Optimization with uniform size queries

Uriel Feige* Moshe Tennenholtz†

May 7, 2015

Abstract

Consider the problem of selecting k items that maximize the value of a monotone submodular set function f, where f can be accessed using value queries. It is well known that polynomially many queries suffice in order to obtain an approximation ratio of $1 - \frac{1}{e}$. We consider a variation on this problem in which the value queries are required to be of uniform size: each queried set, like the desired solution itself, must contain k items. We show that polynomially many uniform size queries suffice in order to obtain an approximation ratio of $\frac{1}{2}$, and that an approximation ratio of $\frac{1+\epsilon}{2}$ requires a number of queries that is exponential in ϵk. For the interesting special case of coverage functions, we show that an approximation ratio strictly better than $\frac{1}{2}$ is attainable with polynomially many uniform size queries.

The "uniform size" requirement is motivated by situations in which a firm may offer a menu of exactly k items to its clients, where k is a parameter determined by external considerations. Performing a query corresponds to physically changing the identities of the items offered, and the reply to the query is deduced by observing the behavior of clients in response to the change. Queries that involve a number of items that differs from k may not be desirable due to these external considerations. In such situations it is natural to ask whether the same approximation ratios that can be guaranteed by general value queries can also be obtained by uniform size queries.

1 Introduction

In this work we study a variation on a well known optimization problem, where our variation is motivated by economic scenarios. The problem is introduced in Section 1.1, whereas some motivation is provided in Section 1.2.

*Department of Computer Science and Applied Mathematics, Weizmann Institute, Rehovot. uriel.feige@weizmann.ac.il.

†Technion–Israel Institute of Technology. moshet@ie.technion.ac.il

1.1 Max k-submodular

We first recall some standard definitions. Let U be a universe of n items, and let $f : U \longrightarrow R_{\geq 0}$ be a nonnegative set function, namely, a function that assigns to each subset $S \subset U$ a nonnegative value $f(S) \geq 0$. Function f is *monotone* if $f(S) \geq f(T)$ whenever $T \subset S$, and *submodular* if $f(S)+f(T) \geq f(S \cup T)+f(S \cap T)$ for all $S, T \subset U$. Equivalently, f is submodular if items have decreasing marginal values in the sense that for every sets $T \subset S \subset U$ and every item $i \notin S$, $f(S \cup \{i\}) - f(S) \leq f(T \cup \{i\}) - f(T)$. A *value query* to f is a set $S \subset U$, and its reply is the value $f(S)$.

The computational problem that we study is the following: given U, a nonnegative monotone submodular f as above that can only be accessed using value queries, and a positive integer parameter k, output a set $S \subset U$ of cardinality $|S| = k$ that maximizes $f(S)$, namely $f(S) = \max_{|T|=k} f(T)$. We call this problem *max k-submodular*, and denote its optimal value by opt.

The operations available for an algorithm that attempts to solve max k-submodular are value queries to f, and polynomial time computations based on the results of these queries. Hence the number of queries is also required to be polynomial (in n and k).

A ρ-approximation algorithm is an algorithm that is allowed to make only a polynomial number of queries (polynomial in n and k), and returns a set T of cardinality k whose value satisfies $f(T) \geq \rho$ opt. For randomized approximation algorithms, we require the expected value of their output to be at least ρ opt.

The problem max k-submodular as described above is well studied. It is known that a greedy algorithm provides a $1 - \frac{1}{e} \simeq 0.632$ approximation ratio, and that this is best possible [14, 7].

In this work we modify the above problem by assuming that not all queries cost the same. Namely, there is a known nonnegative cost function $c : U \longrightarrow R$ associated with queries, and the complexity of an algorithm is measured in terms of sum of costs of all queries that it makes. Specifically, we shall assume that $c(S) = 1$ whenever $|S| = k$, and $c(S)$ is prohibitively large if $|S| \neq k$ (e.g., $c(S) = 2^n$ in this case). Hence approximation algorithms will need to use only value queries for sets of cardinality exactly k, and are forbidden from asking value queries for sets of other sizes.

1.2 Examples

Suppose that a firm can offer a menu with k items, where k is a parameter governed by some external limitations (such as physical size of menu, or span of attention of costumers). For example: the firm might be a restaurant, and the items are dishes on its menu; or the firm might be a physical shop and the menu includes the items in the shop window; or the firm might be a web service provider, and the items can be entries on its home page; or the firm can be a provider of a personal assistant (say, on a mobile phone), which may provide potential customers with k different persona in the menu list (e.g. one that optimizes social value, one that optimizes monetary value, etc.). Suppose further that each potential client desires some subset of items, and will find a menu satisfactory if and only if the menu contains at least one desired item. The firm wishes to produce a menu that will satisfy as many clients as possible. One can readily observe that the function f assigning to each menu the number of satisfied costumers is a *coverage function*, and hence the problem the firm is faced with is max k-coverage, which is a special case of max k-submodular. In some other embodiment, where e.g. the customer either likes or dislikes the menu as a whole, the percentage of satisfied customers for the menu determines the value of the menu, and often can be modeled by a sub-modular function that is not a coverage function.

The question then becomes how can the firm "learn" the function f (learn what clients want). A reasonable approach is to do so by an exploration procedure. In a single exploration step, the firm offers an "experimental" menu and estimates based on observed behavior what fraction of costumers are satisfied by the menu. For example, a client who visited the web page and chose to click on some link is counted as satisfied and a client who left the web page without following any of the links is counted as not satisfied. Hence, such an exploration step, when conducted along a period of time, can serve as an approximate value query (assuming that the set of clients who were exposed to the menu is random). Thereafter, the firm can change the experimental menu and perform an additional exploration step, thus performing (in effect, and subject to some approximation error) a sequence of value queries.

Why would different value queries (experimental menus) have different costs? There can be many reasons. Here is one of them. Suppose that the firm is already offering a menu with k items, and a consultant to the firm proposes to experiment with other menus so as to optimize the contents of the menu. The firm might be willing to give the consultant a chance to come up with an improved menu, as long as the exploratory phase is not

too disruptive to its current business. Temporarily replacing a k item menu by a much smaller menu might be regarded as being very disruptive (and hence of high cost), whereas replacing just one item on the existing menu may be regarded as not disruptive, and hence has low cost. Another reason is due to competition. Assume that a firm can afford the space/capability of offering k items, matching the number of items offered by a competitor. Such a firm might not like to ever offer on its menu a number of items that is smaller than the number offered by the competitor.

Notice that the above examples refer to the value obtained as statistics about an experimental period. In the coverage problem the feedback discussed is whether a customer likes at least one of the items. This brings the question of whether we (as optimizers) see the item actually chosen (if any) by a given customer. This issue is discussed in Section 3.2.

The purpose of the above examples is mainly to illustrate that there are natural settings in which different value queries have different costs. This and similar settings motivate our definition of max k-submodular with query costs. Needless to say, our problem is a mathematical abstraction rather than a detailed model of reality. Some aspects that may be relevant in practice are discussed in Section 3.1, whereas others are completely ignored in this manuscript.

