# A Polynomial Lower Bound on the Number of Rounds for Parallel Submodular Function Minimization

Deeparnab Chakrabarty
*Dartmouth College*
*Hanover, NH*
*deeparnab@dartmouth.edu*

Yu Chen
*University of Pennsylvania*
*Philadelphia, PA*
*chenyu2@cis.upenn.edu*

Sanjeev Khanna
*University of Pennsylvania*
*Philadelphia, PA*
*sanjeev@cis.upenn.edu*

2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS) | 978-1-6654-2055-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/FOCS52979.2021.00013

***Abstract*—The problem of minimizing a submodular function (SFM) is a common generalization of several fundamental combinatorial optimization problems, including minimum $s$-$t$ cuts in graphs and matroid intersection. It is well-known that a submodular function can be minimized with only $\mathrm{poly}(N)$ function evaluation queries where $N$ denotes the universe size. However, all known polynomial query algorithms for SFM are highly adaptive, requiring at least $N$ rounds of adaptivity. A natural question is if SFM can be efficiently solved in a highly parallel manner, namely, with $\mathrm{poly}(N)$ queries using only poly-logarithmic rounds of adaptivity. An important step towards understanding the adaptivity needed to solve SFM efficiently was taken in the very recent work of Balkanski and Singer who showed that any SFM algorithm with $\mathrm{poly}(N)$ queries. This left open the possibility of efficient SFM algorithms with poly-logarithmic rounds of adaptivity. In this work, we strongly rule out this possibility by showing that any, possibly randomized, algorithm for submodular function minimization making $\mathrm{poly}(N)$ queries requires $\tilde{\Omega}\left(N^{1/3}\right)$ rounds of adaptivity. In fact, we show a polynomial lower bound on the number of rounds of adaptivity even for algorithms that make up to $2^{N^{1-\delta}}$ queries, for any constant $\delta > 0$.**

***Keywords*-Submodular Function Minimization; Parallel Algorithms; Lower Bounds**

## I. Introduction

A function $f : 2^U \to \mathbb{Z}$ defined over subsets of a ground set $U$ of $N$ elements is submodular if for any two sets $A \subseteq B$ and an element $e \notin B$, the *marginal* of $e$ on $A$, that is, $f(A \cup e) - f(A)$ is at least $f(B \cup e) - f(B)$. The submodular function minimization (SFM) problem is to find a subset $S$ minimizing $f(S)$ given only access to an evaluation oracle for the function that returns the function value on any specified subset. This is a fundamental discrete optimization problem which generalizes problems such as minimizing global and $s$-$t$ cuts in graphs and hypergraphs, matroid intersection, and more recently has found many applications in areas such as image segmentation [9], [10], [35] and speech analysis [29], [30].

A remarkable fact is that SFM *can* be solved in polynomial time with polynomially many calls to the evaluation oracle. This was first established by Grötschel, Lovász, and Schrijver [25] using the ellipsoid method. Since then, a lot of work [18], [27], [45], [43], [28], [13], [36], [37], [14], [19], [4], [31] has been done trying to understand the query complexity of SFM. The current best known algorithms are an $O(N^3)$-query polynomial-time and an $O(N^2 \log N)$-query exponential time algorithm by Jiang [31] building on the works [37], [19], an $\tilde{O}(N^2 \log M)$-query and time algorithm by Lee, Sidford, and Wong [37] where $|f(S)| \leq M$ for all $S \subseteq U$, and an $\tilde{O}(NM^2)$ query and time algorithm by Axelrod, Liu, and Sidford [4] improving upon [14].

All the above algorithms are *sequential*. That is, the queries made by the algorithms depend on answers to queries made earlier. More precisely, any SFM algorithm accesses the evaluation oracle in rounds, where the queries made in a certain round depend only on the answers to queries made in previous rounds. There is a trade-off between the number of queries (per round) made by the algorithm, and the number of rounds needed to find the answer : there is an obvious 1-round algorithm which makes all $2^N$ queries. All known algorithms which make $\mathrm{poly}(N)$ queries proceed in $\Omega(N)$ rounds. Can the number of rounds be substantially decreased (made poly-logarithmic in $N$) while still keeping the number of queries bounded by $\mathrm{poly}(N)$? In spirit, this is related to the $\mathbf{P}$ versus $\mathbf{NC}$ question which at a high-level asks can problems with polynomial time algorithms be solved by poly-sized circuits with poly-logarithmic depth. From a practical standpoint, given the applications of SFM to problems involving huge data and the availability of computing infrastructure to

*This extended abstract is missing several proofs which can be found in this version [1]. Since the submission of this paper to FOCS 2021, we have extended our results to prove a polynomial lower bound on rounds needed for parallel matroid intersection as well. Details of this can be found in the full version of this paper available at [2].*

2575-8454/21/$31.00 ©2021 IEEE
DOI 10.1109/FOCS52979.2021.00013

perform parallel computation, the question of parallel SFM algorithms is very relevant.

For SFM, a study of this question was initiated by Balkanski and Singer in [8] who proved that any polynomial query SFM algorithm must have $\Omega(\frac{\log N}{\log \log N})$ rounds of adaptivity. This leaves open the possibility of polynomial query poly-logarithmic round algorithms. Indeed for the related problem of submodular function *maximization* subject to cardinality constraint, in a different paper [7], Balkanski and Singer showed that the correct answer is indeed $\tilde{\Theta}(\log N)$. They proved that with polynomially many queries no constant factor approximation is possible with $o\left(\frac{\log N}{\log \log N}\right)$ rounds, while an $1/3$-approximation can be obtained in $O(\log N)$-rounds[1]. Can the situation be the same for SFM?

In this paper we answer this question in the negative. We prove a ***polynomial*** lower bound on the number of rounds of any polynomial query SFM algorithm.

**Theorem 1.** *For any constant $\delta > 0$ and any $1 \leq c \leq N^{1-\delta}$, any possibly randomized algorithm for SFM on an $N$ element universe making $\leq N^c$ evaluation oracle queries per round and succeeding with probability $\geq 2/3$ must have $\Omega\left(\frac{N^{1/3}}{(c \log N)^{1/3}}\right)$ rounds-of-adaptivity. This is true even when the range of the submodular function is $\{-N, -N+1, \ldots, N-1, N\}$, and even if the algorithm is allowed to make an additive error of $O(N^{1/3})$.*

We note that a polynomial lower bound on the number of rounds holds even if the algorithm is allowed to make $2^{N^{1-\delta}}$ queries per round for any $\delta > 0$, and the lower bound on the number of rounds is $\tilde{\Theta}(N^{1/3})$ for polynomial query algorithms. Furthermore, since the range of our functions is $[-N, +N]$ and we rule out algorithms making additive $O(N^{1/3})$-error, we also obtain an $\widetilde{\Omega}(\frac{1}{\sqrt{\varepsilon}})$-lower bound on the number of rounds for $\varepsilon$-approximate minimization of submodular functions with range $[-1, +1]$. In fact, one can modify our construction (please see full version) slightly to give an $\Omega(1/\varepsilon)$ lower bound for $\varepsilon$-additive approximation. We remark that since the functions of Balkanski and Singer in [8] when scaled to integer values have range as large as $N^{\Theta(r)}$ where $r = \Omega(\log N / \log \log N)$, it is not clear if their construction implies any non-trivial lower bound for approximate SFM. The only previous work ruling out $\varepsilon$-approximate minimizers is another work of Balkanski and Singer [6] who proved that *non-adaptive* algorithms, that is single round algorithms, cannot achieve any non-trivial approximation with polynomially many queries.

