Maximizing non-monotone submodular functions

Uriel Feige* Vahab S. Mirrokni* Jan Vondrák†

April 19, 2007

Abstract

Submodular maximization is a central problem in combinatorial optimization, generalizing many important problems including Max Cut in directed/undirected graphs and in hypergraphs, certain constraint satisfaction problems and maximum facility location problems. Unlike the problem of minimizing submodular functions, the problem of maximizing submodular functions is NP-hard.

In this paper, we design the first constant-factor approximation algorithms for maximizing nonnegative submodular functions. In particular, we give a deterministic local search $\frac{1}{3}$-approximation and a randomized $\frac{2}{5}$-approximation algorithm for maximizing nonnegative submodular functions. We also show that a uniformly random set gives a $\frac{1}{4}$-approximation. For symmetric submodular functions, we show that a random set gives a $\frac{1}{2}$-approximation, which can be also achieved by deterministic local search.

These algorithms work in the value oracle model where the submodular function is accessible through a black box returning $f(S)$ for a given set S. We show that in this model, our $\frac{1}{2}$-approximation for symmetric submodular functions is the best one can achieve with a subexponential number of queries.

For the case that the submodular function is given explicitly (specifically, as a sum of polynomially many nonnegative submodular functions, each on a constant number of elements) we prove that for any fixed $\epsilon > 0$, it is NP-hard to achieve a $(\frac{5}{6} + \epsilon)$-approximation for symmetric nonnegative submodular functions, or a $(\frac{3}{4} + \epsilon)$-approximation for general nonnegative submodular functions.

*Microsoft Research. Email: {urifeige,mirrokni}@microsoft.com
†Princeton University, Email: jvondrak@math.princeton.edu

1 Introduction

We consider the problem of *maximizing a nonnegative submodular function.* This means, given a submodular function $f : 2^X \to \mathbb{R}_+$, we want to find a subset $S \subseteq X$ maximizing $f(S)$.

Definition 1.1. *A function $f : 2^X \to \mathbb{R}$ is submodular if for any $S, T \subseteq X$,*

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

An alternative definition of submodularity is the property of *decreasing marginal values*: For any $A \subseteq B \subseteq X$ and $x \in X \setminus B$, $f(B \cup \{x\}) - f(B) \leq f(A \cup \{x\}) - f(A)$. This can be deduced from the first definition by substituting $S = A \cup \{x\}$ and $T = B$; the reverse implication also holds.

We assume a *value oracle* access to the submodular function; i.e., for a given set S, an algorithm can query an oracle to find its value $f(S)$.

Background. Submodularity, a discrete analog of convexity, has played an essential role in combinatorial optimization [26]. It appears in many important settings including cuts in graphs [16, 32, 14], rank function of matroids [8, 15], set covering problems [10], and plant location problems [5, 6]. In many settings such as set covering or matroid optimization, the relevant submodular functions are *monotone*, meaning that $f(S) \leq f(T)$ whenever $S \subseteq T$. Here, we are more interested in the general case where $f(S)$ is not necessarily monotone. A canonical example of such a submodular function is $f(S) = \sum_{e \in \delta(S)} w(e)$, where $\delta(S)$ is a cut in a graph (or hypergraph) induced by a set of vertices S and $w(e)$ is the weight of edge e. Cuts in undirected graphs and hypergraphs yield *symmetric* submodular functions, satisfying $f(S) = f(\bar{S})$ for all sets S. Symmetric submodular functions have been considered widely in the litrature [13, 32]. It appears that symmetry allows better/simpler approximation results, and thus deserves separate attention.

The problem of maximizing a submodular function is of central importance, with special cases including Max Cut [16], Max Directed Cut [21], hypergraph cut problems, maximum facility location [1, 5, 6], and certain restricted satisfiability problems [22, 9]. While the Min Cut problem in graphs is a classical polynomial-time solvable problem, and more generally it has been shown that any submodular function can be *minimized* in polynomial time [34, 14], maximization turns out to be more difficult and indeed all the aforementioned special cases are NP-hard.

A related problem is Max-k-Cover, where the goal is to choose k sets whose union is as large as possible. It is known that a greedy algorithm provides a $(1 - 1/e)$-approximation for Max-k-Cover and this is optimal unless $P = NP$ [10]. More generally, this problem can be viewed as maximization of a monotone submodular function under a cardinality constraint. I.e., we have $0 \leq f(S) \leq f(T)$ for any $S \subseteq T$ and we seek a set S of size k maximizing $f(S)$. Again, the greedy algorithm provides a $(1 - 1/e)$-approximation for this problem [29]. A 1/2-approximation has been developed for maximizing monotone submodular functions under a matroid constraint [30]. A $(1 - 1/e)$-approximation has been also obtained for a knapsack constraint [35], and for a special class of submodular functions under a matroid constraint [3].

In contrast, here we consider the unconstrained maximization of a submodular function which is not necessarily monotone. We only assume that the function is nonnegative.[1] Typical examples of such a problem are Max Cut and Max Directed Cut. Here, the best approximation factors have been achieved using semidefinite programming: 0.878 for Max Cut [16] and 0.859 for Max Di-Cut [9]. The approximation factor for Max Cut has been proved optimal, assuming the Unique Games Conjecture [24, 28]. It should be noted that the best known combinatorial algorithms for

[1]For submodular functions without any restrictions, verifying whether the maximum of the function is greater than zero or not is NP-hard. Thus, no approximation algorithm can be found for this problem unless P=NP. For a general submodular function f with minimum value f^*, we can design an approximation algorithm to maximize a *normalized submodular function* g where $g(S) = f(S) - f^*$.

Max Cut and Max Di-Cut achieve only a 1/2-approximation, which is trivial for Max Cut but not for Max Di-Cut [21].

More generally, submodular maximization encompasses such problems as Max Cut in hypergraphs and Max SAT with no mixed clauses (every clause contains only positive or only negative literals). Tight results are known for Max Cut in k-uniform hypergraphs for any fixed $k \geq 4$ [22, 19] (and the same result for Max $(k-1)$-SAT with no mixed clauses [19, 20]) where the optimal approximation factor $(1 - 2^{-k+1})$ is achieved by a random solution. The lowest approximation factor (7/8) is achieved for $k = 4$; for $k < 4$, better than random solutions can be found by semidefinite programming.

Submodular maximization also appears in maximizing the difference of a monotone submodular function and a modular function. An illustrative example of this type is the maximum facility location problem in which we want to open a subset of facilities and maximize the total profit from clients minus the openning cost of facilities. In a series of papers, approximation algorithms have been developed for a variant of this problem which is a special case of maximizing nonnegative submodular functions [5, 6, 1]. The best approximation factor known for this problem is 0.828 [1].

In the general case of non-monotone submodular functions, the maximization problem has been studied in the operations research community. Many efforts have been focused on designing heuristics for this problem, including data-correcting search methods [17, 18, 23], accelatered greedy algorithms [31], and polyhedral algorithms [25]. Prior to our work, to the best of our knowledge, no guaranteed approximation factor was known for maximizing non-monotone submodular functions.

Our results. We design several constant-factor approximation algorithms for maximization of nonnegative submodular functions. We also prove negative results, in particular a query complexity result matching our algorithmic result in the symmetric case.

Model	Rnd.Set	Non-adapt.	Determ.Adapt.	Rnd.Adapt.	CC bound	NP-hardness
Symmetric	1/2	1/2	1/2	1/2	1/2	5/6
Asymmetric	1/4	1/3	1/3	2/5	1/2	3/4

Figure 1: Summary of our results.

Non-adaptive algorithms. A non-adaptive algorithm is allowed to generate a (possibly random) sequence of polynomially many sets, query their values and then produce a solution. In this model, we show that a 1/4-approximation is achieved in expectation by a uniformly random set. For symmetric submodular functions, this gives a 1/2-approximation. This coincides with the approximation factors obtained by random sets for Max Di-Cut and Max Cut. We prove that these factors cannot be improved, assuming that the algorithm returns one of the queried sets. However, we also design a non-adaptive algorithm which performs a polynomial-time computation on the obtained values and achieves a 1/3-approximation. In the symmetric case, we prove that the 1/2-approximation is optimal even among adaptive algorithms (see below).

Adaptive algorithms. An adaptive algorithm is allowed to perform a polynomial time computation including a polynomial number of queries to a value oracle. In this (most natural) model, we develop a local search 1/2-approximation in the symmetric case and a 1/3-approximation in the general case. Then we improve this to a 2/5-approximation using a randomized "non-oblivious local search". This is perhaps the most noteworthy of our algorithms; it proceeds by locally optimizing a smoothed variant of $f(S)$, obtained by biased sampling depending on S. Non-oblivious local search has been used before to achieve a 2/5-approximation for the Max Di-Cut problem [2]; another (simpler) 2/5-approximation algorithm for Max Di-Cut appears in [21]. However, these algorithms do not generalize naturally to ours and the re-appearance of the same approximation factor seems coincidental.