1.3 Results

Throughout this section, the problem max-k-submodular refers to the setting in which only value queries for sets of size k are allowed. We refer to queries of this form as k-queries.

In Section 2.1 we present a simple local search algorithm, similar to the *interchange heuristic* of [14] (the difference is that we explicitly introduce a parameter ϵ so as to bound the running time). The following theorem provides performance guarantees for that algorithm (which are basically the same as in [14]).

Theorem 1 *For every $\epsilon > 0$, the local search algorithm with parameter ϵ approximates max-k-submodular within a ratio of at least $\frac{k}{2k-1+\epsilon}$, and uses at most $O(\frac{nk^3 \log k}{\epsilon})$ k-queries.*

The approximation bounds provided in Theorem 1 are best possible for the local search algorithm, as shown by the following proposition. (Similar negative results appear also in [14].)

Proposition 2 *There are instances of max k-coverage (which is a special case of max k-submodular) on which the local search algorithm might stop at a solution of value no better than a* $\frac{k}{2k-1}$ *fraction of the optimum solution, regardless of the value of* ϵ*.*

More generally, no algorithm can offer substantial improvement over the approximation ratio of $\frac{1}{2}$.

Theorem 3 *For* $\epsilon \leq \frac{1}{2}$*, any algorithm that has probability at least* $\frac{1}{2}$ *of approximating max-k-submodular within a ratio of* $\frac{1+\epsilon}{2}$ *must use at least* $\Omega\left(\left(\frac{\epsilon n}{ek}\right)^{\epsilon k}\right)$ *k-queries.*

A natural and well studied subclass of submodular functions is that of coverage functions. In fact, the example presented in Section 1.2 refers to a coverage function. The negative example provided in Theorem 3 involves a submodular function that is not a coverage function. Indeed, we can show that for max k-coverage an approximation ratio strictly better than $\frac{1}{2}$ is achievable using polynomially many k-queries. The algorithm that we design for this purpose will be referred to as the *conditional greedy* algorithm.

Theorem 4 *There is some* $\rho_{CG} > \frac{1}{2}$*, such that for every k, the conditional greedy (randomized) algorithm approximates max-k-coverage within a ratio of at least* ρ_{GD}*, and uses at most polynomially many k-queries.*

We do not attempt to compute ρ_{CG} exactly. We only show that $0.50004 < \rho_{CG} < 0.582$ (for the upper bound see Proposition 8). The lower bound suffices in order to illustrate that max k-coverage can be approximated within a strictly better ratio than max k-submodular in our context of k-queries. The upper bound establishes that the conditional greedy algorithm does not suffice for attaining an approximation ratio of $1 - \frac{1}{e}$.

1.4 Related work

Accessing a function via value queries is a common theme in combinatorial optimization. In some work (e.g., [10]) the query model is used so as to present general algorithmic results that hold for every function within some class (e.g., minimization of any submodular function). In algorithmic game theory settings , the query model is sometimes used in order to overcome communication bottlenecks. For example, in the maximum welfare problem (e.g., [13]), bidders are not assumed to communicate their valuation functions (whose description might be exponential in the number of items), but

rather to answer value queries with respect to their valuation functions. For some classes of valuation functions value queries appear to be too weak (so as to provide a reasonable approximation ratio for the problem at hand), and one resorts to stronger classes of queries, such as demand queries [3]. In our context we think of the queries as representing the outcome of a physical experiment. This is a common view in multiple settings, ranging from settings quite similar to ours (see for example [15] and references therein), to very different settings (e.g., a clinical trial can be thought of as a physical experiment whose purpose it to answer a query). Under this view, it is natural to assign different costs to different queries. Often, they are physical costs of the experiment. However, we think of them here as economic costs associated with exploration, in terms of short term loss in revenue due to the exploration step, or long term loss in reputation or client basis due to what appears to an outside observer to be erratic behavior of the firm.

Submodular function maximization under a cardinality constraint (max k-submodular) is a well studied problem. A simple greedy algorithm achieves an approximation ratio of $1 - \frac{1}{e}$ using $O(nk)$ value queries [14]. As noted in Section 1.3, an approximation algorithm that is only based on k-queries (referred to as an *interchange heuristic*) also appears in [14], though we remark that there it is introduced not so as to avoid value queries to sets of size different than k (as we do in our manuscript), but simply because it is a natural heuristic to study.

Max k-coverage is an extensively studied special case of max k-submodular. The reader is alerted that the terminology used in this special case is somewhat inconsistent with the terminology for the general case: for max k-coverage each member of U is referred to as a set of items (or elements), and a solution is referred to as a collection of k sets, and the value of the solution is the number of items covered. When the input instance is given explicitly (as a list of polynomially many sets), the problem can be approximated by a ratio of $1 - 1/e$ either using the greedy algorithm, or using rounding of a linear programming relaxation (both algorithms can be found as exercises in [19]), and obtaining an approximation ratio better than $1-1/e$ is NP-hard [7].

Restricting queries to values for sets of size exactly k is a theme that appears also in the context of linearity testing, when the input is the kth slice of a supposedly multivariable linear function. See [5] for more details.

Settings such as those described in Section 1.2 refer to a most basic challenge of fixed menu selection. While menu selection has been discussed explicitly and experimentally in the Human Computer Interaction (HCI) community (see e.g. [1] and the references within), and is implicit in methods

for vector and feature selection in Machine Learning (see e.g. [11] for an overview), the most directly relevant work is in the context of contract menus: menus that are offered in order to optimize a firm/social value where a buyer selects one of the proposed contracts or none. Indeed, the topic of finding a fixed size simple optimized contract has been discussed (see e.g. [17]) following the seminal work of Laffont and Tirole [12]. The notion of searching by value queries of experimental fixed size menus for an optimal menu, and doing it efficiently, nicely complements that perspective.

The special case of maximizing k-coverage using only k-queries is studied in [15], but there the reply model is different from our paper, and consequently the approximation ratio is $1 - \frac{1}{e}$. See more details in Section 3.2. Extensions of the setting of [15] to more general submodular functions were studied in [18], which also achieves a $1 - \frac{1}{e}$ approximation ratio, but using general value queries (and not just k-queries).

2 Proofs

2.1 Approximating max k-submodular using k-queries

We show that for every ϵ, an approximation ratio of $\frac{k}{2k-1} - \epsilon$ can be obtained with polynomially many k-queries.

We refer to two sets T and T' as neighbors if both sets have the same cardinality ($|T| = |T'|$), and T' can be obtained from T by one swap (namely $|T \backslash T'| = |T' \backslash T| = 1$). A set T will be called a local ϵ-maximum (with respect to function f) if for every neighboring set T' one has $(1 + \frac{\epsilon}{k^2}) f(T) \geq f(T')$. Observe that $k(n-k)$ k-queries suffice in order to check whether T of size k is a local ϵ-maximum, and provide a witness T' if not.

To address the problem max k-submodular, we propose the following algorithm that uses only k-queries.