Our result shows that in the general query model, SFM cannot be solved in polynomial time in poly-logarithmic rounds, even with randomization. This is in contrast to *specific* explicitly described succinct submodular function minimization problems : global minimum cuts in an undirected graph is in $\mathbf{NC}$ [32], linear and graphic matroid intersection is in $\mathbf{RNC}$ [39], finding minimum $s$-$t$-cuts with poly-bounded capacities is in $\mathbf{RNC}$ [33], etc. Very recently, inspired by some of these special cases, Gurjar and Rathi [26] defined a class of submodular functions called *linearly representable* submodular functions and gave $\mathbf{RNC}$ algorithms for the same.

It is not very common to find examples of problems that require polynomial rounds of adaptivity to achieve a polynomial query complexity. In a thought provoking paper [34], Karp, Upfal and Wigderson considered this question. They proved that any efficient algorithm that finds a maximum independent set in a matroid with access to an *independence oracle*, that is, one which takes a subset $S$ and returns a Boolean answer of whether $S$ is independent or not, must proceed in $\widetilde{\Omega}(N^{1/3})$ rounds. On the other hand, with access to a *rank* oracle which takes $S$ and returns $r(S)$, the size of the largest independent set in $S$ (since $r(S)$ is a submodular function, this is more in lines with the evaluation oracle), there is a simple algorithm[2] which makes $N$ queries in a single round and finds the optimal answer. Our lower bound thus provides an example of the polynomial round lower bound for a much more general class of query functions.

Our lower bounding submodular functions fall in a class introduced by Balkanski and Singer [8] which we call *partition submodular functions*. Given a partition $P = (P_1, \ldots, P_r)$ of the universe $U$, the value of a partition submodular function $f(S)$ depends only on the *cardinalities* of the $|S \cap P_i|$'s. In particular, $f(S) = h(\mathbf{x})$ where $\mathbf{x}$ is an $r$-dimensional non-negative integer valued vector with $\mathbf{x}_i := |S \cap P_i|$, and $h$ is a discrete submodular function on a hypergrid. Note that when $r = 1$, the function $h$ is a univariate concave function, while $r = n$ captures general submodular functions. Thus, partition submodular functions form a nice way of capturing the complexity of a submodular function.

The functions in [8] are partition submodular and they prove an $\Omega(r)$-lower bound for their specific functions.

[1]This constant has since been made close to optimal [5], [15], [16], [21], [22], [38]; see Section I-A for more details.

[2]Order elements as $e_1, \ldots, e_N$ and query $r(\{e_1, \ldots, e_i\})$ for all $i$, and return the points at which the rank changes.

As we explain in Section II, their construction idea has a bottleneck of $r = O(\log N)$, and thus cannot prove a polynomial lower bound. Our lower bound functions are also partition submodular, and we also prove an $\Omega(r)$ lower bound though we get $r$ to be polynomially large in the size of the universe.

### A. Related Work

The rounds-of-adaptivity versus query complexity question has seen a lot of recent work on submodular function *maximization*. As mentioned before, Balkanski and Singer [7] introduced this problem in the context of maximizing a non-negative monotone submodular function $f(S)$ subject to a cardinality constraint $|S| \leq k$. This captures **NP**-hard problems, has a *sequential* greedy $(1-\frac{1}{e})$-approximation algorithm [41], and obtaining anything better requires [40], [46] exponentially many queries. [7] showed that obtaining even an $O\left(\frac{1}{\log N}\right)$-approximation with polynomially many queries requires $\Omega\left(\frac{\log N}{\log\log N}\right)$ rounds, and gave an $O(\log N)$-round, polynomial query, $\frac{1}{3}$-approximation. Soon afterwards, several different groups [5], [21], [23], [16], [15], [22] gave $\left(1-\frac{1}{e}-\varepsilon\right)$-approximation algorithms making polynomially many queries which run in $\mathrm{poly}(\log N, \frac{1}{\varepsilon})$-rounds, even when the constraint on which $S$ to pick is made more general. More recently, Li, Liu and Vondrák [38] showed that the dependence of the number of rounds on $\varepsilon$ (the distance from $1-1/e$) must be a polynomial. Also related is the question of maximizing a non-negative non-monotone submodular function without any constraints. It is known that a random set gives a $\frac{1}{4}$-approximation, and a sequential "double-greedy" $\frac{1}{2}$-approximation was given by Buchbinder, Feldman, Naor, and Schwartz [12], and this approximation factor is tight [24]. Chen, Feldman, and Karabasi [17] gave a nice parallel version obtaining an $\left(\frac{1}{2}-\varepsilon\right)$-approximation in $O(\frac{1}{\varepsilon})$-rounds.

In the continuous optimization setting, the question of understanding the "parallel complexity" of minimizing a non-smooth convex function was first studied by Nemirovski [42]. In particular, the paper studied the problem of $\varepsilon$-minimizing a bounded-norm convex (non-smooth) function over the unit $\ell_\infty$ ball in $N$-dimensions, and showed that any polynomial query (value oracle or gradient oracle) algorithm must have $\widetilde{\Omega}(N^{1/3}\ln(1/\varepsilon))$ rounds of adaptivity. Nemirovski [42] conjectured that the lower bound should be $\widetilde{\Omega}(N\ln(1/\varepsilon))$, and this is still an open question. When the dependence on $\varepsilon$ is allowed to be polynomial, then the sequential vanilla gradient descent outputs an $\varepsilon$-minimizer in $O(1/\varepsilon^2)$-rounds (over Euclidean unit norm balls), and the question becomes whether parallelism can help over gradient descent in some regimes of $\varepsilon$. Duchi, Bartlett, and Wainwright [20] showed an $O(N^{1/4}/\varepsilon)$-query algorithm which is better than gradient-descent when $\frac{1}{\varepsilon^2} \approx \sqrt{N}$. A matching lower bound in this regime $\frac{1}{\varepsilon^2} = \tilde{O}(\sqrt{N})$ was shown recently by Bubeck et al. [11]. It is perhaps worth noting that submodular function minimization can also be thought of as minimizing the Lovász extension which is a non-smooth convex function. Unfortunately, the domain of interest (the unit cube) has $\ell_2$-radius $\sqrt{N}$, and the above algorithms do not imply "dimension-free" $\varepsilon$-additive approximations for submodular function minimization. Currently the best known upper bound is the sequential $\tilde{O}(N/\varepsilon^2)$-query algorithm by Axelrod, Liu, and Sidford [4]. Our work shows that $\Omega(1/\varepsilon)$-rounds are needed, and it is an interesting open question whether a $\mathrm{poly}(N, \frac{1}{\varepsilon})$-lower bound can be shown on the number of rounds, or whether one can achieve efficient $\varepsilon$-approximations in rounds independent of $N$.

