^{k+1}n \geq \tilde{f}(A)^2 / 2^{r+5}n$. Since S_i^k is maximal, using the decreasing marginal cost definition for submodular functions, $\tilde{f}\left(T_k \bigcap S_i^k\right) \geq \tilde{f}\left(A \bigcap T_k \bigcap S_i^k\right) \geq \tilde{f}(A)^2 / 2^{r+5}n \geq \tilde{f}(A)/8\sqrt{n}$. ■

4 Approximately Answering Submodular Demand Queries

In this section we prove Theorem 3. The proof of Theorem 13 does not apply to some submodular valuation functions, because Lemma 15 need not hold. Consider for example a submodular function g with $g(A) = |A| + 2$ for every nonempty A. Its maximum value is $n + 2$. The rounded down version of it rounds down the linear part from 3 to 2, giving $\tilde{g}(A) = |A| + 2 - |A| = 2$, and the maximum drops to 2. Our solution is to work at a finer scale than powers of 2. A factor of 2 is broken to $n^{1/4}$ intermediate scales, and this will cost another factor of $n^{1/4}$ (beyond $n^{1/2}$) in the approximation ratio.

Preprocessing. Given a submodular valuation function $f : 2^U \rightarrow \mathbb{N}$ that is M-bounded and a nonnegative integer k, define the functions $f_k(A) = \left(1 + n^{-1/4}\right)^k |A| + H_f(A)$. Namely, f_k makes the linear part equal to $\left(1 + n^{-1/4}\right)^k$ for every item.

Denote the greedy optimal sets for f_k by $\left\{S_i^k\right\}$. These sets, for all nonnegative integer k satisfying $\left(1 + n^{-1/4}\right)^k \leq M$, will be referred to as the *advice sets*.

Answering a demand query. Given a query vector $\bar{q}$, let $\tilde{y} = \mathrm{argmax}_x \{q_x\}$ be the highest valued item in U according to $\bar{q}$.
Let $T_k = \left\{x \in U | \left(1 + n^{-1/4}\right)^k \leq q_x < \left(1 + n^{-1/4}\right)^{k+1}\right\}$.

Answer with the highest valued set according to $H_f + \bar{q}$ from $\left\{\{\tilde{y}\}, T_k \bigcap S_i^k\right\}$.

We now prove Theorem 3 by showing that the answer to the query is an $\mathrm{O}\left(n^{3/4}\right)$ approximation to the optimal set with respect to $H_f + \bar{q}$.

Proof. Define $\tilde{f}_1(C) = \sum_{\{x\} \subseteq C} \rho(q_x)$, where $\rho(0) = 0$ and $\rho(z)$ rounds positive integer z down to the nearest power of $1 + n^{-1/4}$. Let $\tilde{f} = \tilde{f}_1 + H_f$ and $\tilde{w} = w_{\tilde{f}}(\{\tilde{y}\})$. If $\{\tilde{y}\}$ is a $8n^{3/4}$ approximation to $H_f + \bar{q}$, we are done. Otherwise, there is a set A such that $f(A) \geq 8n^{3/4}\tilde{w}$.

Proposition 16 *If $f(A) \geq 2n^{3/4}\tilde{w}$, then $\tilde{f}(A) \geq f(A)/2$.*

Proof. Note that $H_f(A) \leq 0$, otherwise f is not submodular. Let α be such that $H_f(A) = -(1-\alpha)L_f(A)$. Then $f(A) = \alpha L_f(A)$. $L_f(A) \leq n\tilde{w}$, so $\alpha \geq 2n^{-1/4}$. $\left(1+n^{-1/4}\right)\tilde{f}_1(A) \geq L_f(A)$, so $\tilde{f}(A) > \left(1-n^{-1/4}\right)L_f(A) - (1-\alpha)L_f(A) \geq (1-\alpha/2)L_f(A) - (1-\alpha)L_f(A) = f(A)/2$. ■

Since $\tilde{f} \leq f$, we only lose a factor 2 when approximating $\tilde{f}$ instead of f.