1. Start with an arbitrary set T with $|T| = k$.
2. If T is a local ϵ-maximum, return T.
3. Else, let T' be the neighbor of T of highest f value. Repeat the algorithm with T' replacing T.

We refer to the above algorithm as the *local search* algorithm. We alert the reader that other local search algorithms have been used in the context of submodular optimization (for example, to maximize nonmonotone submodular functions, see [8]), but they may differ from our algorithm in the way they define the neighborhood of a set.

Lemma 5 *Let f be a nonnegative monotone submodular set function. Let S denote the set of cardinality k of highest $f(S)$. When the local search algorithm ends, $f(T) \geq \frac{k}{2k-1+\epsilon} f(S)$.*

Proof. By submodularity of f, there is an item $x \in T$ such that $f(T \setminus \{x\}) \geq \frac{k-1}{k} f(T)$, and an item $y \in S$ such that $f((T \setminus \{x\}) \cup \{y\}) \geq \frac{k-1}{k} f(T) + \frac{f(S) - \frac{k-1}{k} f(T)}{k} = f(T)(1 - \frac{2}{k} + \frac{1}{k^2}) + \frac{f(S)}{k}$. Let us denote $T \setminus \{x\}) \cup \{y\}$ by T'. Then due to local ϵ-maximality of T we have that $(1 + \frac{\epsilon}{k^2}) f(T) \geq f(T') \geq f(T)(1 - \frac{2}{k} + \frac{1}{k^2}) + \frac{f(S)}{k}$ implying that $\frac{2k-1+\epsilon}{k^2} f(T) \geq \frac{f(S)}{k}$. ■

Lemma 6 *The local search algorithm stops after at most $O(\frac{k^2 \log k}{\epsilon})$ iterations.*

Proof. Let S denote the set of cardinality k of highest $f(S)$. By submodularity, there is an item $x \in S$ with $f(x) \geq f(S)/k$. Moreover, for every set T there is a neighboring set T' that contains x. Furthermore, by monotonicity of f, for this T' we have $f(T') \geq f(x)$. If follows that after one iteration the set T held by the local search algorithm has value satisfying $f(T) \geq f(S)/k$. Thereafter, it takes at most $O(k^2/\epsilon)$ iterations to double the value of $f(T)$. The number of iterations cannot exceed $O(k^2 \log k/\epsilon)$, as otherwise $f(T)$ will exceed $f(S)$. ■

We now prove Theorem 1

Proof. The claim regarding the value of the solution follows from Lemma 5, and the claim regarding the number of k-queries follows from Lemma 6, combined with the fact that a single iteration of the local search algorithm can be implemented using $O(nk)$ k-queries. ■

2.2 Hardness results

We now prove Proposition 2.

Proof. Consider an instance of max k-coverage with $n = k(2k-1)$ items, arranged in a matrix with k rows and $2k-1$ columns. Each row is a set (with $2k-1$ items), and each one of the first k columns is a set (with k items). The optimal solution is composed of the k rows, and has value $k(2k-1)$. The collection of first k columns is a locally optimal solution (which the local search algorithm may start with and then immediately end, regardless of the value of ϵ), and its value is k^2. The ratio between the two is $\frac{k}{2k-1}$. ■

Proof of Theorem 3.

Proof. For ϵ in the range $\frac{1}{k} \leq \epsilon < 1$, let $\rho = \frac{1+\epsilon}{2}$. For simplicity, assume that ρ divides k. Consider the following instance. There are k *good* items and $n-k$ *bad* items. For every $0 \leq g \leq k$ and $0 \leq b \leq n-k$ and any set T that contains g good items and b bad items, we define $f(T) = \min\left[k, \rho b + g \max[\frac{k-\rho b}{k}, \rho]\right]$.

Write f as $f = \min[k, f']$ with $f'(T) = \rho b + g \max[\frac{k-\rho b}{k}, \rho]$. Monotonicity and submodularity of f are inherited from f', because they are preserved under taking a minimum with a nonnegative constant (k, in our case).

To see that f' is monotone, one easily observes that adding a good item to a set cannot decrease its value. Likewise, adding a bad item to a set increases the contribution of the bad items by ρ, and decreases the contribution of each good item by at most $\frac{\rho}{k}$. As there are at most k good items, there cannot be a net decrease.

Observe that when $b \geq k/\rho$ the value of f is k, and then the marginal value of each additional item is 0. Hence to show that f is submodular, it suffices to show that f' is submodular when its domain is restricted to have k/ρ bad items (and k good items). For a set T, the marginal value of an additional good item depends only on the number of bad items in T, and cannot increase when the number of bad items grows. The marginal value of an additional bad item is $\rho - g\frac{\rho}{k}$ (when $b < k/\rho$, and 0 otherwise) which is independent of b and decreasing in g. Hence f' is submodular over its restricted domain.

Having established that f is nonnegative, monotone and submodular, we consider the number of k-queries required in order to approximate max k-submodular well. For this we suppose that the items are permuted at random so that a-priori, the algorithm cannot distinguish between good and bad items.

The set of k good items has value k, and this is optimal.

Every set of k items that contains at least $\frac{1-\rho}{\rho}k = \frac{1-\epsilon}{1+\epsilon}k$ bad items has value ρk. We say that a q-query is *informative* if it returns a value larger than ρk. The fraction of k-queries that are informative is precisely the probability that a random set of k items contains more than $\frac{2\epsilon}{1+\epsilon}k$ good items. For notational convenience, denote $\delta = \frac{2\epsilon}{1+\epsilon}$. Then this probability is at most:

$$\binom{k}{\delta k}\left(\frac{k}{n-k}\right)^{\delta k} \simeq \left(\frac{ek}{\delta(n-k)}\right)^{\delta k} \leq \left(\frac{ek}{\epsilon n}\right)^{\epsilon k}$$

where the last inequality holds when $\frac{2\epsilon(n-k)}{1+\epsilon} \geq \epsilon n$, or equivalently, when

$k \leq \frac{1-\epsilon}{2}n$. This last inequality can indeed be assumed, because if it does not hold (and $\epsilon \leq \frac{1}{2}$ as in the statement of the theorem), then $\frac{\epsilon n}{ek} \leq 1$ and there is nothing to prove in the theorem.

Hence for a random permutation over items, the probability that at least one of the first $\frac{1}{2}(\frac{\epsilon n}{ek})^{\epsilon k}$ k-queries is informative is at most $\frac{1}{2}$. As we may assume without loss of generality that the last query of an algorithm is its final output (once an algorithm decides on its output, it can always ask it as a query as well), an algorithm with fewer than $\frac{1}{2}(\frac{\epsilon n}{ek})^{\epsilon k}$ k-queries has probability at most $\frac{1}{2}$ of obtaining an approximation better than $\rho = \frac{1+\epsilon}{2}$. ■

2.3 An algorithm for max k-coverage

In this section we consider max k-coverage, which is a common special case of max k-submodular. For max k-coverage we shall use here terminology and notation that is consistent with that used for max k-submodular, even though this notation is nonstandard for coverage functions.