The question of rounds-of-adaptivity versus query complexity has been asked for many other computational models, and also is closely related to other fields such as communication complexity and streaming. We note a few results which are related to submodular function minimization. Assadi, Chen, and Khanna [3] considered the problem of finding the minimum $s$-$t$-cut in an undirected graph in the streaming setting. They showed that any $p$-pass algorithm must take $\widetilde{\Omega}(n^2/p^5)$-space, where $n$ is the number of vertices. Their result also implied that any sub-polynomial round algorithm for the $s$-$t$-cut submodular function must make $\widetilde{\Omega}(n^2)$ queries; note that with $O(n^2)$ queries, the whole graph can be non-adaptively learned. Rubinstein, Schramm, and Weinberg [44] considered the global minimum cut function in an undirected graph, and showed that $\tilde{O}(n)$ queries suffice, and their algorithm can be made to run in $O(1)$-rounds.

## II. Technical Overview

In this section, we give a technical overview of our approach to proving a polynomial lower bound on the rounds of adaptivity. We start by describing the Balkanski-Singer [8] framework for proving rounds-of-adaptivity lower bounds as it serves as a starting point for our work. Our presentation will first briefly highlight why the approach taken in [8] can not yield better than a logarithmic lower bound on the rounds of adaptivity and then describe the approach we take to sidestep the logarithmic bottleneck.

*The Lower Bound Framework.:* Balkanski and Singer [8] consider a class of submodular functions which we call ***partition submodular functions***. Given

a partition $P = (P_1, \ldots, P_r)$ of the universe $U$, a set function is partition submodular if its value at a subset $S$ depends only on the *cardinalities* of the number of elements it contains from each part. That is, $f_P(S) = h(|S \cap P_1|, \ldots, |S \cap P_r|)$ for some function $h$ whose domain is $r$-dimensional non-negative integer vectors. The lower bound framework dictates the following three conditions on the functions $h$ and the resulting partition submodular function $f_P$.

(P1) The function $h$ is defined such that $f_P$ is *submodular*.

(P2) The last part $P_r$ is the unique minimizer of $f_P$. We also assume $f_P(\emptyset) = h(0, 0, \ldots, 0) = 0$, and thus $f_P(P_r)$ is necessarily $< 0$.

(P3) For any $1 \leq i \leq r$, even if we know the identity of the parts $P_1, \ldots, P_{i-1}$, a single round of polynomially many queries tells us nothing about the identity of the parts $P_{i+1}$ to $P_r$. More precisely, a random re-partitioning of the elements in $P_{i+1} \cup P_{i+2} \cup \cdots \cup P_r$ will, with high probability, give the same values to the polynomially many queries made in the current round.

(P3) is the key property for proving the lower bound. Let $\mathcal{P}$ be the uniform distribution over partitions with given sizes $|P_1|$ to $|P_r|$ which induces a distribution over submodular functions. By Yao's lemma it suffices to show that any $(r-2)$-round deterministic algorithm making polynomially many queries fails to find the minimizer with any non-trivial probability. (P3) implies that after $(r-2)$ rounds of queries and obtaining their answers, the algorithm cannot distinguish between two functions $f_P$ and $f_{P'}$ where the partitions $P$ and $P'$ agree on the first $(r-2)$ parts, but $(P_{r-1}, P_r)$ and $(P'_{r-1}, P'_r)$ are random re-partitioning of the elements of $P_{r-1} \cup P_r$. Since (P2) implies the minimizer of $f_P$ is $P_r$ and $f_{P'}$ is $P'_r$, and these will be different with high probability, any algorithm will make a mistake on one of them. The non-triviality is therefore in the construction of the "$h$" functions, and in particular for how large an $r$ can one manage while maintaining (P1), (P2), and (P3).

*The Balkanski-Singer Construction:* For now, let us fix a *random* partition $P := (P_1, \ldots, P_r)$ of the universe $U$. Given a subset $S$, let $\mathbf{x} := (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_r)$, where $\mathbf{x}_i := |S \cap P_i|$ be its *signature*. Before we describe Balkanski and Singer's construction, let us understand what one needs for establishing a condition like (P3). Consider the case $i = 1$, that is, the first round of queries. (P3) requires that the answers should not leak any information about $P_2, P_3, \ldots, P_r$.

Consider a query $S$. Since the partition $P$ is random, we expect $S$'s signature $\mathbf{x}$ to be random as well. More precisely, we expect $\frac{\mathbf{x}_i}{|P_i|}$ to be "roughly same" for all $i \in [r]$. Call such vectors *balanced*; we are deliberately not defining them precisely at this point. For (P3) to hold, we ***must*** have that $\partial_i h(\mathbf{x})$, the marginal increase in the function upon adding an element from $P_i$, is the same for all $2 \leq i \leq r$ for ***balanced*** vectors. Otherwise, the algorithm can distinguish between different parts. On the other hand, the marginals cannot be same for *all* vectors $\mathbf{x}$, as that would imply the sets $P_2$ to $P_r$ have the same value, which would violate the constraint (P2) since $P_r$ is the unique minimizer.

To orchestrate this, Balkanski and Singer use the idea of masking. All marginals $\partial_i h(\mathbf{x})$ are between $[-1, 1]$. In the first round, the masking is done via the first coordinate $\frac{\mathbf{x}_1}{|P_1|}$ of the signature. At a very high level, when $\frac{\mathbf{x}_1}{|P_1|}$ is "large", all the marginals $\partial_i h(\mathbf{x})$, for $2 \leq i \leq r$, take the value $-1$, while $\partial_1 h(\mathbf{x})$ takes the value $0$. In plain English, if any set $S$ contains a large fraction of elements from $P_1$, then *all* elements in $P_2 \cup \cdots \cup P_r$ have marginal $-1$; the preponderance of these $P_1$ elements masks all the other parts outs, and only $P_1$ distinguishes itself in this round.

More generally one requires this kind of property to hold *recursively* as the algorithm discovers $P_1, P_2, \ldots$ in successive rounds. For any $i$, if one takes a set $S$ such that $\frac{|S \cap P_i|}{|P_i|}$ is "large" for some $i$, then for all elements $e$ in parts $P_j$, $j > i$, the marginals are $-1$. In this way, they are able to maintain the property (P3). Of course, one has to be careful about what occurs when $|S \cap P_i|$'s are small, and the whole construction is rather technical, but this aspect described above is key to how they maintain indistinguishability.