Let A be the set of maximum value. $\tilde{f}(A) \geq 4\tilde{w}n^{3/4}$. Recall that $T_k = \left\{x \in S | w_{\tilde{f}}(\{x\}) = \left(1+n^{-1/4}\right)^k\right\}$. Let r be such that $\left(1+n^{-1/4}\right)^r = \tilde{w}$. $r \geq k$ for any non-empty T_k.

Using subadditivity, $\tilde{f}(A \bigcap T_k) \geq \frac{1}{4}n^{-1/4}\tilde{f}(A)\left(1+n^{-1/4}\right)^{(k-r)/2}$ for some k (otherwise, $\tilde{f}(A) \leq \sum_{k\leq r}\tilde{f}(A\bigcap T_k) < \frac{1}{4}n^{-1/4}\tilde{f}(A)\sum_{k\leq r}\left(1+n^{-1/4}\right)^{(k-r)/2} < \frac{1}{2}n^{-1/4}\tilde{f}(A)\sum_{k\leq r}\left(1+n^{-1/4}\right)^{(k-r)} < \tilde{f}(A)$). The value of $A\bigcap T_k$ approximates the value of A within $4n^{1/4}\left(1+n^{-1/4}\right)^{(k-r)/2}$. Using Lemma 14, $\tilde{f}\left(A\bigcap T_k \bigcap S_i^k\right) \geq \tilde{f}(A\bigcap T_k)^2\left(1+n^{-1/4}\right)^{-k}/2n \geq \tilde{f}(A)^2\left(1+n^{-1/4}\right)^{-r}/32n^{3/2} \geq \tilde{f}(A)/8n^{3/4}$ for some i. Therefore, the value of $A\bigcap T_k\bigcap S_i^k$ is a $8n^{3/4}$ approximation to the value of A. Since S_i^k is maximal, the marginal value of every item is non-negative (otherwise, the set without this item has higher value). Using the decreasing marginal value definition for submodular functions, every item has non-negative marginal value for every subset of S_i^k. Hence, $\tilde{f}\left(T_k\bigcap S_i^k\right) \geq \tilde{f}\left(A\bigcap T_k\bigcap S_i^k\right)$. This proves that the answer to the query is a set of higher or equal value to a set that is an $\mathrm{O}\left(n^{3/4}\right)$ approximation to the optimal set with respect to $H_f + \bar{q}$. ■

5 Hardness for Approximately Answering MIS Queries

In this section we revisit the setting of MIS queries introduced in Section 2. Given an input graph $G(V, E)$, one is allowed to preprocess this graph for arbitrary time and record a polynomial size advice string $A(G)$. Thereafter a subset $U \subset V$ is given as a query, and one is required to approximate the maximum independent set in the subgraph induced on U, and do so in polynomial time. We shall show that for some $\delta > 0$, even after preprocessing, MIS queries cannot be approximately answered within a ratio better than $\Omega(n^\delta)$, unless $NP \subset P/poly$. Given that MIS queries are a special case of demand queries with respect to a submodular function, this will thus prove Theorem 4.

We shall use Theorem 17, taken from [8]. Recall the problem max-3SAT: given a 3CNF formula with n variables and m clauses (each containing three literals), find an assignment that satisfies the maximum number of clauses. A *factor graph* is a template for a 3CNF formula that specifies the variables in each clause, but leaves the polarities of the variables unspecified. A family of factor graphs includes for each sufficiently large value of n one template. (The need to consider an infinite family is a complexity theory technicality. The reader may fix one n of interest and think of just one factor graph in this case.) Given a factor graph, a *3SAT query* is an assignment of polarities to the occurrences of variables in the template, and one is asked to solve the resulting max-3SAT instance in polynomial time. Preprocessing the factor graph before receiving the query is allowed.