Let U be a universe of n items, and let $f : U \longrightarrow R$ be a nonnegative set function, namely, a function that assigns to each subset $S \subset U$ a nonnegative value $f(S) \geq 0$. For f to be a coverage function, there should be a set M of m elements (these elements need not be physical elements, but rather a mental aid to the way coverage functions are defined), and a collection of n subsets $S_i \subset M$, for $1 \leq i \leq n$. Item $i \in U$ corresponds to set S_i (item i *covers* the elements of set S_i), and its value under f is the number of elements that S_i covers. Namely, $f(i) = |S_i|$. More generally, given a set $S \subset U$ of items, its value is the number of elements covered by the respective sets, namely $f(S) = |\bigcup_{i \in S} S_i|$. A coverage function as defined above is always monotone and submodular.

Max k-coverage is the following computational problem: given U, a coverage function f as above, and a positive integer parameter k, output a set $S \subset U$ of cardinality $|S| = k$ that maximizes $f(S)$, namely $f(S) = \max_{|T|=k} f(T)$. We denote its optimal value by opt.

In our setting f can only be accessed using k-queries. Namely, a query is a set $S \subset U$ with $|S| = k$, and the answer to the query is $f(S)$. We emphasize that the set M of elements is not explicitly accessible via queries. In particular, the number m of elements is not given as part of the input to the problem, and it need not be related to the number n of items (e.g., m might be exponential in n). Our goal is to approximate max k-coverage using a number of queries that is polynomial in n and k. We propose an algorithm,

that is referred to as *conditional greedy*, that achieves an approximation strictly greater than 1/2.

To describe the algorithm, let as introduce some notation. Given an item $i \in U$, a set $T \subset U$ with $i \notin T$ and $|T| \leq k-1$, we let $f_k(i|T)$ denote the expected value of a random set S of cardinality k, conditioned on S containing T and i (the other items are chosen uniformly at random from $U \setminus (T \cup \{i\})$). Namely, $f_k(i|T) = E_{S|(T\cup\{i\})\subset S} f(S)$.

Given i and T, our algorithm would like to query the value of $f_k(i|T)$. We refer to this as a *conditional query*. Our query model does not allow one to ask these conditional queries directly. However, it is possible to implement an approximate conditional query (up to any desired level of accuracy) using k-queries.

Proposition 7 *Given a desired accuracy parameter $\eta > 0$ and tolerable failure probability $\delta > 0$, and given i and T with $i \notin T$ and $|T| \leq k-1$, the conditional query $f_k(i|T)$ can be simulated by making $\frac{\log \frac{1}{\delta}}{\eta^2}$ k-queries. With probability $1-\delta$, the result one obtains will approximate $f_k(i|T)$ within an additive error of $O(\eta \text{ opt})$.*

Proof. Generate $\frac{\log \frac{1}{\delta}}{\eta^2}$ sets S of size k, where each such set S contains i, T, and $k-1-|T|$ additional distinct items sampled without repetition independently at random. Perform one k-query for each such set S, and return the average the replies as the reply to the conditional query $f_k(i|T)$. As the value of each k-query is bounded between 0 and opt, the proposition follows from standard application of upper bounds (such as the Chernoff bound) on the probability of large deviations from the average. ■

To simplify the presentation, we shall describe the algorithm as if it performs conditional queries. Proposition 7 implies that such an algorithm can be implemented using only k-queries. For the implementation to provide guarantees that are within additive ϵopt of the guarantees provided by true conditional queries, it suffices to set $\eta = \frac{\epsilon}{k}$ (and then the accumulated error throughout the algorithm is at most $\eta k \leq \epsilon$), and $\delta = n^{-O(1)}$ (so that a union bound implies that with high probability all polynomially many conditional queries made by the algorithm are accurate within η). We remark that there are ways of improving the efficiently of the implementation (one k-query S can participate in the implementation of $k-|T|$ conditional queries, one for each $i \in (S \setminus T)$; one can adapt the algorithm to run with a smaller value of δ, using principles as in [9]) but these issues will not be discussed in this paper.

The conditional greedy algorithm has k iterations. It enters iteration ℓ for $1 \leq \ell \leq k$ with a set $T_{\ell-1} \subset U$ of cardinality $\ell - 1$, derived from the previous iteration. Initially, T_0 is the empty set. At iteration ℓ, the algorithm adds to $T_{\ell-1}$ the item with highest conditional value, thus deriving the new set T_ℓ.

Below is a more structured description of the **conditional greedy algorithm**.

1. T_0 is the empty set.
2. Repeat for $\ell = 1$ up to k:
   (a) For each $i \in (U \setminus T_{\ell-1})$, perform the conditional query $f_k(i|T_{\ell-1})$. Let $i^* \in (U \setminus T_{\ell-1})$ be the item with highest $f_k(i^*|T_{\ell-1})$, breaking ties arbitrarily.
   (b) Update $T_\ell = T_{\ell-1} \cup \{i^*\}$.
3. Output T_k.

The conditional greedy algorithm can be shown to be equivalent to starting with a distribution over uniformly random solutions to the max k-coverage problem, in which each item is chosen independently with equal probability, and then derandomizing this random solution using the method of conditional expectations [6, 16]. Often, such derandomization is done so as to obtain a single solution whose quality is at least as good as the average quality of solutions in the starting distribution. In our context, we want to achieve more, because there are no guarantees on the average quality of solutions in the starting distribution. Indeed, our analysis will provide guarantees on the the quality of the final solution that do not hold with respect to the initial distribution. A previous example in which the method of conditional expectations offers guarantees better than the original distribution appears in [2] in the context of max directed cut. The results in [4] for max SAT can also be viewed in this way.

For max k-submodular, the conditional greedy algorithm will not give an approximation ratio strictly above $\frac{1}{2}$, as is evident from Theorem 3. However, as we shall see, for max k-coverage it does. Providing a tight analysis of the approximation ratio offered by the conditional greedy algorithm appears difficult, so here we shall limit ourselves to establishing that the approximation ratio, which we denote as ρ_{CG}, is strictly above $\frac{1}{2}$, and strictly below $1 - \frac{1}{e}$ (which is the approximation ratio of the standard greedy algorithm).

We start by presenting an example showing that $\rho_{CG} < 1 - \frac{1}{e}$.

Proposition 8 *There are examples on which the approximation ratio ρ_{CG} achieved by the conditional greedy algorithm is no better than* 0.582*.*

Proof. Consider a universe M of $m = \frac{8}{5}k^2$ elements, arranged in a matrix with k rows and $\frac{8}{5}k$ columns. In the coverage function f, the items 1 to k (that compromise the optimal solution) each covers one row of the matrix. We refer to them as the *optimal* items. Hence opt $= m$. In addition, there are items $k+1$ up to $\frac{8}{5}k$ that each covers the elements in its own numbered column (item i for $k+1 \leq i \leq \frac{8}{5}k$ covers the elements of column i). We refer to them as *decoy* items. Finally, there are $n - \frac{8}{5}k$ additional items, which each cover k random elements in the first k columns of the matrix. We refer to them as the *main* items, and assume that n is some large polynomial in k (and specifically $n > k^2$).