*A Logarithmic Bottleneck:* Unfortunately, this powerful masking property is also a bottleneck. One can argue that the above construction ***cannot*** have $r = \omega(\log N)$. To appreciate this, imagine for the moment that all marginals are $\{-1, +1\}$ (they are not for the Balkanski-Singer construction, but it helps explain the point). Consider the first round of queries. If $\frac{|S \cap P_1|}{|P_1|}$ is "large", say $\geq \theta$, then *all* $e \in P_2 \cup \cdots \cup P_r$ give a marginal of $-1$. How large can this $\theta$ be? If $\theta \gg \frac{1}{2}$, then a random set $R$ where every element is sampled with probability $1/2$, will be in a situation where the elements of $P_r$ are not giving a negative marginal with respect to $R$. But $|R \cap P_r| \approx |P_r|/2$, and by submodularity, this would mean if we consider $P_r$ alone, we do not get negative marginals till we see half the elements. However, that means $f(P_r)$ cannot be negative, violating (P2).

In sum, for any subset $S$ with $|S \cap P_1| \approx |P_1|/2$,

the marginal of *every* element in $P_2 \cup \cdots \cup P_r$ with respect to $S$ must be $-1$. In particular, this is true *even when* $S$ is just half the elements of $P_1$ alone. And this is problematic, as this implies : $f(S \cup P_2 \cup \cdots \cup P_r) = f(S) - \sum_{i=2}^{r} |P_i|$. But $S \cup P_2 \cup \cdots \cup P_r$ is not the minimizer. So $f(S)$ itself must be quite large. But $f(S) \leq |S| \approx |P_1|/2$, since the marginals are at most $+1$. In turn, this implies

$$|P_1| = \Omega(|P_2| + \cdots + |P_r|).$$

The first part is thus required to be much bigger than the sum of the rest. And recursively, the second part is bigger than the sum of the rest. And so on. This implies[3] $r = O(\log N)$ and the strong masking idea **cannot** give a polynomial lower bound.

### A. Ideas Behind Our Construction

Let us again focus on the first round of queries. In the Balkanski-Singer construction, whenever $\mathbf{x}_1$ is "large" ***irrespective*** of how the other $\mathbf{x}_i$'s look like, the marginals $\partial_i h(\mathbf{x}) = -1$ for $i \geq 2$. This strong masking property led to $|P_1|$ being much larger than the sum of the remaining parts so as to compensate for all the negative marginals coming from the elements in the other parts.

Our approach is not to set $\partial_i h(\mathbf{x})$ depending on just $\mathbf{x}_1$, but ***rather by looking at the whole suffix*** $\mathbf{x}_1 : \mathbf{x}_r$. More precisely, if $\mathbf{x}_1$ is "large" (say, even the whole part $P_1$), but all the rest are empty, even in that case we want *all* marginals $\partial_i h(\mathbf{x})$ to be in fact $+1$. Only when (almost) *all* coordinates $\mathbf{x}_i$ are "large", do we switch to $\partial_i h(x) = -1$ for all $i \geq 2$. Therefore, in some sense, for elements of any part to contribute negative to function value, we must have already picked up a significant number of elements from that part which contributed positively which cancel out the negative marginals. This allows our construction to have all parts of equal size $n = N/r$, setting the stage for a polynomial lower bound.

Although deciding a marginal depending on the suffix may sound complicated, in the end our lower bound functions are simple to describe. Indeed, all marginals are in the set $\{-1, 0, +1\}$ and thus not only do we prove a polynomial lower bound on exact SFM, we also prove a $\mathrm{poly}(1/\varepsilon)$-lower bound even for $\varepsilon$-approximate SFM. In the rest of this subsection, we give more details on how the marginals depend on the suffixes. This discussion is still kept informal and is meant to help the reader understand the rationale behind the construction. The full formal details along with all the properties we need are deferred to Section III, which the reader can feel free to skip to.

For our lower bound, we construct two partition submodular functions, $f_P(S) = h(\mathbf{x})$ and $f_P^* = h^*(\mathbf{x})$, where (a) the minimizer of $f_P$ is the empty set and the minimizer of $f_P^*$ is the set $P_r$ (satisfying (P2)), and *both* these functions satisfy (P3) for $1 \leq i < r/2$, and furthermore, no polynomial query randomized algorithm using only $o(N^{1/3}/\log^{1/3} N)$ rounds of adaptivity can distinguish between these two functions unless it makes a super-polynomial number of queries. It is easier to understand the functions $h$ and $h^*$ via their marginals. Here are the properties we desire from these marginal functions.

- (Submodularity.) Both function's marginals should be monotonically decreasing. Thus, once $\partial_j h(\mathbf{x})$ or $\partial_j h^*(\mathbf{x})$ becomes $-1$, they should stay $-1$ for all $\mathbf{y}$ "larger" than $\mathbf{x}$.
- (Unique Minima.) The part $P_r$ should be the unique minimizer for $h^*$. This will restrict how negative $\partial_j h^*$ can be for $j \neq r$. This is in tension with the previous requirement.
- (Suffix Indistinguishability.) For any $\mathbf{x}$ which is $i$-*balanced*, that is, $\mathbf{x}_i \approx \mathbf{x}_{i+1} \approx \cdots \approx \mathbf{x}_r$, we need that $\partial_j h^*(\mathbf{x})$ and $\partial_j h(\mathbf{x})$ for such $\mathbf{x}$'s should be the ***same*** for all $i + 1 \leq j \leq r$. This is what we call suffix indistinguishability. This would imply $h$ and $h^*$ would give the same values on all queried points with high probability.

At any point $\mathbf{x}$, let us first describe the $r$ marginals $\partial_i h(\mathbf{x})$ for $1 \leq i \leq r$. As mentioned above, the marginals will be in the set $\{-1, 0, +1\}$. It is best to think of this procedure constructively as an algorithm. Initially, all the $r$ marginals are set to $+1$. Next, we select up to two coordinates $a$ and $b$ in $\{1, 2, \ldots, r\}$, which depend on the query point $\mathbf{x}$. Given these coordinates, we decrement *all* marginals $a \leq i \leq r$ and *all* marginals $b \leq i \leq r$ by 1. For instance, if $r = 5$ and we choose the coordinates $a = 2$ and $b = 4$ at some $\mathbf{x}$, then the marginals $(\partial_1 h(\mathbf{x}), \ldots, \partial_5 h(\mathbf{x}))$ are $(1, 0, 0, -1, -1)$. The 4th and 5th coordinate decrement twice and thus go from $+1$ to $-1$, while the 2nd and 3rd coordinate only decrement once and thus go from $+1$ to 0. The first coordinate is never decremented in this example. Note that the vector of marginals when considered from 1 to $r$ is always in decreasing order.