Hardness results. We show that it is impossible to improve the 1/2-approximation algorithm for maximizing symmetric nonnegative submodular functions, at least in the value oracle model. We prove that for any fixed $\epsilon > 0$, a $(1/2+\epsilon)$-approximation algorithm would require exponentially many queries. This settles the status of symmetric submodular maximization in the value oracle model. Note that this query complexity lower bound does not assume any computational restrictions. In particular, in the special case of Max Cut, polynomially many value queries suffice to infer all edge weights in the graph, and thereafter an exponential time computation (involving no further queries) would actually produce the optimal cut.

For explicitly represented submodular functions, known NP-hardness hardness of approximation results for Max Cut in graphs and hypergraphs provide an obvious limitation to the best possible approximation ratio. We prove stronger limitations. For any fixed $\epsilon > 0$, it is NP-hard to achieve an approximation factor of $(3/4 + \epsilon)$ (or $5/6 + \epsilon$) in the general (or symmetric) case, respectively. These results are valid even when the submodular function is given as a sum of polynomially many nonnegative submodular functions, each on a constant number of variables, which is the case for all the aforementioned problems.

2 Non-adaptive algorithms

It is known that simply choosing a random cut is a good choice for Max Cut and Max DiCut, achieving an approximation factor of 1/2 and 1/4 respectively. We show the natural role of submodularity here by presenting the same approximation factors in the case of general submodular functions.

The Random Set Algorithm: RS.

- Return $R = X(1/2)$, a uniformly random subset of X.

Theorem 2.1. *Let $f : 2^X \to \mathbb{R}_+$ be a submodular function, $OPT = \max_{S\subseteq X} f(S)$ and let R denote a uniformly random subset $R = X(1/2)$. Then $\mathbf{E}[f(R)] \geq \frac{1}{4}OPT$. In addition, if f is symmetric ($f(S) = f(X \setminus S)$ for every $S \subseteq X$), then $\mathbf{E}[f(R)] \geq \frac{1}{2}OPT$.*

Before proving this result, we show a useful probabilistic property of submodular functions (extending the considerations of [11, 12]). This property will be essential in the analysis of our more involved randomized algorithms as well.

Lemma 2.2. *Let $g : 2^X \to \mathbb{R}$ be submodular. Denote by $A(p)$ a random subset of A where each element appears with probability p. Then*

$$\mathbf{E}[g(A(p))] \geq (1-p)\ g(\emptyset) + p\ g(A).$$

Proof. By induction on the size of A: For $A = \emptyset$, the lemma is trivial. So assume $A = A' \cup \{x\}, x \notin A'$. We also write $A'(p) = A(p) \cap A'$; then

$$\begin{aligned} \mathbf{E}[g(A(p))] &= \mathbf{E}[g(A'(p))] + \mathbf{E}[g(A(p)) - g(A(p) \cap A')] \\ &\geq \mathbf{E}[g(A'(p))] + \mathbf{E}[g(A' \cup A(p)) - g(A')] \end{aligned}$$

using submodularity on A' and $A(p)$. The set $A' \cup A(p)$ is either equal to A (when $x \in A(p)$, which happens with probability p) or otherwise it's equal to A'. Therefore we get

$$\mathbf{E}[g(A(p))] \quad \geq \quad \mathbf{E}[g(A'(p)] + p(g(A) - g(A'))$$

and using the inductive hypothesis, $\mathbf{E}[g(A'(p))] \geq (1-p)\ g(\emptyset) + p\ g(A')$, we get the statement of the lemma. $\square$

By a double application of Lemma 2.2, we obtain the following.

Lemma 2.3. *Let $f : 2^X \to \mathbb{R}$ be submodular, $A, B \subseteq X$ two (not necessarily disjoint) sets and $A(p), B(q)$ their independently sampled subsets, where each element of A appears in $A(p)$ with probability p and each element of B appears in $B(q)$ with probability q. Then*

$$\mathbf{E}[f(A(p) \cup B(q))] \geq (1-p)(1-q)\ f(\emptyset) + p(1-q)\ f(A) + (1-p)q\ f(B) + pq\ f(A \cup B).$$

Proof. Condition on $A(p) = A'$ and define $g(T) = f(A' \cup T)$. This is a submodular function as well and Lemma 2.2 implies $\mathbf{E}[g(B(q))] \geq (1-q)\ f(A') + q\ f(A' \cup B)$. Also, $\mathbf{E}[g(B(q))] = \mathbf{E}[f(A(p) \cup B(q)) \mid A(p) = A']$, and by unconditioning: $\mathbf{E}[f(A(p) \cup B(q))] \geq \mathbf{E}[(1-q)\ f(A(p)) + q\ f(A(p) \cup B)]$. Finally, we apply Lemma 2.2 once again: $\mathbf{E}[f(A(p))] \geq (1-p)\ f(\emptyset) + p\ f(A)$, and by applying the same to the submodular function $h(S) = f(S \cup B)$, $\mathbf{E}[f(A(p) \cup B)] \geq (1-p)\ f(B) + p\ f(A \cup B)$. This implies the claim. □

This lemma gives immediately the performance of Algorithm RS.

Proof. Denote the optimal set by S and its complement by $\bar{S}$. We can write $R = S(1/2) \cup \bar{S}(1/2)$. Using Lemma 2.3, we get

$$\mathbf{E}[f(R)] \geq \frac{1}{4} f(\emptyset) + \frac{1}{4} f(S) + \frac{1}{4} f(\bar{S}) + \frac{1}{4} f(X).$$

Every term is nonnegative and $f(S) = OPT$, so we get $\mathbf{E}[f(R)] \geq \frac{1}{4} OPT$. In addition, if f is symmetric, we also have $f(\bar{S}) = OPT$ and then $\mathbf{E}[f(R)] \geq \frac{1}{2} OPT$. □

We can show that the factor of 1/4 is optimal, assuming that an algorithm samples a polynomial number of random sets and returns one of them. However, it is possible to design a 1/3-approximation algorithm which samples random sets non-adaptively and then returns an answer after a polynomial-time computation. Due to space constraints, we defer the details to Appendix A. In the symmetric case, the factor of 1/2 turns out to be *optimal even for adaptive algorithms* in the value oracle model, as we show in Section 4.2.

3 Adaptive algorithms

3.1 A deterministic local search algorithm

Our deterministic algorithm is based on a simple local search technique. We try to increase the value of our solution S by either including a new element in S or discarding one of the elements of S. We call S a *local optimum* if no such operation increases the value of S. Local optima have the following property which was first observed in [4, 18].

Lemma 3.1. *Given a submodular function f, if S is a local optimum of f, and I and J are two sets such that $I \subseteq S \subseteq J$, then $f(I) \leq f(S)$ and $f(J) \leq f(S)$.*

This property turns out to be very useful in proving that a local optimum is a good approximation to the global optimum. However, it is known that finding a local optimum for the Max Cut problem is PLS-complete [33]. Therefore, we relax our local search and find an *approximate local optimal solution.*

Local Search Algorithm: LS.

1. Let $S := \{v\}$ where $f(\{v\})$ is the maximum over all singletons $v \in X$.
2. If there exists an element $a \in X \setminus S$ such that $f(S \cup \{a\}) > (1 + \frac{\epsilon}{n^2}) f(S)$, then let $S := S \cup \{a\}$, and go back to Step 2.

3. If there exists an element $a \in S$ such that $f(S\backslash\{a\}) > (1+\frac{\epsilon}{n^2})f(S)$, then let $S := S\backslash\{a\}$, and go back to Step 2.

4. Return the maximum of $f(S)$ and $f(X\backslash S)$.

It is easy to see that if the algorithm terminates, the set S is a $(1+\frac{\epsilon}{n^2})$-approximate local optimum, in the following sense.

Definition 3.2. *Given $f : 2^X \to \mathbb{R}$, a set S is called a $(1+\alpha)$-*approximate local optimum, *if $(1+\alpha)f(S) \geq f(S \setminus \{v\})$ for any $v \in S$, and $(1+\alpha)f(S) \geq f(S \cup \{v\})$ for any $v \notin S$.*

We prove the following analogue of Lemma 3.1.