Observe that a random set of $k-1$ items will most likely be composed only of main items. Consequently, it can be shown that the marginal contribution of main items will be roughly $\frac{k}{e}$, the marginal contribution of optimal items will be roughly $\frac{1}{e}k + \frac{3}{5}k < 0.97k$, and the marginal contribution of decoy items will be essentially k. Hence in its first iteration the conditional greedy algorithm will pick a decoy item. Similarly, it can be shown that in the first $\frac{3k}{5}$ rounds, only decoy items will be chosen, exhausting the list of decoy items. Regardless of what the algorithm picks in the remaining rounds, the total number of elements covered will be at most k^2, giving an approximation ratio of $\frac{5}{8} < 1 - \frac{1}{e}$. In fact, in the last $\frac{2k}{5}$ rounds the algorithm is likely to select main items rather than optimal ones (their marginal contributions are similar, but there are many more main items than decoy ones), and these items will cover only roughly $k^2\left(1 - (1 - \frac{1}{k})^{\frac{2k}{5}}\right) \simeq 0.33k^2$ elements. Hence the approximation ratio will be roughly $\frac{0.33+0.6}{1.6} < 0.582$. Further details are omitted. ■

The example given in the proof of Proposition 8 has slackness in it: the number of columns can be increased from $\frac{8k}{5}$ to nearly $(2 - \frac{1}{e})k$; moreover, as iterations progress smaller decoy items suffice. Removing the sources of slackness will result in an improved upper bound on ρ_{CG}, but we have not tried to calculate an exact numerical value for the improved upper bound.

We now establish that $\rho_{CG} \geq \frac{1}{2} + \epsilon$ for some $\epsilon > 0$ independent of k, n and m. More concretely, our proof will show that $\rho_{CG} > 0.50004$.

Notation and terminology used in the proof. Our proof is partly based on case analysis. As such, we shall consider four different events. With each event we shall associate certain parameters. We use the following conventions in naming these parameters. A parameter ϵ_i refers to fraction of

elements covered, whereas a parameter δ_i refers to fraction of rounds. The subscript i refers to the event number (e.g., ϵ_1 is associated with Event 1). The values of the parameters (all in the range $(0,1)$) will be instantiated towards the end of the proof. (Our instantiation is not optimal: a better optimized instantiation will lead to a stronger lower bound on ρ_{CG}.)

Let $O \subset M$ denote the elements covered by an optimal solution, with $\text{opt} = |O|$. For a set T of items, let $f(T)$ denote the set of elements covered by T, and let $|f(T)|$ denote their number. Let T_ℓ denote the set of items held by the conditional greedy algorithm after round ℓ. Let $R_\ell = f(T_\ell) \setminus f(T_{\ell-1})$ denote the set of newly covered elements by the item selected by the conditional greedy algorithm in round ℓ. Hence we need to show that $\sum_{\ell=1}^{k} \frac{|R_\ell|}{\text{opt}} \geq \frac{1}{2} + \epsilon$, or in other words, that on average (over $1 \leq \ell \leq k$) $\frac{|R_\ell|}{\text{opt}} \geq \frac{1}{2k} + \frac{\epsilon}{k}$.

For each element j of $O \setminus f(T_{\ell-1})$, let y_j denote the probability that none of $k - \ell$ items chosen at random from $U \setminus T_{\ell-1}$ cover element j. Let $Y_\ell = \frac{1}{\text{opt}} \sum_{j \in O \setminus f(T_{\ell-1})} y_j$.

We alert the reader that the definition above of y_j (and consequently of Y_ℓ) is a key definition in our proof. Some subtleties associated with this definition are pointed out in parenthetical remarks in the proofs of propositions 9 and 10.

Event 1. For some ℓ it holds that $Y_\ell \leq \frac{1}{2} - \epsilon_1$.

Proposition 9 *If Event 1 happens, then* $\rho_{CG} \geq \frac{1}{2} + \epsilon_1$.

Proof. As the conditional greedy algorithm is equivalent to derandomization using the method of conditional expectations, its final output has value at least as large as the expected value of any distribution encountered in intermediate rounds. Given $T_{\ell-1}$, the expected value of a random set of k items that contains $T_{\ell-1}$ is at least $(1 - Y_\ell)\text{opt}$, because each element j of O is either already covered by $T_{\ell-1}$, or else has probability at least $1 - y_j$ of being covered by the additional $k - \ell + 1$ random items (the probability is at least $1 - y_j$ rather than exactly $1 - y_j$ because y_j is computed with respect to only $k - \ell$ additional random items). For $Y_\ell \leq \frac{1}{2} - \epsilon_1$ the proof follows. ■

Event 2. There are more than $\delta_2 k$ rounds in which $Y_\ell \geq \frac{1}{2} + \epsilon_2$.

Proposition 10 *If Event 2 happens and Event 1 does not happen then* $\rho_{CG} \geq \frac{1}{2} + \delta_2\epsilon_2 - \epsilon_1$.

Proof. Since the optimal solution contains k items that cover all elements of O, it must be that one of the items of the optimal solution, if selected, would contribute a marginal value of at least $\frac{Y_\ell}{k}$opt. (Here we point out a subtlety in our definitions. Let i be an item belonging to the optimal solution but not to $T_{\ell-1}$, and let $R_{i,\ell}$ denote those elements that i contains that are not already covered by $T_{\ell-1}$. Let $Y_{i,\ell} = \sum_{j \in R_{i,\ell}} y_j$. Then if $Y_{i,\ell} > 0$, the marginal contribution of item i to $T_{\ell-1}$ is not just $Y_{i,\ell}$, but in fact strictly larger. The reason for the strict inequality is that y_j is computed by considering the random event of selecting $k - \ell$ items at random from $U \setminus T_{\ell-1}$, whereas for the marginal contribution of i we need to consider the random event of selecting $k - \ell$ items at random from $U \setminus (T_{\ell-1} \cup \{i\})$. The latter random event gives strictly larger marginal value for i, because the realizations that it excludes are realizations that lead to i having no marginal value.) The conditional greedy rule then implies that indeed $\frac{|R_\ell|}{\text{opt}} \geq \frac{Y_\ell}{k}$. Hence $\sum_{\ell=1}^{k} \frac{|R_\ell|}{\text{opt}} \geq \sum_{\ell=1}^{k} \frac{Y_\ell}{k}$. Event 1 not happening implies that $Y_\ell \geq \frac{1}{2} - \epsilon_1$ in every round, whereas Event 2 implies that $Y_\ell \geq \frac{1}{2} + \epsilon_2$ for at least $\delta_2 k$ rounds. Hence $\sum_{\ell=1}^{k} \frac{Y_\ell}{k} \geq \frac{1}{2} - \epsilon_1 + \delta_2\epsilon_2$, as desired. ■

The above two propositions imply (after an appropriate instantiations of the parameters) that either an approximation ratio of $\frac{1}{2} + \epsilon$ is achieved via events 1 or 2, or else we may assume that $Y_\ell \simeq \frac{1}{2}$ in almost all rounds.