The crux of the construction is, therefore, in the choice of the $a$ and the $b$ at a certain point $\mathbf{x}$. These

[3] It is not easy to even orchestrate a $\Omega(\log N)$ lower bound this way. The masking functions that Balkanski-Singer constructed needed to be quite delicate to preserve submodularity, and in the end, the sets $P_1$ is in fact $r$ times bigger than the rest. This leads to an $\Omega(\log N/\log\log N)$ lower bound.

will clearly depend on $\mathbf{x}$, but how? Submodularity tells us that if we move from $\mathbf{x}$ to $\mathbf{y} = \mathbf{x} + \mathbf{e}_i$, then the $a$'s and the $b$'s should only *move left*, that is, become smaller; that would ensure decreasing marginals. This in turn implies that $a$ and $b$ should be defined by the ***suffix sums*** at $\mathbf{x}$. More precisely, if we decide to choose $a$ and $b$ as the coordinates which *maximize* some function $\phi(\cdot)$ which depends on the suffix sums $\sum_{i \geq t} \mathbf{x}_i$, $t$ ranging from 1 to $r$, then increasing a coordinate can only move $a$'s and $b$'s to the left. This is precisely what we do, and now the crux shifts to the choice of this function $\phi(\cdot)$.

Consider an $i$-balanced vector $\mathbf{x}$. We need that when all the coordinates are "large", then the marginals of $h^*$ should be $-1$; otherwise, $P_r$ would not be the minimizer. Since $h$ and $h^*$ should be indistinguishable, the same should be true for $h$. On the other hand, when all the coordinates are "small", most marginals of both function should be $+1$, otherwise $U$ would be the minimizer. In sum, when the coordinates of $\mathbf{x}$ are "large", we should have the $a$ and $b$ to the left, close to 1; this would make most marginals $-1$. And when they are small, $a$ and $b$ should be towards the right; this would make most marginals $+1$. This motivates the following rule that we formalize in the next section : we define $r$ different functions (called $\ell_t(\mathbf{x})$ for $1 \leq t \leq r$) where the $t$th function $\ell_t(\mathbf{x})$ is the sum of $(\mathbf{x}_i - \tau)$ over all coordinates $t \leq i \leq r$ where $\tau$ is a "threshold" which is "close" to $n/2$. Here $n$ is the size of each part $|P_i|$. After taking the sum over these coordinates, we further subtract an"offset" $\gamma$. In sum, the functions look like $\ell_t(\mathbf{x}) := (\sum_{i=t}^{r}(\mathbf{x}_i - \tau)) - \gamma$.

We choose $a$ and $b$ to be the coordinates such that $\ell_t(\mathbf{x})$ is largest. Thus, if the maximizers are to the left (that is closer to 1), then it must be because *many* coordinates are over the threshold $\tau$. In that case, most marginals are $-1$. On the other hand, if most coordinates are under the threshold $\tau$, then $a$ and $b$'s are to the right (closer to $r$), and we have most marginals are $+1$. This threshold, therefore, ensures that for a coordinate to give a $-1$ marginal, it must also, in some sense, already have contributed lots of $+1$ as well. This is *precisely* how we overcome the geometrically decaying partition size problem in [8].

What should this threshold $\tau$ be? Ideally, we would like this threshold to be $n/2$; that way, for any $i$ it won't contribute to the summand of $\ell_t$ till $\mathbf{x}_i \geq n/2$. In essence, as $\mathbf{x}_i$ ranges from 0 to $n$, the $+1$ contributions to the marginal will cancel out the $-1$ contributions, and $P_i$ won't have a negative marginal. In reality, we need to provide a "gap" and set $\tau = n/2 - g$ where $g = \tilde{\Theta}(\sqrt{n})$. This is done because we want Suffix Indistinguishability to hold not only for perfectly $i$-balanced vectors but also for vectors whose values differ by "a few standard deviations". More precisely, a random subset should satisfy Suffix Indistinguishability with high probability, and choosing this threshold allows us to achieve this property.

Indeed the fact that this gap $g = \tilde{\Theta}(\sqrt{n})$ also is the reason why we cannot get better than $N^{1/3}$ lower bound. Indeed, if we take the set $S = U = P_1 \cup \cdots \cup P_r$, then the first part $P_1$ leads to a value of $\approx n/2$ since it never gives negative marginals and gives marginal $+1$ for the first $n/2 - g \approx n/2$ elements. The remaining parts, in essence, contribute $(n/2 - g) - (n/2 + g) = -2g$ to the function value. More precisely, one can prove that $f(U) = n - \Theta(gr)$. And thus, if we want $f(U) > 0$, we must have $n > \Theta(gr)$. Since $g \geq \sqrt{n}$, we get $r \leq \sqrt{n} = N^{1/3}$ because $N = nr$.

We end this informal discussion by describing the function $h^*$. It is simply the function $h$ if $\mathbf{x}_r < \frac{n}{2} - g/4$, but if $\mathbf{x}_r \geq \frac{n}{2} - g/4$, the $r$th coordinate has marginal $-1$ irrespective of the other $\mathbf{x}_j$'s. This makes $P_r$ become the minimizer of $f_P^*$ with value $-\Theta(g)$. Since we only modify the behavior of the last index in going from $h$ to $h^*$, in the beginning few rounds $h$ and $h^*$ behave similarly. Indeed, if $\mathbf{x}_r > \frac{n}{2} - g/4$, then any $i$-balanced vector for $i \leq r/2$, has half the coordinates $\geq \frac{n}{2} - O(g)$. The offset $\gamma$ is chosen such that in this case $h$ also has marginal $-1$ for the $r$th coordinate. Thus, $h$ and $h^*$ are indistinguishable in the first $r/2$ rounds. This, in turn, shows that if an algorithm runs for $< r/2$ rounds, then it cannot distinguish between these two functions, and therefore, cannot distinguish between the case when the minimum value is 0 and the minimum value is $-\Theta(g) = -\Omega(N^{1/3})$.

## III. Lower Bound Construction

We begin by formally defining partition submodular functions and some properties of such functions. We then describe in detail the lower bound functions that we use in the proof of Theorem 1.

### A. Partition Submodular Functions

Let $U$ be a universe of elements and $P = (P_1, \ldots, P_r)$ be a partition of the elements of $U$. Let $h : \mathbb{Z}_{\geq 0}^r \to \mathbb{R}$ be a function whose domain is the $r$-dimensional non-negative integer hypergrid. Given $(P, h)$, one can define a set-function $f_P : 2^U \to \mathbb{R}$ as follows:

$$f_P(S) = h\left(|S \cap P_1|, \ldots, |S \cap P_r|\right) \tag{1}$$

In plain English, the value of $f_P(S)$ is a function only of the *number* of elements of each part that is present

in $S$. We say that $f_P$ is induced by the partition $P$ and $h$. A ***partition submodular function*** is a submodular function which is induced by some partition $P$ and some hypergrid function $h$.