Theorem 17 *For some $\rho < 1$, there is a family of factor graphs such that even after preprocessing, one cannot distinguish between satisfiable 3SAT queries, and those that are at most ρ-satisfiable, unless $NP \subset P/poly$.*

The value of ρ in Theorem 17 can be taken to be roughly $77/80$. A factor graph from the family of Theorem 17 is referred to as a *universal factor graph* (UFG). Proposition 18 is a first step towards proving Theorem 4.

Proposition 18 *For some $\rho < 1$, even after preprocessing, MIS queries cannot be answered with an approximation ratio better than $1/\rho$, unless $NP \subset P/poly$.*

Proof. Given a UFG F that is a template for 3CNF formulas with n variables and m clauses, use the following variation on the FGLSS reduction [7] to obtain a graph $G(V, E)$ on $8m$ vertices. Every clause $v \in F$ is associated with a cluster of eight vertices $v_1, \cdots, v_8 \in V$, one for each possible assignment to the three variables in the clause. There is an edge in E between two vertices iff their corresponding assignments disagree on some variable.

A 3SAT query to F can be cast as a MIS query to G. Setting the polarities to a clause v of F is equivalent to discarding the unique member of v's cluster in G that corresponds to an assignment not satisfying the clause, and keeping the remaining vertices of the cluster. The MIS query is the set U of vertices that remains. The size of the maximum independent set in the subgraph $U(G)$ induced on U is exactly the maximum number of satisfiable clauses in the 3SAT query. The proposition now follows from Theorem 17. ■

Before we continue, we review the notion of *derandomized graph products* [1]. For a desired value of ϵ, we say that a d-regular graph is an *ϵ-expander* if the eigenvalues $d = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ of its adjacency matrix satisfy $\max[|\lambda_2|, |\lambda_n|] \leq \epsilon\lambda_1$. It is known that for every $\epsilon > 0$ there is sufficiently large d (one would need $d > \Omega(1/\epsilon^2)$), such that there are ϵ-expanders of size n for all sufficiently large n. Moreover, it is known how to construct such ϵ-expanders.

Given a graph $G(V, E)$ its derandomized graph product $DG^k = (U, E_U)$ uses an arbitrary auxiliary d-regular ϵ-expander $F(V, E_F)$ defined on the same set of vertices V (and a set E_F edges unrelated to E). U consists of all walks with $k - 1$ steps in F (hence $|U| = |V|d^{k-1}$), and there is an edge in DG^k between $(v_1, \cdots, v_k)$ and $(u_1, \cdots, u_k)$ iff there are i, j such that $(v_i, u_j) \in E$. Let $\alpha(G)$ denote the size of the maximum independent set in graph G. The following theorem is from [1].

Theorem 19 *For derandomized graph products as defined above and using an ϵ-expander with $\epsilon < \frac{\alpha(G)}{|V|}$:*

$$\alpha(G)d^{k-1}\left(\frac{\alpha(G)}{|V|} - \epsilon\right)^{k-1} \leq \alpha(DG^k) \leq \alpha(G)d^{k-1}\left(\frac{\alpha(G)}{|V|} + \epsilon\right)^{k-1}$$

In order to use Theorem 19 it will be convenient for us to use a special class of expanders. Given a graph $F = (V, E)$ on m vertices and a positive integer k, define the graph $F_k = (V \times k, E_k)$ as follows: every vertex of F is replaced by an independent set of size k, and every edge of F is replaced by a complete bipartite graph between the corresponding independent sets.

Proposition 20 *If F is an ϵ-expander, then so is F_k for every k.*

Proof. The adjacency matrix of F_k is a tensor product of two matrices: the adjacency matrix of F, and a k by k all 1 matrix (whose eigenvalues are k and 0 with multiplicity $k - 1$). The eigenvalues of the tensor are all products of the eigenvalues of its factors. ■

We can now prove Theorem 4.