As argued above, the greedy rule implies that $|R_\ell| \geq \text{opt}\frac{Y_{\ell-1}}{k}$. The assumption $Y_{\ell-1} \simeq \frac{1}{2}$ implies a lower bound of $\simeq \frac{\text{opt}}{2k}$ on $|R_\ell|$. The next observation that we make is that R_ℓ (for typical ℓ) can be assumed to contain only a negligible amount of items not from O. Event 3 gives a precise definition of the failure of this assumption. The choice of constant $1/10$ in the definition of Event 3 is to some extent arbitrary (though coordinated with the choice 0.9 in the statement of Corollary 14), and was done so as not to introduce too many uninstantiated parameters in the proof.

Event 3. Among the first $k/10$ rounds there are $\delta_3 k$ rounds in which R_ℓ contains at least $\frac{\epsilon_3}{k}$opt elements not from O.

Proposition 11 *If Event 3 happens and Event 1 does not happen then* $\rho_{CG} \geq \frac{1}{2} + \frac{9\epsilon_3\delta_3}{19} - \frac{\epsilon_1}{10}$.

Proof. Suppose that Event 3 happens. Then $f(T_{k/10})$ contains at least $\epsilon_3\delta_3$opt elements not from O. Consider now rounds $\ell > \frac{k}{10}$. If in any such round $Y_\ell \leq \frac{1}{2} + \frac{10\epsilon_3\delta_3}{19}$ then an argument similar to Proposition 9 implies that $\rho_{CG} \geq \frac{1}{2} - \frac{10\epsilon_3\delta_3}{19} + \epsilon_3\delta_3 = \frac{1}{2} + \frac{9\epsilon_3\delta_3}{19}$.

Hence we may assume that $Y_\ell \geq \frac{1}{2} + \frac{10\epsilon_3\delta_3}{19}$ for all $\ell > \frac{k}{10}$. Also, given that Event 1 does not happen then $R_\ell \geq (\frac{1}{2k} - \frac{\epsilon_1}{k})\text{opt}$ for every round ℓ, implying that $|f(T_{k/10})| \geq (\frac{1}{2} - \epsilon_1)\frac{\text{opt}}{10}$. Combining these two facts together, an argument similar to Proposition 10 implies that $\rho_{CG} \geq \frac{1}{2} + \frac{9}{10}\frac{10\epsilon_3\delta_3}{19} - \frac{\epsilon_1}{10} = \frac{1}{2} + \frac{9\epsilon_3\delta_3}{19} - \frac{\epsilon_1}{10}$. ■

Hence we deduce that typically $|R_\ell \cap O| \simeq \frac{\text{opt}}{2k}$, and $|R_\ell \setminus O|$ is negligible in comparison. We will next argue that almost all those elements j covered by R_ℓ are elements that had negligible probability to be covered by a random set S of $k-1$ items that includes $T_{\ell-1}$.

Event 4. Among the first $\frac{k}{10}$ rounds there are at least $\delta_4 k$ rounds for which $\sum_{j \in R_\ell \cap O} y_j \leq (\frac{1}{2k} - \frac{\epsilon_4}{k})\text{opt}$.

Proposition 12 *Suppose that $\delta_4 \geq \delta_3$ and that $\epsilon_4 > \epsilon_1 + \epsilon_3$. Then if events 1 and 3 do not happen then neither does Event 4.*

Proof. Suppose that Event 4 happens and Event 3 does not. Then $\delta_4 \geq \delta_3$ implies that there is some round ℓ for which $\sum_{j \in R_\ell \cap O} y_j \leq (\frac{1}{2k} - \frac{\epsilon_4}{k})\text{opt}$ and R_ℓ contains at most $\frac{\epsilon_3}{k}\text{opt}$ elements not from O. Hence the marginal value contributed by including the item that corresponds to R_ℓ in the random set S is at most $(\frac{1}{2k} - \frac{\epsilon_4}{k} + \frac{\epsilon_3}{k})\text{opt}$.

Suppose that Event 1 does not happen. Then in every round, there must be an item of marginal value at least $(\frac{1}{2k} - \frac{\epsilon_1}{k})\text{opt}$. The conditional greedy rule implies that $\frac{1}{2k} - \frac{\epsilon_4}{k} + \frac{\epsilon_3}{k} \geq \frac{1}{2k} - \frac{\epsilon_1}{k}$, implying that $\epsilon_4 \leq \epsilon_1 + \epsilon_3$. ■

We now instantiate the values of the parameters with two goals in mind. One is to ensure that if any of the events 1 to 4 happens, then $\rho_{CG} \geq \frac{1}{2} + \epsilon$. The other is to ensure that if none of the events 1 to 4 happens, then also in this case $\rho_{CG} \geq \frac{1}{2} + \epsilon$. The δ_i parameters are instantiated to values that are anticipated to be useful for establishing the second goal (though are not necessarily optimized for this goal). Based on the values of the δ_i parameters we instantiate the values of the ϵ_i parameters to be essentially the smallest possible that suffice for the first goal. The values of the ϵ_i parameters are expressed as a function of ϵ, where ϵ will be instantiated later.

- $\epsilon_1 = \epsilon$.
- $\delta_2 = \frac{1}{20}$.
- $\epsilon_2 = 40\epsilon$.
- $\delta_3 = \delta_4 = \frac{1}{400}$.

- $\epsilon_3 = 928.9\epsilon$
- $\epsilon_4 = 930\epsilon$.

Given the above parameters, if Event 1 happens the Theorem 4 follows from Proposition 9, because $\epsilon_1 = \epsilon$. If Event 1 does not happen but Event 2 happens then the theorem follows from Proposition 10, because $\delta_2\epsilon_2 - \epsilon_1 = \epsilon$. If Event 1 does not happen and Event 3 happens the theorem follows from Proposition 11 because $\frac{9\epsilon_3\delta_3}{19} - \frac{\epsilon_1}{10} > \epsilon$. If neither Event 1 nor Event 3 happen, then Proposition 12 with our setting of the parameters implies that also Event 4 does not happen.

From now on we assume that none of the events 1 to 4 happen. The above shows that for our setting of the parameters, this assumption can be made without loss of generality, because when it does not hold the theorem is proved. To orient the reader, let us now present a plan of how the proof of Theorem 4 might be completed, given the assumption.

Event 2 not happening implies that there is a round among the first $\frac{k}{20}$ rounds for which $Y_\ell \leq \frac{1}{2} + 40\epsilon$. Thereafter, Event 4 not happening implies that Y_ℓ decreases by $\frac{1}{2k} - \frac{930\epsilon}{k}$ in every round up to round $k/10$, except for at most $\frac{k}{400}$ rounds. Hence by round $k/10$ the value of Y_ℓ is at most $\frac{1}{2} + 40\epsilon - (\frac{k}{20} - \frac{k}{400})(\frac{1}{2k} - \frac{930\epsilon}{k}) = \frac{1}{2} - \frac{19}{800} + \frac{16835\epsilon}{200}$. Setting $\epsilon = \frac{1}{4000}$ implies that $Y_{k/10} < \frac{1}{2} - \epsilon$, contradicting the assumption that Event 1 does not hold.