A function defined by $(P, h)$ is submodular if and only if $h$ satisfies the same decreasing marginal property as $f$. To make this precise, let us settle on some notation. Throughout the paper, for any integer $k$, we use $[k]$ to denote the set $\{0, 1, \ldots, k\}$. First, note that the domain of $h$ is the $r$-dimensional hypergrid $[|P_1|] \times [|P_2|] \times \cdots \times [|P_r|]$. For brevity's sake, we call this $\mathbf{dom}(h)$. We use boldfaced letters like $\mathbf{x}, \mathbf{y}$ to denote points in $\mathbf{dom}(h)$. When we write $\mathbf{x} + \mathbf{y}$ we imply coordinate-wise sum. Given $i \in \{1, \ldots, r\}$, we use $\mathbf{e}_i$ to denote the $r$-dimensional vector having 1 at the $i$th coordinate and 0 everywhere else. The function $h$ induces $r$ different ***marginal*** functions defined as

$$\text{For } 1 \leq i \leq r, \quad \partial_i h(\mathbf{x}) := h(\mathbf{x} + \mathbf{e}_i) - h(\mathbf{x}) \tag{2}$$

The domain of $\partial_i h$ is $[|P_1|] \times [|P_2|] \times \cdots \times [|P_i| - 1] \times \cdots \times [|P_r|]$.

**Definition 1.** *We call a function $h : \mathbb{Z}^r \to \mathbb{R}$ defined over a integer hypergrid $\mathbf{dom}(h)$ (hypergrid) submodular if and only if for every $1 \leq i \leq r$, for every $\mathbf{x} \in \mathbf{dom}(h)$ with $\mathbf{x}_i < |P_i|$, and every $1 \leq j \leq r$, we have*

$$\partial_j h(\mathbf{x}) \geq \partial_j h(\mathbf{x} + \mathbf{e}_i) \tag{3}$$

**Lemma 1.** *A set function $f_P$ defined by a partition $P$ and hypergrid function $h$ as in (1) is (partition) submodular if and only if $h$ is (hypergrid) submodular.*

The following lemma shows that minima of partition submodular functions can be assumed to take all or nothing of each part.

**Lemma 2.** *Let $f_P$ be a partition submodular function induced by a partition $P = (P_1, \ldots, P_r)$ and hypergrid function $h$. Let $O$ be a maximal by inclusion minimizer of $f$. Then, $O \cap P_i \neq \emptyset$ implies $O \cap P_i = P_i$.*

### *B. Description of Our Lower Bound Functions*

The lower bound functions we construct are partition submodular functions defined with respect to a partition $P$ of the universe $U$ of $N$ elements into $r$ parts, $P = (P_1, \ldots, P_r)$. The number of parts $r$ is an integer whose value will be set to be $\tilde{\Theta}(N^{1/3})$. Each part $P_i$ has the same size $n$, where $n$ is an even positive integer such that $nr = N$. To define these functions, we need to define the hypergrid submodular functions.

Let $g$ be an integer which is divisible by 4 and which is $\tilde{\Theta}(\sqrt{n})$. That is, $\left(\frac{n}{2} - g\right)$ is "many standard deviations" away from $\frac{n}{2}$, and in particular, any random subset of an $n$-universe set is within $\pm g$ of the expected value with all but inverse polynomial probability. As described in the previous informal discussion, the following linear functions play a key role in the description of the marginals. Define

$$\text{For any } 1 \leq t \leq r, \quad \ell_t(\mathbf{x}) := \sum_{s=t}^{r} \left(\mathbf{x}_s - \left(\frac{n}{2} - g\right)\right) - \frac{gr}{4} \tag{4}$$

Given $\mathbf{x}$, define a strict order $\succ_{\mathbf{x}}$ where

$$t \succ_{\mathbf{x}} s \quad \text{if} \quad \ell_t(\mathbf{x}) > \ell_s(\mathbf{x}) \quad \text{or} \quad \ell_t(\mathbf{x}) = \ell_s(\mathbf{x}) \text{ and } t < s \tag{5}$$

Given (4) and (5) at a point $\mathbf{x}$, let $\{a, b\}$ be the first two coordinates in the strict order $\succ_{\mathbf{x}}$. Note that we are not insisting $a$ is the first coordinate; we are considering $\{a, b\}$ as an unordered pair. Now we are ready to describe our lower bounding functions. First define the function $h : [n]^r \to \mathbb{Z}$ as follows

$$h(\mathbf{x}) = \|\mathbf{x}\|_1 - \Big(\max(0, \ell_a(\mathbf{x})) + \max(0, \ell_b(\mathbf{x}))\Big) \tag{6}$$

The above function contains the seed of the hardness, and satisfies (P1) and (P3). However, the above function is in fact non-negative. To obtain the lower bounding functions which treats $P_r$ specially, define

$$h^*(\mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{if } \mathbf{x}_r \leq \frac{n}{2} - \frac{g}{4} \\ h(\mathbf{x}_\downarrow) - \left(\mathbf{x}_r - \left(\frac{n}{2} - \frac{g}{4}\right)\right) & \text{otherwise} \end{cases}$$
$$\text{where, } \mathbf{x}_\downarrow := \left(\mathbf{x}_1, \ldots, \mathbf{x}_{r-1}, \min(\mathbf{x}_r, \frac{n}{2} - \frac{g}{4})\right) \tag{7}$$

In Section III-D, we prove that both functions, $h$ and $h^*$ are hypergrid submodular. In Section III-E, we prove that $P_r$ is the unique minimizer of the function $f_P^*$ defined over any partition $P$ with the $h^*$ function, and that $\emptyset$ is the unique minimizer of the function $f_P$ defined using $h$. In Section III-F, we prove that any polynomial query algorithm making $< r/2$ rounds-of-adaptivity cannot find the part $P_r$; indeed, no such algorithm can distinguish between a partition submodular function generated by $h$ and one generated by $h^*$.

### *C. Marginals*

To prove submodularity, it is easier if we establish the marginals of the function defined in (6). Indeed, this may even help in understanding these functions. To this end, define the following indicator functions. For

any $1 \le t \le n$ and for any $1 \le i \le n$, define

$$\mathsf{C}_t(\mathbf{x}) = \begin{cases} -1 & \text{if } \ell_t(\mathbf{x}) \ge 0 \\ 0 & \text{otherwise} \end{cases}$$

and

$$\mathsf{C}^i_t(\mathbf{x}) = \mathsf{C}_t(\mathbf{x}) \cdot \mathbf{1}_{\{i \ge t\}}$$

where $\mathbf{1}_{\{i \ge t\}}$ is the indicator function taking the value 1 if $i \ge t$ and 0 otherwise. Using this notation, we can describe the $r$ different marginals at $\mathbf{x}$ succinctly as given by the lemma below.