Lemma 3.3. *If S is an $(1+\alpha)$-approximate local optimum for a submodular function f, then for any subsets such that $I \subseteq S \subseteq J$, $f(I) \leq (1+n\alpha)f(S)$ and $f(J) \leq (1+n\alpha)f(S)$.*

Proof. Let $I = T_1 \subseteq T_2 \subseteq \ldots \subseteq T_k = S$ be a chain of sets where $T_i\backslash T_{i-1} = \{a_i\}$. For each $2 \leq i \leq k$, we know that $f(T_i)-f(T_{i-1}) \geq f(S)-f(S\backslash\{a_i\}) \geq -\alpha f(S)$ using the submodularity and approximate local optimality of S. Summing up these inequalities, we get $f(S) - f(I) \geq -k\alpha f(S)$. Thus $f(I) \leq (1+k\alpha)f(S) \leq (1+n\alpha)f(S)$. This completes the proof for set I. The proof for set J is very similar. □

Theorem 3.4. *Algorithm LS is a $\left(\frac{1}{3}-\frac{\epsilon}{n}\right)$-approximation algorithm for maximizing nonnegative submodular functions, and a $\left(\frac{1}{2}-\frac{\epsilon}{n}\right)$-approximation algorithm for maximizing nonnegative symmetric submodular functions. The algorithm uses at most $O(\frac{1}{\epsilon}n^3\log n)$ oracle calls.*

Proof. Consider an optimal solution C and let $\alpha = \frac{\epsilon}{n^2}$. If the algorithm terminates, the set S obtained at the end is a $(1+\alpha)$-approximate local optimum. By Lemma 3.3, $f(S\cap C) \leq (1+n\alpha)f(S)$ and $f(S\cup C) \leq (1+n\alpha)f(S)$. Using submodularity, $f(S\cup C)+f(X\backslash S) \geq f(C\backslash S)+f(X) \geq f(C\backslash S)$, and $f(S\cap C) + f(C\backslash S) \geq f(C) + f(\emptyset) \geq f(C)$. Using these inequalities, we get

$$2(1+n\alpha)f(S) + f(X\backslash S) \geq f(S\cap C) + f(S\cup C) + f(X\backslash S) \geq f(S\cap C) + f(C\backslash S) \geq f(C).$$

For $\alpha = \frac{\epsilon}{n^2}$, this implies that either $f(S) \geq (\frac{1}{3} - o(1))OPT$ or $f(X\backslash S) \geq (\frac{1}{3} - o(1))OPT$.

For symmetric submodular functions, we get

$$2(1+n\alpha)f(S) \geq f(S\cap C) + f(S\cup \bar{C}) = f(S\cap C) + f(\bar{S}\cap C) \geq OPT$$

and hence $f(S)$ is a $(\frac{1}{2} - o(1))$-approximation.

To bound the running time of the algorithm, let v be the element with the maximum $f(\{a\})$ over all elements of X. It is simple to see that $OPT \leq nf(\{a\})$. Since after each iteration, the value of the function increases by a factor of at least $(1+\frac{\epsilon}{n^2})$, if the number of iterations of the algorithm is k, then $(1+\frac{\epsilon}{n^2})^k \leq n$. Therefore, $k = O(\frac{1}{\epsilon}n^2\log n)$ and the number of queries is $O(\frac{1}{\epsilon}n^3\log n)$. □

3.2 A randomized $2/5$-approximation algorithm

Next, we present a randomized algorithm which improves the approximation ratio of $1/3$. The main idea behind this algorithm is to find a "smoothed" local optimum, where elements are sampled randomly but with different probabilities, based on some underlying set A. The general approach of local search, based on a function derived from the one we are interested in, has been referred to as "non-oblivious local search" in the literature [2].

Definition 3.5. *We say that a set is sampled with bias δ based on A, if elements in A are sampled independently with probability $p = (1+\delta)/2$ and elements outside of A are sampled independently with probability $q = (1-\delta)/2$. We denote this random set by $\mathcal{R}(A,\delta)$.*

The Smooth Local Search algorithm: SLS.

1. Choose $\delta \in [0,1]$ and start with $A = \emptyset$. Let $n = |X|$ denote the total number of elements. In the following, use an estimate for OPT, for example from Algorithm RS.

2. For each element x, estimate $\omega_{A,\delta}(x) = \mathbf{E}[f(\mathcal{R}(A,\delta) \cup \{x\})] - \mathbf{E}[f(\mathcal{R}(A,\delta) \setminus \{x\})]$, within an error of $\frac{1}{n^2}OPT$. Call this estimate $\tilde{\omega}_{A,\delta}(x)$.

3. If there is an element $x \in X \setminus A$ such that $\tilde{\omega}_{A,\delta}(x) > \frac{2}{n^2}OPT$, include x in A and go to Step 2.

4. If there is $x \in A$ such that $\tilde{\omega}_{A,\delta}(x) < -\frac{2}{n^2}OPT$, remove x from A and go to Step 2.

5. Choose $\delta' \in [-1,1]$ and return a random set from the distribution $\mathcal{R}(A,\delta')$.

In effect, we find an approximate local optimum of a derived function $\Phi(A) = \mathbf{E}[f(\mathcal{R}(A,\delta))]$. Then we return a set sampled according to $\mathcal{R}(A,\delta')$; possibly for $\delta' \neq \delta$. One can run Algorithm SLS with $\delta = \delta'$ and prove that the best approximation factor for such parameters is achieved by setting $\delta = \delta' = \frac{\sqrt{5}-1}{2}$, the golden ratio. Then, we get an approximation factor of $\frac{3-\sqrt{5}}{2} - o(1) \geq 0.38$. Interestingly, we can improve this approximation factor to 0.4 by choosing two parameter pairs (δ,δ') and taking the maximum of the two solutions.

Theorem 3.6. *Algorithm SLS runs in polynomial time. If we run SLS for two choices of parameters, $(\delta = \frac{1}{3}, \delta' = \frac{1}{3})$ and $(\delta = \frac{1}{3}, \delta' = -1)$, the better of the two solutions has expected value at least $(\frac{2}{5} - o(1))OPT$.*

Proof. Let $\Phi(A) = \mathbf{E}[f(\mathcal{R}(A,\delta))]$. Recall that in $\mathcal{R}(A,\delta)$, elements from A are sampled with probability $p = (1+\delta)/2$, while elements from B are sampled with probability $q = (1-\delta)/2$. Consider Step 3 where an element x is added to A. Also, let $B' = X \setminus (A \cup \{x\})$. The reason why x is added to A is that $\tilde{\omega}_{A,\delta}(x) > \frac{2}{n^2}OPT$; i.e. $\omega_{A,\delta}(x) > \frac{1}{n^2}OPT$. During this step, $\Phi(A)$ increases by

$$\begin{aligned}
\Phi(A \cup \{x\}) - \Phi(A) &= \mathbf{E}[f((A \cup \{x\})(p) \cup B'(q)] - \mathbf{E}[f(A(p) \cup (B' \cup \{x\})(q))] \\
&= (p-q)\ \mathbf{E}[f(A(p) \cup B'(q) \cup \{x\}) - f(A(p) \cup B'(q))] \\
&= \delta\ \mathbf{E}[f(\mathcal{R}(A,\delta) \cup \{x\}) - f(\mathcal{R}(A,\delta) \setminus \{x\})] = \delta\ \omega_{A,\delta}(x) > \frac{\delta}{n^2}OPT.
\end{aligned}$$

Similarly, executing Step 4 increases $\Phi(A)$ by at least $\frac{\delta}{n^2}OPT$. Since the value of $\Phi(A)$ is always between 0 and OPT, the algorithm cannot iterate more than n^2/δ times and thus it runs in polynomial time. Also, note that finding a local maximum of $\Phi(A)$ is equivalent to finding a set A such that its elements $x \in A$ have $\omega_{A,\delta}(x) \geq 0$, while elements $x \notin A$ have $\omega_{A,\delta}(x) \leq 0$. Here, we find a set satisfying this up to a certain error.

From now on, let A be the set at the end of the algorithm and $B = X \setminus A$. We also use $R = A(p) \cup B(q)$ to denote a random set from the distribution $\mathcal{R}(A,\delta)$. We denote by C the optimal solution, while our algorithm returns either R (for $\delta' = \delta$) or B (for $\delta' = -1$). When the algorithm terminates, we have $\omega_{A,\delta}(x) \geq -\frac{3}{n^2}OPT$ for any $x \in A$, and $\omega_{A,\delta}(x) \leq \frac{3}{n^2}OPT$ for any $x \in B$. Consequently, for any $x \in B$ we have $\mathbf{E}[f(R \cup \{x\})] - f(R)] = \frac{1}{2}\mathbf{E}[f(R \cup \{x\}) - f(R \setminus \{x\})] = \frac{1}{2}\omega(x) \leq \frac{2}{n^2}OPT$. Let's order the elements of $B \cap C = \{b_1, \dots, b_\ell\}$ and write

$$f(R \cup (B \cap C)) = f(R) + \sum_{j=1}^{\ell} (f(R \cup \{b_1, \dots, b_j\}) - f(R \cup \{b_1, \dots, b_{j-1}\})).$$

By the property of decreasing marginal values, we get $f(R \cup \{b_1, \dots, b_j\}) - f(R \cup \{b_1, \dots, b_{j-1}\}) \leq f(R \cup \{b_j\}) - f(R)$ and hence

$$\mathbf{E}[f(R \cup (B \cap C))] \leq \mathbf{E}[f(R)] + \sum_{x \in B \cap C} \mathbf{E}[f(R \cup \{x\}) - f(R)] \leq \mathbf{E}[f(R)] + \frac{2}{n}OPT.$$