However, the above argument is flawed for the following reason. For an element $j \in (O \setminus f(T_\ell))$, the value of y_j increases from round ℓ to round $\ell+1$, because in round $\ell + 1$ one less random item is chosen, and then element j has higher probability of remaining uncovered. To complete the proof, we need to account for this effect. We make a short digression to prove an inequality (Corollary 14) that will be used for this purpose.

2.3.1 An inequality concerning drawing balls from an urn

Suppose that one draws at random without repetition k balls out of an urn that contains $n \geq k$ balls, where p balls in the urn are black and $n - p$ balls are white. Let W denote the event that all k balls are white, and let $P_W(n, k, p)$ denote its probability. Using this notation, $P_W(n - 1, k - 1, p)$ is the probability of W, conditioned on the first ball being white. Let $\Delta(n, k, p) = P_W(n - 1, k - 1, p) - P_W(n, k, p)$ denote the increase in the probability of W gained by the first ball being white. Observe that if $n < p + k$ then $P_W(n, k, p) = 0$ and $\Delta(n, k, p) = 0$. Likewise, if $p = 0$ then $P_W(n, k, p) = 1$ and $\Delta(n, k, p) = 0$. Hence we shall assume that $0 < p \leq n - k$.

Proposition 13 *Using the notation above,* $\Delta(n,k,p) \leq (1+o(1))\frac{1}{ek}$ *regardless of* p*, where the* $o(1)$ *term tends to 0 as* k *grows.*

Proof. $P_W(n,k,p) = \frac{\binom{n-p}{k}}{\binom{n}{k}} = \frac{(n-p)_k}{(n)_k}$, where $(a)_b$ for integers $a > b > 0$ is shorthand notation for $\frac{a!}{(a-b)!}$. Likewise, $P_W(n-1,k-1,p) = \frac{(n-p-1)_{k-1}}{(n-1)_{k-1}} = \frac{n}{n-p}P_W(n,k,p)$. Hence $\Delta(n,k,p) = \frac{p}{n-p}P_W(n,k,p) = \frac{p(n-p-1)_{k-1}}{(n)_k}$.

We wish now to determine the value of p that maximizes $\Delta(n,k,p) = \frac{p(n-p-1)_{k-1}}{(n)_k}$. Changing p to $p+1$ we get that $\Delta(n,k,p+1) = \frac{(p+1)(n-p-2)_{k-1}}{(n)_k}$. It follows that:

$$\Delta(n,k,p+1) - \Delta(n,k,p) = \frac{(p+1)(n-p-2)_{k-1} - p(n-p-1)_{k-1}}{(n)_k}$$

$$= \left(\frac{p+1}{n-p-1} - \frac{p}{n-p-k}\right)\frac{(n-p-1)_k}{(n)_k} = (n-kp)\frac{(n-p-2)_{k-2}}{(n)_k}$$

Hence $\Delta(n,k,p)$ increases with p as long as $p < \frac{n}{k}$, and decreases with p for $p > \frac{n}{k}$. The maximum is attained when $p \simeq \frac{n}{k}$, and then $\Delta(n,k,p) = \frac{p(n-p-1)_{k-1}}{(n)_k} \leq \frac{p}{n}(1-\frac{1}{k})^{k-1} \simeq \frac{1}{ek}$. ■

Corollary 14 *Let* k_0 *be sufficiently large, and let* $k \geq k_0$ *be arbitrary subject to the constraint* $k_0 \geq 0.95k$*. Choose an integer* k' *uniformly at random in the range* $[k_0 - \frac{k}{20}, k_0]$*, and let* $n' > k'$ *and* p *be arbitrary. Then in the setting of Proposition 13,* $E[\Delta(n',k',p)] \leq \frac{0.4}{k}$*, where expectation is taken over choice of random* k'*.*

Proof. Proposition 13 implies that $\Delta(n',k',p) \leq (1+o(1))\frac{1}{ek'}$, and hence $E[\Delta(n',k',p)] \leq (1+o(1))E[\frac{1}{ek'}]$. The expectation of $\frac{1}{k'}$ is $20\int_{k_0-\frac{k}{20}}^{k_0}\frac{1}{x}dx \leq 20\int_{0.9k}^{0.95k}\frac{1}{x}dx = \frac{20}{k}\ln\frac{95}{90}$. Hence:

$$E[\Delta(n',k',p)] \leq (1+o(1))\frac{1}{e}\frac{20}{k}\ln\frac{95}{90} \leq \frac{0.4}{k}$$

for sufficiently large k (so that rounding up to 0.4 absorbs the $o(1)$). ■

2.3.2 Back to the analysis of the conditional greedy algorithm

Let us consider how the value y_j for a yet uncovered (by $T_{\ell-1}$) element $j \in O$ is computed in round ℓ. Let $n' = n - \ell + 1$ denote the number of items not in $T_{\ell-1}$, and let $k' = k - \ell$ denote the number of random items that need to be added to $T_{\ell-1}$ in order to compute y_j. Let p denote the number of items in $U \setminus T_{\ell-1}$ that cover element j. Then y_j is precisely $P_W(n', k', p)$ from Section 2.3.1. If the item chosen in round ℓ does not cover element j, then the value of y_j in round $\ell + 1$ (which we denote by y'_j) will be precisely $P_W(n' - 1, k' - 1, p)$ from Section 2.3.1. Let Δ_j denote the increase in y_j, namely, $\Delta_j = y'_j - y_j$. This is precisely $\Delta(n', k', p)$ from Section 2.3.1. Corollary 14 then implies that for a window of width $k/20$ lying in the range $[0.9k, k]$, on average over the choice of k' in this window, $\Delta_j \leq \frac{0.4}{k}$.

We can now return to the proof plan outlined before Section 2.3.1. Event 2 not happening implies that there is a round ℓ^* among the first $\frac{k}{20}$ rounds for which $Y_{\ell^*} \leq \frac{1}{2} + 40\epsilon$. Thereafter, Event 4 not happening implies that Y_ℓ decreases by $\frac{1}{2k} - \frac{930\epsilon}{k}$ in every round up to round $k/10$, except for at most $\frac{k}{400}$ rounds. In each of these rounds Corollary 14 contributes to Y_ℓ a possible increase of at most $\frac{0.4}{k}$. Hence by round $\ell = \ell^* + \frac{k}{20} \leq \frac{k}{10}$ the value of Y_ℓ is at most

$$\frac{1}{2} + 40\epsilon - (\frac{k}{20} - \frac{k}{400})(\frac{1}{2k} - \frac{930\epsilon}{k}) + \frac{k}{20}\frac{0.4}{k} = \frac{1}{2} - \frac{3}{800} + \frac{16835\epsilon}{200}$$

Setting $\epsilon = \frac{1}{24000}$ implies that $Y_\ell < \frac{1}{2} - \epsilon$, contradicting the assumption that Event 1 does not hold.