**Lemma 3.** *Fix $\mathbf{x}$ in the domain of $h$. Let $\{a, b\}$ be the first two elements of $\succ_{\mathbf{x}}$ at $\mathbf{x}$. Then,*

$$\forall 1 \le i \le r, \;\; \partial_i h(\mathbf{x}) = 1 + \mathsf{C}^i_a(\mathbf{x}) + \mathsf{C}^i_b(\mathbf{x}) \quad \text{(Marginals)}$$

It is helpful to interpret these marginals in plain English. Given a point $\mathbf{x}$, one first finds the first two coordinates $\{a, b\}$ in $\succ_{\mathbf{x}}$. That is, $\ell_a(\mathbf{x})$ and $\ell_b(\mathbf{x})$ are the largest among all coordinates, with ties broken towards smaller indices. If any of these function values are negative, throw them away from consideration: the suffixes aren't large enough. Next, given a coordinate $i$, the marginal $\partial_i h(\mathbf{x})$ depends on where $i$ lies in respect to $a$ and $b$ (if they are still in consideration). If $i$ is smaller than both, then the marginal is 1, if $i$ is smaller than one, then the marginal is 0, if $i$ is greater than or equal to both, the marginal is $-1$. This establishes what we wanted in the informal description.

The proof of the lemma uses the following easy structural claim which captures how these $\{a, b\}$s change when one increments a coordinate. In particular, it states that these can only "move left". This claim will also be used to establish submodularity of (6).

**Claim 1.** *Let $\mathbf{x}$ be any point and let $\mathbf{y} := \mathbf{x} + \mathbf{e}_i$. Suppose $\{a, b\}$ are two coordinates such that $a \succ_{\mathbf{x}} b$ but $b \succ_{\mathbf{y}} a$. Then, (i) $b \le i < a$, and (ii) $\ell_b(\mathbf{y}) = \ell_a(\mathbf{y})$.*

### D. Submodularity

We first prove that $h : [n]^r \to \mathbb{Z}$ is submodular, and then use this to prove that $h^* : [n]^r \to \mathbb{Z}$ is submodular. The high-level reason why $h$ is submodular is when one moves from $\mathbf{x}$ to $\mathbf{y} = \mathbf{x} + \mathbf{e}_i$, the first two elements $\{a, b\}$ of $\succ$ can only "move to the left", that is, become smaller. And thus, if a coordinate $j$ was larger than $\{a, b\}$, it remains larger when one moves to $\mathbf{y}$.

**Lemma 4.** *Fix $\mathbf{x}$ and a coordinate $1 \le i \le r$. Let $\mathbf{y} := \mathbf{x} + \mathbf{e}_i$. Let $j$ be any arbitrary coordinate. Then,*

$$\partial_j h(\mathbf{x}) \ge \partial_j h(\mathbf{y}) \tag{8}$$

*Proof:* Let $\{a_1, b_1\}$ be the first two elements of $\succ_{\mathbf{x}}$. Let $\{a_2, b_2\}$ be the first two elements of $\succ_{\mathbf{y}}$. From the definition of the marginals, what we need to show is

$$\mathsf{C}^j_{a_1}(\mathbf{x}) + \mathsf{C}^j_{b_1}(\mathbf{x}) \ge \mathsf{C}^j_{a_2}(\mathbf{y}) + \mathsf{C}^j_{b_2}(\mathbf{y}) \tag{9}$$

For any $1 \le t \le r$, observe that $\ell_t(\mathbf{y}) \ge \ell_t(\mathbf{x})$, and thus $\mathsf{C}_t(\mathbf{x}) \ge \mathsf{C}_t(\mathbf{y})$.

*Case 1::* $\{a_1, b_1\} = \{a_2, b_2\}$. In this case (9) follows directly from the above observation.

*Case 2::* $|\{a_1, b_1\} \cap \{a_2, b_2\}| = 1$. Without loss of generality, suppose $a_1 = a_2$ and $b_1 \ne b_2$. First, we get $\mathsf{C}^j_{a_1}(\mathbf{x}) = \mathsf{C}^j_{a_2}(\mathbf{x}) \underbrace{\ge}_{\text{above observation}} \mathsf{C}^j_{a_2}(\mathbf{y})$. Second, since $b_1 \succ_{\mathbf{x}} b_2$ and $b_2 \succ_{\mathbf{y}} b_1$, Claim 1(i) gives us $b_2 \le i < b_1$. Also, $\ell_{b_2}(\mathbf{y}) \underbrace{\ge}_{b_2 \succ_{\mathbf{y}} b_1} \ell_{b_1}(\mathbf{y}) \underbrace{\ge}_{\text{above observation}} \ell_{b_1}(\mathbf{x})$. This implies $\mathsf{C}_{b_1}(\mathbf{x}) \ge \mathsf{C}_{b_2}(\mathbf{y})$. Since $b_2 < b_1$, we get that $\mathbf{1}_{\{j \ge b_2\}} \ge \mathbf{1}_{\{j \ge b_1\}}$. Since $\mathsf{C}$ is non-positive, we get $\mathsf{C}^j_{b_1}(\mathbf{x}) = \mathbf{1}_{\{j \ge b_1\}} \cdot \mathsf{C}_{b_1}(\mathbf{x}) \ge \mathbf{1}_{\{j \ge b_2\}} \cdot \mathsf{C}_{b_2}(\mathbf{y}) = \mathsf{C}^j_{b_2}(\mathbf{y})$.

*Case 3::* $|\{a_1, b_1\} \cap \{a_2, b_2\}| = 0$. This is analogous to Case 2. The proof of $\mathsf{C}^j_{b_1}(\mathbf{x}) \ge \mathsf{C}^j_{b_2}(\mathbf{y})$ is exactly the same, and the proof of $\mathsf{C}^j_{a_1}(\mathbf{x}) \ge \mathsf{C}^j_{a_2}(\mathbf{y})$ is analogous with "$b$"s replaced by "$a$"s. ∎

**Lemma 5.** *The function $h^*$ as defined in (7) is submodular*

### E. Unique Minima

Given a partition $P = (P_1, \ldots, P_r)$, let $f^*_P$ be the partition submodular function defined as $f^*_P(S) = h^*(\mathbf{x})$ where $\mathbf{x}_i = |S \cap P_i|$. Similarly, define $f_P(S) := h(\mathbf{x})$. Then,

**Lemma 6.** *Suppose the parameters $n, g$ and $r$ chosen such that $5gr \le n$. Then the following are true. $\emptyset$ is the unique minimizer of $f_P$ achieving the value 0. $P_r$ is the unique minimizer of $f^*_P$ achieving the value $-\frac{g}{2}$.*

### F. Suffix Indistinguishability

We now establish the key property about $h$ and $h^*$ which allows us to prove a polynomial lower bound on the rounds of adaptivity. To do so, we need a definition.