Proof. Let $G(V, E)$ be a graph with $8m$ vertices, the outcome of Proposition 18. Recall that its vertices are arranged in m clusters of size 8. Pick ϵ sufficiently small (e.g., $\epsilon = \frac{1-\rho}{20}$ for ρ as in Proposition 18). Let F be an arbitrary d-regular ϵ-expander on m vertices. F_8 is an $8d$-regular graph on $8m$ vertices. Match vertices of G with those in F_8, with each cluster of G mapped to a cluster of F_8. For $k = \Theta(\log m)$, consider the derandomized graph product DG^k with respect to F_8. The number of its vertices is polynomial in m.

Given a MIS query U to $G(V, E)$ for which one wants to distinguish between the case that the optimal answer is m and the case that it is at most ρm, transform it into a MIS query U' to DG^k, where a vertex of DG^k belongs to U' iff the k vertices of that walk that it corresponds to are all in U. Observe that the set of vertices in U' is precisely what one would get by taking a k-fold derandomized graph product of the subgraph $U(G)$ with respect to F_7 rather than F_8. Proposition 20 implies that F_7 is an ϵ-expander. Theorem 19 (and some straightforward calculations that are omitted) implies that the ratio between the m versus ρm cases has been amplified to some polynomial N^δ, where N is the total number of vertices in DG^k, and $\delta > 0$. ■

Acknowledgements

Work supported in part by the Israel Science Foundation (grant No. 621/12) and by the I-CORE Program of the Planning and Budgeting Committee and The Israel Science Foundation (grant No. 4/11). We thank an anonymous ICALP reviewer for detailed comments.

References

[1] Alon, N., Feige, U., Wigderson, A., Zuckerman, D.: Derandomized graph products. Computational Complexity 5(1), 60–75 (1995)

[2] Bruck, J., Naor, M.: The hardness of decoding linear codes with preprocessing. Information Theory, IEEE Transactions on 36(2), 381 –385 (1990)

[3] Buchbinder, N., Feldman, M., Naor, J., Schwartz, R.: A tight linear time (1/2)-approximation for unconstrained submodular maximization. In: FOCS. pp. 649–658 (2012)

[4] Cunningham, W.H.: On submodular function minimization. Combinatorica 5(3), 185–192 (1985)

[5] Dobzinski, S., Nisan, N., Schapira, M.: Approximation algorithms for combinatorial auctions with complement-free bidders. In: STOC. pp. 610–618 (2005)

[6] Dobzinski, S., Schapira, M.: An improved approximation algorithm for combinatorial auctions with submodular bidders. In: SODA. pp. 1064–1073 (2006)

[7] Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Interactive proofs and the hardness of approximating cliques. J. ACM 43(2), 268–292 (1996)

[8] Feige, U., Jozeph, S.: Universal factor graphs. In: ICALP. pp. 339–350 (2012)

[9] Feige, U., Micciancio, D.: The inapproximability of lattice and coding problems with preprocessing. J. Comput. Syst. Sci. 69(1), 45–67 (2004)

[10] Feige, U., Vondrák, J.: The submodular welfare problem with demand queries. Theory of Computing 6(1), 247–290 (2010)

[11] Goemans, M.X., Harvey, N.J.A., Iwata, S., Mirrokni, V.S.: Approximating submodular functions everywhere. In: SODA. pp. 535–544 (2009)

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

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

[14] Khot, S., Popat, P., Vishnoi, N.K.: $2^{\log^{1-\epsilon} n}$ hardness for the closest vector problem with preprocessing. In: STOC. pp. 277–288 (2012)

[15] Lehmann, B., Lehmann, D.J., Nisan, N.: Combinatorial auctions with decreasing marginal utilities. Games and Economic Behavior 55(2), 270–296 (2006)

[16] Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. In: STOC. pp. 681–690 (2006)