The above completes our proof for Theorem 4.

3 Discussion

In this work we considered the problem max k-submodular and showed that the best approximation ratio guaranteed by polynomial time algorithms that use only k-queries is $\frac{1}{2} + o(1)$ (where the $o(1)$ term tends to 0 as k grows), which is not as good as the ratio $1 - \frac{1}{e}$ that is achievable if general value queries are allowed. This shows that in settings in which "small" queries (to sets much smaller than k) are more costly than k-queries (see example in Section 1.2), the benefits that small queries provide in reducing the number of queries may more than compensate for their higher costs.

We also showed that max k-coverage can be approximated within a ratio strictly better than max k-submodular, in our setting in which only k-queries

are allowed. We remark that in many other settings (e.g., if the function f is described explicitly, or if arbitrary value queries are allowed), the best approximation ratios for both problems are the same.

3.1 Robustness of our results

One may ask whether our results are robust to small changes in the model. One change is to enlarge the set of uniform size queries to queries that involve sets of cardinality k' for all $(1-\epsilon')k \leq k' \leq (1+\epsilon')k$ (where $\epsilon' > 0$ is some small constant). Naturally, the positive results (Theorem 1) still hold. We point out that the main negative result, Theorem 3, still holds as well (with some correction factor that depends on ϵ').

A different form of robustness alluded to in the example in Section 1.2 is tolerating inexact replies to queries. That is, in reply to a query S, the reply r might not be exactly $r = f(S)$, but rather some value chosen by an adversary but satisfying $(1-\epsilon')f(S) \leq r \leq (1+\epsilon')f(S)$ (where again $\epsilon' > 0$ is some small constant). Here the negative results (Theorem 3) obviously continue to hold. We point out that the positive results, Theorem 1 and Theorem 4, require ϵ' to be considerable smaller than $1/k$, because the algorithms (local search and conditional greedy) need to observe a change in value when one out of k items in a set is replaced.

3.2 Max k-coverage

The following question remains open:

Question 15 *Using only k-queries and polynomial time computations, is an approximation ratio of $1 - \frac{1}{e}$ achievable for max k-coverage?*

If the answer is negative, it would be interesting to determine where between $\frac{1}{2}$ and $1 - \frac{1}{e}$ the best approximation ratio for max k-coverage lies.

In the context of max k-coverage it is natural to consider a query model that allows for more informative answers to the queries. Recall that in the example from Section 1.2, a query corresponds to offering a menu of items, and a reply is the (empirically observed) fraction of clients that are satisfied by the menu. In some contexts, it is natural to assume that the firm can also observe which item the satisfied client chooses from the menu. Casting this in the setting of a value query (whose answer is the average behavior of a large set of clients), the reply to a query (a set) is a vector of probabilities specifying what fraction of clients chose each item in the menu. We call such replies vector replies (in contrast to the standard scalar replies). One

may ask whether having vector replies to k-queries helps in improving the approximation ratio for max k-coverage. The answer may depend on how one models the clients: if a client has several desirable items on a menu, which one of the items will the client choose?

One possible client model assumes that the items on the menu are ordered (say, from top to bottom), every client scans the menu in this order and chooses the first desirable item that he encounters (if there is a desirable item). This model was previously studied in [15], who show that the standard greedy algorithm for max k-coverage can be implemented in this model, giving an approximation ratio of $1 - \frac{1}{e}$.

A different client model assumes that each client has his own preference order among his desirable items, and given a menu, chooses the desirable item of his highest preference. In this client model the standard greedy algorithm no longer works. Moreover, the negative example of Proposition 2 extends to this model. In the proof of Proposition 2, the clients are the cells of the matrix, the menu items are the k rows and the first k columns, the desirable items for a client are his column (if the client is in the first k columns) and his row. Hence each client desires either one or two items. For those clients that desire two items, suppose that their preference is such that they prefer the column item over the row item. Then the set of first k columns will be a menu on which the vector reply will be $\frac{k}{(2k-1)k}$ for every item, and the same will hold for every neighboring menu (that contains $k-1$ columns and one row).

Of course, one may consider other client models as well. The absence of one client model that is clearly more natural than the others suggests that the scalar reply model is more fundamental than the vector reply model (fewer assumptions need to be justified), and hence this is the model considered in our paper. (This discussion of reply models relates to max k-coverage. For max k-submodular the scalar reply model is the standard model, and it is not clear whether a vector reply model even makes sense.)

Acknowledgements

This work was partly done in Microsoft Research, Herzeliya, Israel. The work of the first author was 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).

References

[1] Bailly, G., Oulasvirta, A., Brumby, D. P., and Howes, A. 2014. Visual search and selection time in linear menus. In *CHI'14*.

[2] Bar-Noy, A. and Lampis, M. 2012. Online maximum directed cut. *J. Comb. Optim. 24(1),* 52–64.

[3] Blumrosen, L. and Nisan, N. 2009. On the computational power of demand queries. *SIAM J. Comput. 39(4),* 1372–1391.

[4] Chen, J., Friesen, D. K., and Zheng, H. 1999. Tight bound on johnson's algorithm for maximum satisfiability. *J. Comput. Syst. Sci. 58(3),* 622–640.

[5] David, R., Dinur, I., Goldenberg, E., Kindler, G., and Shinkar, I. 2015. Direct sum testing. In *ITCS-2015*. 327–336.

[6] Erdos, P. and Selfridge, J. 1973. Online maximum directed cut. *Journal of Combinatorial Theory, Series A 14(3),* 298–301.

[7] Feige, U. 1998. Threshold of $\ln n$ for approximating set cover. *J. ACM 45(4),* 634–652.

[8] Feige, U., Mirrokni, V. S., and Vondrak, J. 2011. Maximizing non-monotone submodular functions. *SIAM J. Comput. 40(4),* 1133–1153.

[9] Feige, U., Raghavan, P. Peleg, D., and Upfal, E. 1994. Computing with Noisy Information. *SIAM J. Comput. 23(5),* 1001–1018.

[10] Grotschel, M., Lovasz, L., and Schrijver, A. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica 1(2),* 169–197.

[11] Guyon, I. and Elisseeff, A. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research 3*, 169–197.

[12] Laffont, J.-J. and Tirole, J. 1986. Using cost observation to regulate firms. *Journal of Political Economy 94*, 169–197.

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

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

[15] Radlinski, F., Kleinberg, R., and Joachims, T. 2008. Learning diverse rankings with multi-armed bandits. In *ICML-2008*. 784–791.

[16] Raghavan, P. 1988. Probabilistic construction of deterministic algorithms. *J. Comput. Syst. Sci. 37(2),* 130–143.

[17] Rogerson, W. P. 2003. Simple menus of contracts in cost-based procurement and regulation. *The American Economic Review 93(3),* 919–926.

[18] Streeter, M. J., Golovin, D., and Krause, A. 2009. Online learning of assignments. In *NIPS-2009*. 1794–1802.

[19] Williamson, D. P. and Shmoys, D. B. 2011. *The Design of Approximation Algorithms*. Cambridge University Press.