Similarly, we can obtain $\mathbf{E}[f(R\cap(B\cup C))] \leq \mathbf{E}[f(R)]+\frac{2}{n}OPT$. This means that instead of R, we can analyze $R\cup(B\cap C)$ and $R\cap(B\cup C)$. In order to estimate $\mathbf{E}[f(R\cup(B\cap C))]$ and $\mathbf{E}[f(R\cap(B\cup C))]$, we use a further extension of Lemma 2.3 which can be proved by another iteration of the same proof:

$$(*) \quad \mathbf{E}[f(A_1(p_1)\cup A_2(p_2)\cup A_3(p_3))] \geq \sum_{I\subseteq\{1,2,3\}} \prod_{i\in I} p_i \prod_{i\notin I}(1-p_i)\ f\left(\bigcup_{i\in I} A_i\right).$$

First, we deal with $R\cap(B\cup C) = (A\cap C)(p)\cup(B\cap C)(q)\cup(B\setminus C)(q)$. We plug in $\delta = 1/3$, i.e. $p = 2/3$ and $q = 1/3$. Then (*) yields

$$\mathbf{E}[f(R\cap(B\cup C))] \geq \frac{8}{27}f(A\cap C)+\frac{2}{27}f(B\cup C)+\frac{2}{27}f(B\cap C)+\frac{4}{27}f(C)+\frac{4}{27}f(F)+\frac{1}{27}f(B)$$

where we denote $F = (A\cap C)\cup(B\setminus C)$ and we discarded the terms $f(\emptyset)\geq 0$ and $f(B\setminus C)\geq 0$. Similarly, we estimate $\mathbf{E}[f(R\cup(B\cap C))]$, applying (*) to a submodular function $h(R) = f(R\cup(B\cap C))$ and writing $\mathbf{E}[f(R\cup(B\cap C))] = \mathbf{E}[h(R)] = \mathbf{E}[h((A\cap C)(p)\cup(A\setminus C)(p)\cup B(q))]$:

$$\mathbf{E}[f(R\cup(B\cap C))] \geq \frac{8}{27}f(A\cup C)+\frac{2}{27}f(B\cup C)+\frac{2}{27}f(B\cap C)+\frac{4}{27}f(C)+\frac{4}{27}f(\bar{F})+\frac{1}{27}f(B).$$

Here, $\bar{F} = (A\setminus C)\cup(B\cap C)$. We use $\mathbf{E}[f(R)]+\frac{2}{n}OPT \geq \frac{1}{2}(\mathbf{E}[f(R\cap(B\cup C))]+\mathbf{E}[f(R\cup(B\cap C))])$ and combine the two estimates.

$$\begin{aligned}\mathbf{E}[f(R)]+\frac{2}{n}OPT \quad \geq \quad & \frac{4}{27}f(A\cap C)+\frac{4}{27}f(A\cup C)+\frac{2}{27}f(B\cap C)+\frac{2}{27}f(B\cup C)\\ & +\frac{4}{27}f(C)+\frac{2}{27}f(F)+\frac{2}{27}f(\bar{F})+\frac{1}{27}f(B).\end{aligned}$$

Now we add $\frac{3}{27}f(B)$ on both sides and apply submodularity: $f(B)+f(F)\geq f(B\cup C)+f(B\setminus C)\geq f(B\cup C)$ and $f(B)+f(\bar{F})\geq f(B\cup(A\setminus C))+f(B\cap C)\geq f(B\cap C)$. This leads to

$$\mathbf{E}[f(R)]+\frac{1}{9}f(B)+\frac{2}{n}OPT \geq \frac{4}{27}f(A\cap C)+\frac{4}{27}f(A\cup C)+\frac{4}{27}f(B\cap C)+\frac{4}{27}f(B\cup C)+\frac{4}{27}f(C)$$

and once again using submodularity, $f(A\cap C)+f(B\cap C)\geq f(C)$ and $f(A\cup C)+f(B\cup C)\geq f(C)$, we get

$$\mathbf{E}[f(R)]+\frac{1}{9}f(B)+\frac{2}{n}OPT \geq \frac{12}{27}f(C) = \frac{4}{9}OPT.$$

To conclude, either $\mathbf{E}[f(R)]$ or $f(B)$ must be at least $(\frac{2}{5}-\frac{2}{n})OPT$, otherwise we get a contradiction. □

4 Inapproximability Results

In this section, we give hardness results for submodular maximization. Our results are of two flavors. First, we consider submodular functions that have a succint representation on the input, in the form of a sum of "building blocks" of constant size. Note that all the special cases such as Max Cut are of this type. For algorithms in this model, we prove complexity-theoretic inapproximability results. The strongest one is that in the general case, a $(3/4+\epsilon)$-approximation for any fixed $\epsilon > 0$ would imply $P = NP$.

In the value oracle model, we show a much tighter result. Namely, any algorithm achieving a $(1/2+\epsilon)$-approximation for a fixed $\epsilon > 0$ would require an exponential number of queries to the value oracle. This holds even in the case of symmetric submodular functions, i.e. our 1/2-approximation algorithm is optimal in this model.

4.1 NP-hardness results

Our reductions are based on Håstad's 3-bit and 4-bit PCP verifiers [22]. Some inapproximability results can be obtained immediately from [22], by considering the known special cases of submodular maximization. The strongest result along these lines is that the Max Cut problem in 4-uniform hypergraphs is NP-hard to approximate within a factor better than 7/8. Therefore, we get the same hardness result for submodular maximization.

We obtain stronger hardness results by reductions from systems of parity equations. The parity function is not submodular, but we can obtain hardness results by a careful construction of a "submodular gadget" for each equation.

Theorem 4.1. *There is no polynomial-time $(5/6+\epsilon)$-approximation algorithm to maximize a nonnegative symmetric submodular function, unless $P = NP$.*

Proof. Consider an instance of Max E4-Lin-2, a system of m parity equations, each on 4 boolean variables. Let's define two elements for each variable, T_i and F_i, corresponding to variable x_i being either true or false. For each equation e on variables (x_i, x_j, x_k, x_ℓ), we define a function $g_e(S)$. (This is our "submodular gadget".) Let $S' = S \cap \{T_i, F_i, T_j, F_j, T_k, F_k, T_\ell, F_\ell\}$. We say that S' is *valid quadruple*, if it defines a boolean assignment, i.e. contains exactly one element from each pair $\{T_i, F_i\}$. The function value is determined by S', as follows:

- If $|S'| < 4$, let $g_e(S) = |S'|$. If $|S'| > 4$, let $g_e(S) = 8 - |S'|$.
- If S' is a valid quadruple satisfying e, let $g_e(S) = 4$ (a *true quadruple*).
- If S' is a valid quadruple not satisfying e, let $g_e(S) = 8/3$ (a *false quadruple*).
- If $|S'| = 4$ but S' is not a valid quadruple, let $g_e(S) = 10/3$ (an *invalid quadruple*).

It can be verified that this is a submodular function, using the structure of the parity constraint. We define $f(S) = \sum_{e \in \mathcal{E}} g_e(S)$ by taking a sum over all equations. This is again a nonnegative submodular function. Observe that for each equation, it is more profitable to choose an invalid assignment than a valid assignment which does not satisfy the equation. Nevertheless, we claim that WLOG the maximum is obtained by selecting exactly one of T_i, F_i for each variable: Consider a set S and call a variable *undecided*, if S contains both or neither of T_i, F_i. For each equation with an undecided variable, we get value at most 10/3. Now, modify S by randomly selecting exactly one of T_i, F_i for each undecided variable. The new set S' induces a valid assignment to all variables. For equations which had a valid assignment already in S, the value does not change. Each equation which had an undecided varible is satisfied by S' with probability 1/2. Therefore, the expected value for each such equation is $(8/3+4)/2 = 10/3$, at least as before, and $\mathbf{E}[f(S')] \geq f(S)$. Hence there must exist a set S' such that $f(S') \geq f(S)$ and S' induces a valid assignment. Consequently, we have $OPT = \max f(S) = (8/3)m + (4/3)\#SAT$ where $\#SAT$ is the maximum number of satisfiable equations. Since it is NP-hard to distinguish whether $\#SAT \geq (1-\epsilon)m$ or $\#SAT \leq (1/2+\epsilon)m$, it is also NP-hard to distinguish between $OPT \geq (4-\epsilon)m$ and $OPT \leq (10/3+\epsilon)m$. □

In the case of general nonnegative submodular functions, we improve the hardness threshold to 3/4. This hardness result is slightly more involved. It requires certain properties of Håstad's 3-bit verifier, implying that Max E3-Lin-2 is NP-hard to approximate even for linear systems of a special structure. We formalize this in the following lemma, which is needed to prove Theorem 4.3.

Lemma 4.2. *Fix any $\epsilon > 0$ and consider systems of weighted linear equations (of total weight 1) over boolean variables, partitioned into $\mathcal{X}$ and $\mathcal{Y}$, so that each equation contains 1 variable $x_i \in \mathcal{X}$ and 2 variables $y_j, y_k \in \mathcal{Y}$. Define a matrix $P \in [0,1]^{\mathcal{Y} \times \mathcal{Y}}$ where P_{jk} is the weight of all equations where the first variable from $\mathcal{Y}$ is y_j and the second variable is y_k. Then it's NP-hard to decide whether there is a solution satisfying equations of weight at least $1-\epsilon$ or whether any solution satisfies equations of weight at most $1/2+\epsilon$, even in the special case where P is positive semidefinite.*

Theorem 4.3. *There is no polynomial-time $(3/4+\epsilon)$-approximation algorithm to maximize a nonnegative submodular function, representable as a sum of functions on a constant number of elements, unless $P = NP$.*

We defer the proofs to Appendix B.

4.2 Query complexity results

Finally, we prove that our 1/2-approximation for symmetric submodular functions is optimal in the value oracle model. First, we present a similar result for the "random set" model, which illustrates some of the ideas needed for the more general result.

Proposition 4.4. *For any $\delta > 0$, there is $\epsilon > 0$ such that for any (random) sequence of queries $\mathcal{Q} \subseteq 2^X$, $|\mathcal{Q}| \le 2^{\epsilon n}$, there is a nonnegative submodular function f such that (with high probability) for all queries $Q \in \mathcal{Q}$,*

$$f(Q) \le \left(\frac{1}{4} + \delta\right) OPT.$$

Proof. Let $\epsilon = \delta^2/32$ and fix a sequence $\mathcal{Q} \subseteq 2^X$ of $2^{\epsilon n}$ queries. We prove the existence of f by the probabilistic method. Consider functions corresponding to cuts in a complete bipartite directed graph on (C, D), $f_C(S) = |S \cap C| \cdot |\bar{S} \cap D|$. We choose a uniformly random $C \subseteq X$ and $D = X \setminus C$. The idea is that for any query, a typical C bisects both Q and its complement, which means that $f_C(Q)$ is roughly $\frac{1}{4}OPT$. We call a query $Q \in \mathcal{Q}$ "successful", if $f_C(Q) > (\frac{1}{4} + \delta)OPT$. Our goal is to prove that with high probability, C avoids any successful query.

We use Chernoff's bound: For any set $A \subseteq X$ of size a,

$$\Pr[|A \cap C| > \frac{1}{2}(1+\delta)|A|] = \Pr[|A \cap C| < \frac{1}{2}(1-\delta)|A|] < e^{-\delta^2 a/2}.$$

With probability at least $1 - 2e^{-2\delta^2 n}$, the size of C is in $[(\frac{1}{2} - \delta)n, (\frac{1}{2} + \delta)n]$, so we can assume this is the case. We have $OPT \ge (\frac{1}{4} - \delta^2)n^2 \ge \frac{1}{4}n^2/(1+\delta)$ (for small $\delta > 0$). No query can achieve $f_C(Q) > (\frac{1}{4}+\delta)OPT \ge \frac{1}{16}n^2$ unless $|Q| \in [\frac{1}{16}n, \frac{15}{16}n]$, so we can assume this is the case for all queries. By Chernoff's bound, $\Pr[|Q \cap C| > \frac{1}{2}(1+\delta)|Q|] < e^{-\delta^2 n/32}$ and $\Pr[|\bar{Q} \cap D| > \frac{1}{2}(1+\delta)|\bar{Q}|] < e^{-\delta^2 n/32}$. If neither of these events occurs, the query is not successful, since $f_C(Q) = |Q \cap C| \cdot |\bar{Q} \cap D| < \frac{1}{4}(1+\delta)^2|Q| \cdot |\bar{Q}| \le \frac{1}{16}(1+\delta)^2 n^2 \le \frac{1}{4}(1+\delta)^3 OPT \le (\frac{1}{4} + \delta)OPT$.

For now, fix a sequence of queries. By the union bound, we get that the probability that any query is successful is at most $2^{\epsilon n} 2e^{-\delta^2 n/32} = 2(2/e)^{\epsilon n}$. Thus with high probability, there is no successful query for C. Even for a random sequence, the probabilistic bound still holds by averaging over all possible sequences of queries. We can fix any C for which the bound is valid, and then the claim of the lemma holds for the submodular function f_C. □

This means that in the model where an algorithm only samples a sequence of polynomially many sets and returns the one of maximal value, we cannot improve our 1/4-approximation (Section 2). Surprisingly, this example can be modified for the model of adaptive algorithms with value queries, to show that our 1/2-approximation for symmetric submodular functions is optimal, even among adaptive algorithms!

Theorem 4.5. *For any $\epsilon > 0$, there are instances of nonnegative symmetric submodular maximization, such that there is no (adaptive, possibly randomized) algorithm using less than $e^{\epsilon^2 n/16}$ queries that always finds a solution of expected value at least $(1/2+\epsilon)OPT$.*

Proof. We construct a nonnegative symmetric submodular function on $[n] = C \cup D$, $|C| = |D| = n/2$, which has the following properties:

- $f(S)$ depends only on $k = |S \cap C|$ and $\ell = |S \cap D|$. Henceforth, we write $f(k, \ell)$ to denote the value of any such set.
- When $|k - \ell| \leq \epsilon n$, the function has the form
$$f(k, \ell) = (k + \ell)(n - k - \ell) = |S|(n - |S|);$$
i.e., it is equal to the cut function of a complete graph. The value depends only on the size of S, and the maximum attained by such sets is $\frac{1}{4}n^2$.
- When $|k - \ell| > \epsilon n$, the function has the form
$$f(k, \ell) = k(n - 2\ell) + (n - 2k)\ell - O(\epsilon n^2),$$
close to the cut function of a complete bipartite graph on (C, D) with edge weights 2. The maximum in this range is $OPT = \frac{1}{2}n^2(1 - O(\epsilon))$, attained for $k = n/2, \ell = 0$ (or vice versa).

If we construct such a function, we can argue as follows. Consider any algorithm, for now deterministic. (For a randomized algorithm, let's condition on its random bits.) Let the partition (C, D) be random and unknown to the algorithm. The algorithm issues some queries Q to the value oracle. Call Q "unbalanced", if $|Q \cap C|$ differs from $|Q \cap D|$ by more than ϵn. For any query Q, the probability that Q is unbalanced is at most $e^{-\epsilon^2 n/8}$, by standard bounds. Therefore, for any fixed sequence of $e^{\epsilon^2 n/16}$ queries, the probability that any query is unbalanced is still at most $e^{\epsilon^2 n/16} \cdot e^{-\epsilon^2 n/8} = e^{-\epsilon^2 n/16}$. As long as queries are balanced, the algorithm gets the same answer regardless of (C, D). Hence, it follows the same path of computation and issues the same queries. With probability at least $1 - e^{-\epsilon^2 n/16}$, all its queries will be balanced and it will never find out any information about the partition (C, D). For a randomized algorithm, we can now average over its random choices; still, with probability at least $1 - e^{-\epsilon^2 n/16}$ the algorithm will never query any unbalanced set.

Alternatively, consider a function $g(S)$ which is defined by $g(S) = |S|(n - |S|)$ for *all sets* S. We proved that with high probability, the algorithm will never query a set where $f(S) \neq g(S)$ and hence cannot distinguish between the two instances. However, $\max_S f(S) = \frac{1}{2}n^2(1 - -O(\epsilon))$, while $\max_S g(S) = \frac{1}{4}n^2$. This means that there is no $(1/2 + \epsilon)$-approximation algorithm with a subexponential number of queries, for any $\epsilon > 0$.

It remains to construct the function $f(k, \ell)$ and prove its submodularity. For convenience, assume that ϵn is an integer. In the range where $|k - \ell| \leq \epsilon n$, we already defined $f(k, \ell) = (k + \ell)(n - k - \ell)$. In the range where $|k - \ell| \geq \epsilon n$, let us define
$$f(k, \ell) = k(n - 2\ell) + (n - 2k)\ell + \epsilon^2 n^2 - 2\epsilon n|k - \ell|.$$
The $O(\epsilon n^2)$ terms are chosen so that $f(k, \ell)$ is a smooth function on the boundary of the two regions. E.g., for $k - \ell = \epsilon n$, we get $f(k, \ell) = (2k - \epsilon n)(n - 2k + \epsilon n)$ for both expressions. Moreover, the marginal values also extend smoothly. Consider an element $i \in C$ (for $i \in D$ the situation is symmetric). The marginal value of i added to a set S is $f(S + i) - f(S) = f(k + 1, \ell) - f(k, \ell)$. We split into three cases:

- If $k - \ell < -\epsilon n$, we have $f(k + 1, \ell) - f(k, \ell) = (n - 2\ell) + (-2\ell) + 2\epsilon n = (1 + 2\epsilon)n - 4\ell$.
- If $-\epsilon n \leq k - \ell < \epsilon n$, we have $f(k+1, \ell) - f(k, \ell) = (k+1+\ell)(n-k-1-\ell) - (k+\ell)(n-k-\ell) = (n - k - 1 - \ell) - (k + 1 + \ell) = n - 2k - 2\ell - 2$. In this range, this is between $(1 \pm 2\epsilon) - 4\ell$.
- If $k - \ell \geq \epsilon n$, we have $f(k + 1, \ell) - f(k, \ell) = (n - 2\ell) + (-2\ell) - 2\epsilon n = (1 - 2\epsilon)n - 4\ell$.

Now it's easy to see that the marginal value is decreasing in both k and ℓ, in each range and also across ranges. $\square$

Acknowledgements. The second author thanks Maxim Sviridenko for pointing out some related work.

References

[1] A. Ageev and M. Sviridenko. An 0.828 approximation algorithm for uncapacitated facility location problem, *Discrete Applied Mathematics* 93:2-3, (1999) 149–156.

[2] P. Alimonti. Non-oblivious Local Search for MAX 2-CCSP with application to MAX DICUT, *Proc. of the 23rd International Workshop on Graph-theoretic Concepts in Computer Science* (1997).

[3] G. Calinescu, C. Chekuri, M. Pál and J. Vondrák. Maximizing a monotone submodular function under a matroid constraint, *to appear in IPCO 2007.*

[4] V. Cherenin. Solving some combinatorial problems of optimal planning by the method of successive calculations, *Proc. of the Conference of Experiences and Perspectives of the Applications of Mathematical Methods and Electronic Computers in Planning* (in Russian), Mimeograph, Novosibirsk (1962).

[5] G. Cornuejols, M. Fischer and G. Nemhauser. Location of bank accounts to optimize float: an analytic study of exact and approximation algorithms, *Management Science*, 23 (1977), 789–810.

[6] G. Cornuejols, M. Fischer and G. Nemhauser. On the uncapacitated location problem, *Annals of Discrete Math* 1 (1977), 163–178.

[7] G. P. Cornuejols, G. L. Nemhauser, and L. A. Wolsey. The uncapacitated facility location problem. In *Discrete Location Theory* (1990), 119–171.

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

[9] U. Feige and M. X. Goemans. Approximating the value of two-prover systems, with applications to MAX-2SAT and MAX-DICUT. *Proc. of the 3rd Israel Symposium on Theory and Computing Systems*, Tel Aviv (1995), 182–189.

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

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

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

[13] S. Fujishige. Canonical decompositions of symmetric submodular systems, *Discrete Applied Mathematics* 5 (1983), 175–190.

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

[15] A. Frank. Matroids and submodular functions, *Annotated Biblographies in Combinatorial Optimization* (1997), 65-80.

[16] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, *Journal of ACM* 42 (1995), 1115–1145.

[17] B. Goldengorin, G. Sierksma, G. Tijsssen, and M. Tso. The data correcting algorithm for the minimization of supermodular functions, *Management Science*, 45:11 (1999), 1539–1551.

[18] B. Goldengorin, G. Tijsssen, and M. Tso. The maximization of submodular Functions: Old and new proofs for the correctness of the dichotomy algorithm, *SOM Report*, University of Groningen (1999).

[19] V. Guruswami. Inapproximability results for set splitting and satisfiability problems with no mixed clauses, *Algorithmica* 38 (2004), 451–469.

[20] V. Guruswami and Subhash Khot. Hardness of Max 3-SAT with no mixed clauses, *CCC 2005*.

[21] E. Halperin and U. Zwick. Combinatorial approximation algorithms for the maximum directed cut problem, *Proc. of 12th SODA* (2001), 1–7.

[22] J. Håstad. Some optimal inapproximability results, *Journal of ACM* 48 (2001): 798-869.

[23] V.R. Khachaturov. Mathematical methods of regional programming (in Russian), *Nauka, Moscow* (1989).

[24] S. Khot, G. Kindler. E. Mossel and R. O'Donnell. Optimal inapproximability results for MAX-CUT and other two-variable CSPs? *Proc. of 45th FOCS* (2004), 146–154.

[25] H. Lee, G. Nemhauser, and Y. Wang. Maximizing a submodular function by integer programming: Polyhedral results for the quadratic case, *European Journal of Operational Research 94*, 154-166.

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

[27] M. Minoux. Accelerated greedy algorithms for maximizing submodular functions, J. Stoer, ed., *Actes Congress IFIP, Springer Verlag, Berlin* (1977), 234–243.

[28] E. Mossel, R. O'Donnell and K. Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality, *Proc. of 46th FOCS* (2005), 21–30.

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

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

[31] T. Robertazzi and S. Schwartz. An accelated sequential algorithm for producing D-optimal designs, *SIAM Journal on Scientific and Statistical Computing* 10, 341–359.

[32] M. Queyranne. A combinatorial algorithm for minimizing symmetric submodular functions, *Proc. of 6th SODA* (1995), 98–101.

[33] A. Schäfer, M. Yannakakis. Simple local search problems that are hard to solve. *SIAM J. Comput.* 20:1 (1991), 56–87.

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

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

A A randomized non-adaptive $1/3$-approximation

We present a non-adaptive algorithm which achieves an approximation factor of $1/3$ for any non-negative submodular function. The intuition behind the algorithm comes from the Max Di-Cut problem: When does a random cut achieve only $1/4$ of the optimum? This is if and only if the optimum contains all the directed edges of the graph, i.e. the vertices can be partitioned into $V = A \cup B$ so that all edges of the graph are directed from A to B. However, in this case it's easy to find the optimal solution, by a local test on the in-degree and out-degree of each vertex. In the language of submodular function maximization, this means that elements can be easily partitioned into those whose inclusion in S always increases the value of $f(S)$, and those which always decrease $f(S)$. Our generalization of this local test is the following.

Definition A.1. *Let $R = X(1/2)$ denote a uniformly random subset of X. For each element x, define*

$$\omega(x) = \mathbf{E}[f(R \cup \{x\}) - f(R \setminus \{x\})].$$

Note that these values can be estimated by random sampling, up to an error polynomially small relative to $\max_{R,x} |f(R \cup \{x\}) - f(R \setminus \{x\})| \le OPT$. This is sufficient for our purposes; in the following, we assume that we have estimates $\tilde{\omega}(x)$ such that $|\omega(x) - \tilde{\omega}(x)| \le OPT/n^2$.

The Non-Adaptive Algorithm: NA.

- Use random sampling to find $\tilde{\omega}(x)$ for each $x \in X$.
- Independently, sample a random set $R = X(1/2)$.
- With prob. $8/9$, return R.
- With prob. $1/9$, return
$$A = \{x \in X : \tilde{\omega}(x) > 0\}.$$

Theorem A.2. *For any nonnegative submodular function, Algorithm NA achieves expected value at least $(1/3 - o(1))$ OPT.*

Proof. Let $A = \{x \in X : \tilde{\omega}(x) > 0\}$ and $B = X \setminus A = \{x \in X : \tilde{\omega}(x) \le 0\}$. Therefore we have $\omega(x) \ge -OPT/n^2$ for any $x \in A$ and $\omega(x) \le OPT/n^2$ for any $x \in B$. We shall keep in mind that (A, B) is a partition of all the elements, and so we have $(A \cap T) \cup (B \cap T) = T$ for any set T, etc.

Denote by C the optimal set, $f(C) = OPT$. Let $f(A) = \alpha$, $f(B \cap C) = \beta$ and $f(B \cup C) = \gamma$. By submodularity, we have

$$\alpha + \beta = f(A) + f(B \cap C) \ge f(\emptyset) + f(A \cup (B \cap C)) \ge f(A \cup C)$$

and

$$\alpha + \beta + \gamma \ge f(A \cup C) + f(B \cup C) \ge f(X) + f(C) \ge OPT.$$

Therefore, either α, the value of A, is at least $OPT/3$, or else one of β and γ is at least $OPT/3$; we prove that then $\mathbf{E}[f(R)] \ge OPT/3$ as well.

Let's start with $\beta = f(B \cap C)$. Instead of $\mathbf{E}[f(R)]$, we show that it's enough to estimate $\mathbf{E}[f(R \cup (B \cap C))]$. Recall that for any $x \in B$, we have $\omega(x) = \mathbf{E}[f(R \cup \{x\}) - f(R \setminus \{x\})] \le OPT/n^2$. Consequently, we also have $\mathbf{E}[f(R \cup \{x\}) - f(R)] = \frac{1}{2}\omega(x) \le OPT/(2n^2)$. Let's order the elements of $B \cap C = \{b_1, \ldots, b_\ell\}$ and write

$$f(R \cup (B \cap C)) = f(R) + \sum_{j=1}^{\ell} (f(R \cup \{b_1, \ldots, b_j\}) - f(R \cup \{b_1, \ldots, b_{j-1}\})).$$

By the property of decreasing marginal values, we get

$$f(R \cup (B \cap C)) \leq f(R) + \sum_{j=1}^{\ell}(f(R \cup \{b_j\}) - f(R)) = f(R) + \sum_{x \in B \cap C}(f(R \cup \{x\}) - f(R))$$

and therefore

$$\begin{aligned}
\mathbf{E}[f(R \cup (B \cap C))] &\leq \mathbf{E}[f(R)] + \sum_{x \in B \cap C} \mathbf{E}[f(R \cup \{x\}) - f(R)] \\
&\leq \mathbf{E}[f(R)] + |B \cap C| \frac{OPT}{2n^2} \leq \mathbf{E}[f(R)] + \frac{OPT}{2n}.
\end{aligned}$$

So it's enough to lower-bound $\mathbf{E}[f(R \cup (B \cap C))]$. We do this by defining a new submodular function, $g(R) = f(R \cup (B \cap C))$, and applying Lemma 2.3 to $\mathbf{E}[g(R)] = \mathbf{E}[g(C(1/2) \cup \bar{C}(1/2))]$. The lemma implies that

$$\begin{aligned}
\mathbf{E}[f(R \cup (B \cap C))] &\geq \frac{1}{4}g(\emptyset) + \frac{1}{4}g(C) + \frac{1}{4}g(\bar{C}) + \frac{1}{4}g(X) \\
&\geq \frac{1}{4}g(\emptyset) + \frac{1}{4}g(C) = \frac{1}{4}f(B \cap C) + \frac{1}{4}f(C) \\
&= \frac{\beta}{4} + \frac{OPT}{4}.
\end{aligned}$$

Note that $\beta \geq OPT/3$ implies $\mathbf{E}[f(R \cup (B \cap C))] \geq OPT/3$. Symmetrically, we show a similar analysis for $\mathbf{E}[f(R \cap (B \cup C))]$. Now we use the fact that for any $x \in A$, $\mathbf{E}[f(R) - f(R \setminus \{x\})] = \frac{1}{2}\omega(x) \geq -OPT/(2n^2)$. Let $A \setminus C = \{a_1, a_2, \ldots, a_k\}$ and write

$$\begin{aligned}
f(R) &= f(R \setminus (A \setminus C)) + \sum_{j=1}^{k}(f(R \setminus \{a_{j+1}, \ldots, a_k\}) - f(R \setminus \{a_j, \ldots, a_k\})) \\
&\geq f(R \setminus (A \setminus C)) + \sum_{j=1}^{k}(f(R) - f(R \setminus \{a_j\})
\end{aligned}$$

using the condition of decreasing marginal values. Note that $R \setminus (A \setminus C) = R \cap (B \cup C)$. By taking the expectation,

$$\begin{aligned}
\mathbf{E}[f(R)] &\geq \mathbf{E}[f(R \cap (B \cup C))] + \sum_{j=1}^{k} \mathbf{E}[f(R) - f(R \setminus \{a_j\})] \\
&= \mathbf{E}[f(R \cap (B \cup C))] - |A \setminus C| \frac{OPT}{2n^2} \geq \mathbf{E}[f(R \cap (B \cup C))] - \frac{OPT}{2n}.
\end{aligned}$$

Again, we estimate $\mathbf{E}[f(R \cap (B \cup C))] = \mathbf{E}[f(C(1/2) \cup (B \setminus C)(1/2))]$ using Lemma 2.3. We get

$$\begin{aligned}
\mathbf{E}[f(R \cap (B \cup C))] &\geq \frac{1}{4}f(\emptyset) + \frac{1}{4}f(C) + \frac{1}{4}f(B \setminus C) + \frac{1}{4}f(B \cup C) \\
&\geq \frac{OPT}{4} + \frac{\gamma}{4}.
\end{aligned}$$

Now we combine our estimates for $\mathbf{E}[f(R)]$:

$$\mathbf{E}[f(R)] + \frac{OPT}{2n} \geq \frac{1}{2}\mathbf{E}[f(R \cup (B \cap C))] + \frac{1}{2}\mathbf{E}[f(R \cap (B \cup C))] \geq \frac{OPT}{4} + \frac{\beta}{8} + \frac{\gamma}{8}.$$

Finally, the expected value obtained by the algorithm is

$$\frac{8}{9}\mathbf{E}[f(R)] + \frac{1}{9}f(A) \geq \frac{2}{9}OPT - \frac{4}{9n}OPT + \frac{\beta}{9} + \frac{\gamma}{9} + \frac{\alpha}{9} \geq \left(\frac{1}{3} - \frac{4}{9n}\right)OPT$$

since $\alpha + \beta + \gamma \geq OPT$. □

B NP-hardness of $(3/4+\epsilon)$-approximation

Here, we show the missing details of Section 4.1, namely the proof that it is NP-hard to achieve a $(3/4+\epsilon)$-approximation.

Proof of Lemma 4.2. We show that the system of equations arising from Håstad's 3-bit verifier (see [22], pages 24-25) in fact satisfies the properties that we need. In his notation, the equations are generated by choosing $f \in \mathcal{F}_U$ and $g_1, g_2 \in \mathcal{F}_W$ where U, W, $U \subset W$, are randomly chosen and $\mathcal{F}_U, \mathcal{F}_W$ are the spaces of all ± 1 functions on $\{-1,+1\}^U$ and $\{-1,+1\}^W$, respectively. The equation corresponds to a 3-bit test on f, g_1, g_2 and its weight is the probability that the verifier performs this particular test. One variable is associated with $f \in \mathcal{F}_U$, indexing a bit in the Long Code of the first prover, and two variables are associated with $g_1, g_2 \in \mathcal{F}_W$, indexing bits in the Long Code of the second prover. This defines a natural partition of variables into $\mathcal{X}$ and $\mathcal{Y}$.

The actual variables appearing in the equations are determined by the folding convention; for the second prover, let's denote them by $y_j = \phi(g_1), y_k = \phi(g_2)$. The particular convention will not matter to us, as long as it is the same for both g_1 and g_2 (which is the case in [22]). Let P_{jk} be the probability that the selected variables corresponding to the second prover are y_j and y_k. Let $P_{jk}^{U,W}$ be the same probability, conditioned on a particular choice of U, W. Since P is a positive linear combination of $P^{U,W}$, it suffices to prove that each $P^{U,W}$ is positive semidefinite. The way that g_1, g_2 are generated (for given U, W) is that $g_1 : \{-1,+1\}^W \to \{-1,+1\}$ is uniformly random and $g_2(y) = g_1(y) f(y|_U) \mu(y)$, where $f : \{-1,+1\}^U \to \{-1,+1\}$ uniformly random and $\mu : \{-1,+1\}^W \to \{-1,+1\}$ is a "random noise", where $\mu(x) = 1$ with probability $1-\epsilon$ and -1 with probability ϵ. The value of ϵ will be very small, certainly $\epsilon < 1/2$.

Let's choose an arbitrary function $A : \mathcal{Y} \to \mathbb{R}$ and analyze

$$\sum_{jk} P_{jk}^{U,W} A(y_j) A(y_k) = \mathbf{E}_{g_1,g_2}[A(\phi(g_1))A(\phi(g_2))] = \mathbf{E}_{g_1,f,\mu}[A(\phi(g_1))A(\phi(g_1 f \mu))]$$

where g_1, f, μ are sampled as described above. If we prove that this quantity is always nonnegative, then $P^{U,W}$ is positive semidefinite. Let $B : \mathcal{F}_W \to \mathbb{R}$, $B = A \circ \phi$; i.e., we want to prove $\mathbf{E}[B(g_1)B(g_1 f \mu)] \geq 0$. We can expand B using its Fourier transform,

$$B(g) = \sum_{\alpha \subseteq \{-1,+1\}^W} \hat{B}(\alpha) \chi_\alpha(g).$$

Here, $\chi_\alpha(g) = \prod_{x \in \alpha} g(x)$ are the Fourier basis functions. We obtain

$$\mathbf{E}[B(g_1)B(g_1 f \mu)] = \sum_{\alpha,\beta \subseteq \{-1,+1\}^W} \mathbf{E}[\hat{B}(\alpha)\chi_\alpha(g_1)\hat{B}(\beta)\chi_\beta(g_1 f \mu)]$$

$$= \sum_{\alpha,\beta \subseteq \{-1,+1\}^W} \hat{B}(\alpha)\hat{B}(\beta) \prod_{x \in \alpha \Delta \beta} \mathbf{E}_{g_1}[g_1(x)] \mathbf{E}_f[\prod_{y \in \beta} f(y|_U)] \prod_{z \in \beta} \mathbf{E}_\mu[\mu(z)].$$

The terms for $\alpha \neq \beta$ are zero, since then $\mathbf{E}_{g_1}[g_1(x)] = 0$ for each $x \in \alpha \Delta \beta$. Therefore,

$$\mathbf{E}[B(g_1)B(g_1 f \mu)] = \sum_{\beta \subseteq \{-1,+1\}^W} \hat{B}^2(\beta)\ \mathbf{E}_f[\prod_{y \in \beta} f(y|_U)] \prod_{z \in \beta} \mathbf{E}_\mu[\mu(z)].$$

Now all the factors are nonnegative, since $\mathbf{E}_\mu[\mu(z)] = 1 - 2\epsilon > 0$ for every z and $\mathbf{E}_f[\prod_{y \in \beta} f(y|_U)] = 1$ or 0, depending on whether every string in $\{-1,+1\}^U$ is the projection of an even number of strings in β (in which case the product is 1) or not (in which case the expectation gives 0 by symmetry). To conclude,

$$\sum_{j,k} P_{jk}^{U,W} A(y_j) A(y_k) = \mathbf{E}[B(g_1)B(g_1 f \mu)] \geq 0$$

for any $A : \mathcal{Y} \to \mathbb{R}$, which means that each $P^{U,W}$ and consequently P is positive semidefinite. □

Proof of Theorem 4.3. We use a reduction from a system of linear equations as in Lemma 4.2. For each variable $x_i \in \mathcal{X}$, we have two elements T_i, F_i and for each variable $y_j \in \mathcal{Y}$, we have two elements $\tilde{T}_j, \tilde{F}_j$. Denote the set of equations by $\mathcal{E}$. Each equation e contains one variable from $\mathcal{X}$ and two variables from $\mathcal{Y}$. For each $e \in \mathcal{E}$, we define a submodular function $g_e(S)$ tailored to this structure. Assume that $S \subseteq \{T_i, F_i, \tilde{T}_j, \tilde{F}_j, \tilde{T}_k, \tilde{F}_k\}$, the elements corresponding to this equation; g_e does not depend on other than these 6 elements. We say that S is a valid triple, if it contains exactly one of each $\{T_i, F_i\}$.

- The value of each singleton T_i, F_i corresponding to a variable in $\mathcal{X}$ is 1.
- The value of each singleton $\tilde{T}_j, \tilde{F}_j$ corresponding to a variable in $\mathcal{Y}$ is 1/2.
- For $|S| < 3$, $g_e(S)$ is the sum of its singletons, except $g_e(\{T_i, F_i\}) = 1$ (*a weak pair*).
- For $|S| > 3$, $g_e(S) = g_e(\bar{S})$.
- If S is a valid triple satisfying e, let $g_e(S) = 2$ (*true triple*).
- If S is a valid triple not satisfying e, let $g_e(S) = 1$ (*false triple*).
- If S is an invalid triple containing exactly one of $\{T_i, F_i\}$ then $g_e(S) = 2$ (*type I*).
- If S is an invalid triple containing both/neither of $\{T_i, F_i\}$ then $g_e(S) = 3/2$ (*type II*).

The analysis of this gadget is more involved. We first emphasize the important points. A true triple gives value 2, while a false triple gives value 1. For invalid assignments of value 3/2, we can argue as before that a random valid assignment achieves expected value 3/2 as well, so we might as well choose a valid assignment. However, in this gadget we also have invalid triples of value 2 (type I). (We cannot avoid this due to submodularity.) Still, we prove that the optimum is attained for a valid boolean assignment. The main argument is, roughly, that if there are many invalid triples of type I, there must be also many equations where we get value only 1 (a weak pair). For this, we use the positive semidefinite property from Lemma 4.2.

Verifying that $g_e(S)$ is submodular is somewhat tedious. We have to check marginal values for two types of elements. First, consider T_i (or equivalently F_i), associated with a variable in $\mathcal{X}$. The marginal value $g_A(T_i)$ is equal to 1 if A does not contain F_i and contains at most two elements for variables in $\mathcal{Y}$, except when these elements would form a false triple with T_i; then $g_A(T_i) = 0$. Also, $g_A(T_i) = 0$ if A does not contain F_i and contains at least three elements for $\mathcal{Y}$, or if A contains F_i and at most two elements for $\mathcal{Y}$, except when these elements would form a true triple with T_i; then $g_A(T_i) = -1$. Finally, $g_A(T_i) = -1$ if A contains F_i and at least three elements for $\mathcal{Y}$. Hence, the marginal value depends in a monotone way on the subset A.

For an element like $\tilde{T}_j$, associated with a variable in $\mathcal{Y}$, we have $g_A(\tilde{T}_j) = 1/2$ if A does not contain any element for $\mathcal{X}$ and contains at most two elements for $\mathcal{Y}$, or A contains one element for $\mathcal{X}$ and at most one element for $\mathcal{Y}$ (but not forming a false triple with $\tilde{T}_j$), or A is a false triple, or A contains both elements for $\mathcal{X}$ and no other element for $\mathcal{Y}$. In all other cases, $g_A(\tilde{T}_j) = -1/2$. Here, the only way submodularity could be violated is that $g_A(\tilde{T}_j) < f_B(\tilde{T}_j)$ for $A \subset B$ where A forms a false triple with $\tilde{T}_j$ and B is a false triple - but then $B = A \cup \{\tilde{T}_j\}$, contradiction.

We define $f(S) = \sum_{e \in \mathcal{E}} w(e) g_e(S)$ where $w(e)$ is the weight of equation e. We claim that $\max f(S) = 1 + \max w_{SAT}$, where w_{SAT} is the weight of satisfied equations. First, for a given boolean assignment, the corresponding set S selecting T_i or F_i for each variable achieves value $f(S) = w_{SAT} \cdot 2 + (1 - w_{SAT}) \cdot 1 = 1 + w_{SAT}$. The non-trivial part is proving that the optimum $f(S)$ is attained for a set inducing a valid boolean assignment.

Consider any set S and define $V : \mathcal{E} \to \{-1, 0, +1\}$ where $V(e) = +1$ if S induces a satisfying assignment to equation e, $V(e) = -1$ if S induces a non-satisfying assignment to e and $V(e) = 0$ if S induces an invalid assignment to e. Also, define $A : \mathcal{Y} \to \{-1, 0, +1\}$, where $A(y_j) = |S \cap \{\tilde{T}_j, \tilde{F}_j\}| - 1$, i.e. $A(y_j) = 0$ if S induces a valid assignment to y_j, and $A(y_j) = \pm 1$ if S contains both/neither of $\tilde{T}_j, \tilde{F}_j$. Observe that for an equation e whose $\mathcal{Y}$-variables are y_j, y_k, only one of $V(e)$ and $A(y_j)A(y_k)$ can be nonzero. The gadget $g_e(S)$ is designed in such a way that

$$g_e(S) \leq \frac{1}{2}(3 - A(y_j)A(y_k) + V(e)).$$

This can be checked case by case: for valid assignments, $A(y_j)A(y_k) = 0$ and we get value 2 or 1 depending on $V(e) = \pm 1$. For invalid assignments, $V(e) = 0$; if at least one of the variables y_j, y_k has a valid assignment, then $A(y_j)A(y_k) = 0$ and we can get at most $3/2$ (an invalid triple of type II). If both y_j, y_k are invalid and $A(y_j)A(y_k) = 1$, then we can get only 1 (a weak pair or its complement) and if $A(y_j)A(y_k) = -1$, we can get 2 (an invalid triple of type I). The total value is

$$f(S) = \sum_{e \in \mathcal{E}} w(e) g_e(S) \leq \sum_{j,k} \sum_{e=(x_i,y_j,y_k)} w(e) \cdot \frac{1}{2}(3 - A(y_j)A(y_k) + V(e)).$$

Now we use the positive semidefinite property of our linear system, which means that $\sum_{j,k} \sum_{e=(x,y_j,y_k)} w(e)A(y_j)A(y_k) = \sum_{j,k} P_{jk} A(y_j)A(y_k) \geq 0$ for any function A. Hence, $f(S) \leq \frac{1}{2} \sum_{e \in \mathcal{E}} w(e)(3 + V(e))$. Now, modify S into a valid boolean assignment by choosing randomly one of T_i, F_i for all variables such that S contains both/neither of T_i, F_i. Denote the new set by S' and the equations containing any randomly chosen variable by $\mathcal{R}$. We satisfy each equation in $\mathcal{R}$ with probability $1/2$, which gives expected value $3/2$ for each such equation, while the value for other equations remains unchanged.

$$\mathbf{E}[f(S')] = \frac{3}{2} \sum_{e \in \mathcal{R}} w(e) + \frac{1}{2} \sum_{e \in \mathcal{E} \setminus \mathcal{R}} w(e)(3 + V(e)) = \frac{1}{2} \sum_{e \in \mathcal{E}} w(e)(3 + V(e)) \geq f(S).$$

This means that there is a set S' of optimal value, inducing a valid boolean assignment. $\square$