**Definition 2.** *For $1 \le i < r$, a point $\mathbf{x} \in [n]^r$ is called $i$-**balanced** if $\mathbf{x}_i - \frac{g}{8} \le \mathbf{x}_j \le \mathbf{x}_i + \frac{g}{8}$ for all $j > i$.*

Suffix Indistinguishability asserts that two points $\mathbf{x}$ and $\mathbf{x}'$ which are $i$-balanced, have the same norm, and which agree on the first $i$ coordinates have the same function value. We first prove Suffix Indistinguishability for $h$, and then show that if $i < \frac{r}{2}$, then $h$ and $h^*$ take the same value on $i$-balanced points, which implies Suffix Indistinguishability for $h^*$ as well (for $i < \frac{r}{2}$).

**Claim 2.** *Let $i \leq r-2$. If $\mathbf{x}$ and $\mathbf{x}'$ are two $i$-balanced points with $\mathbf{x}_j = \mathbf{x}'_j$ for $j \leq i$ and $\|\mathbf{x}\|_1 = \|\mathbf{x}'\|_1$, then $h(\mathbf{x}) = h(\mathbf{x}')$.*

*Proof:* First note that for any $t \in \{1, 2, \ldots, i+1\}$, $\ell_t(\mathbf{x}) = \ell_t(\mathbf{x}')$; this follows from the fact that $\|\mathbf{x}\|_1 = \|\mathbf{x}'\|_1$ and that $\mathbf{x}$ and $\mathbf{x}'$ agree on the first $i$-coordinates. Let $\{a, b\}$ be the first two elements in $\succ_{\mathbf{x}}$ and $\{a', b'\}$ be the first two elements in $\succ_{\mathbf{x}'}$. Note that if all four of these are known to be in $\{1, 2, \ldots, i+1\}$, then we must have $\{a, b\} = \{a', b'\}$. This, in turn, implies $h(\mathbf{x}) = h(\mathbf{x}')$ since $\|\mathbf{x}\|_1 = \|\mathbf{x}'\|_1$.

Case 1: $\mathbf{x}_i = \mathbf{x}'_i < \frac{n}{2} - \frac{7g}{8}$. Since $\mathbf{x}$ and $\mathbf{x}'$ are both $i$-balanced, we have $\mathbf{x}_j, \mathbf{x}'_j < \frac{n}{2} - \frac{7g}{8} + \frac{g}{8} = \frac{n}{2} - \frac{3g}{4}$ for all $j \geq i$. This, in turn, implies that for any $t \geq i$, $\ell_t(\mathbf{x}), \ell_t(\mathbf{x}')$ are both $\leq \frac{gr}{4} - \frac{gr}{4} = 0$, since each summand in the definition (4) contributes at most $\frac{g}{4}$. In turn, this means that either $h(\mathbf{x}) = \|\mathbf{x}\|_1$ and $h(\mathbf{x}') = \|\mathbf{x}'\|_1$ and thus they are equal since the norms are equal, or $\{a, b\}$ and $\{a', b'\}$ lie in $\{1, 2, \ldots, i+1\}$ and we are done by the discussion in the first paragraph.

Case 2: $\mathbf{x}_i = \mathbf{x}'_i \geq \frac{n}{2} - \frac{7g}{8}$. Since $\mathbf{x}$ and $\mathbf{x}'$ are both $i$-balanced, we have $\mathbf{x}_j, \mathbf{x}'_j \geq \frac{n}{2} - g$ for all $j \geq i$. Thus each term in the summands of (4) is $\geq 0$. This, in turn implies that the maximizers of $\ell_t(\mathbf{x}), \ell_t(\mathbf{x}')$, both lie in $\{1, 2, \ldots, i+1\}$. From the argument in the first paragraph, we get $h(\mathbf{x}) = h(\mathbf{x}')$. ∎

Next, we prove that when $i$ is bounded away from $r$, for any $i$-balanced vector $\mathbf{x}$, we have $h^*(\mathbf{x}) = h(\mathbf{x})$. This lemma is useful to prove the indistinguishability of $h^*$ and $h$.

**Lemma 7.** *If $i < \frac{r}{2}$ and $\mathbf{x}$ is $i$-balanced, then $h^*(\mathbf{x}) = h(\mathbf{x})$.*

*Proof:* If $\mathbf{x}_r \leq \frac{n}{2} - \frac{g}{4}$, we have $h^*(\mathbf{x}) = h(\mathbf{x})$ by definition. So we only need to consider the case when $\mathbf{x}_r \geq \frac{n}{2} - \frac{g}{4}$. Let $k := \mathbf{x}_r - \left(\frac{n}{2} - \frac{g}{4}\right)$, by definition $\|\mathbf{x}\|_1 = \|\mathbf{x}_\downarrow\|_1 + k$ and $h^*(\mathbf{x}) = h(\mathbf{x}_\downarrow) - k$. For any $1 \leq t \leq r$, we have $\ell_t(\mathbf{x}) = \ell_t(\mathbf{x}_\downarrow) + k$, which means total orders $\succ_{\mathbf{x}}$ and $\succ_{\mathbf{x}_\downarrow}$ are the same. Let $\{a, b\}$ be the first two coordinates in both strict orders.

Since $\mathbf{x}$ is $i$-balanced and $\mathbf{x}_r \geq \frac{n}{2} - \frac{g}{4}$, we have $\mathbf{x}_i \geq \frac{n}{2} - \frac{3g}{8}$, and thus, for any $j \geq i$, $\mathbf{x}_j \geq \frac{n}{2} - \frac{g}{2}$. Thus, all summands in (4) for $j \geq i$ give non-negative contribution. This means both $a$ and $b$ lie in $\{1, 2, \ldots, i+1\}$. On the other hand, both $\ell_i(\mathbf{x}_\downarrow)$ and $\ell_{i+1}(\mathbf{x}_\downarrow)$ are at least $(r-i-1)\frac{g}{2} - \frac{gr}{4} \geq 0$ since $i \leq \frac{r}{2} - 1$. So both $\ell_a(\mathbf{x}_\downarrow)$ and $\ell_b(\mathbf{x}_\downarrow)$ are at least 0, which implies that both $\ell_a(\mathbf{x})$ and $\ell_b(\mathbf{x})$ are at least 0. Therefore, we have

$$\begin{aligned} h^*(\mathbf{x}) = h(\mathbf{x}_\downarrow) - k &= \left(\|\mathbf{x}_\downarrow\|_1 - \ell_a(\mathbf{x}_\downarrow) - \ell_b(\mathbf{x}_\downarrow)\right) - k \\ &= \|\mathbf{x}\|_1 - \ell_a(\mathbf{x}) - \ell_b(\mathbf{x}) = h(\mathbf{x}). \end{aligned}$$

∎

Claim 2 and Lemma 7 implies the following Suffix Indistinguishability property of $h^*$ and $h$.

**Lemma 8.** *Let $i < \frac{r}{2}$. If $\mathbf{x}$ and $\mathbf{x}'$ are two $i$-balanced points with $\mathbf{x}_j = \mathbf{x}'_j$ for $j \leq i$ and $\|\mathbf{x}\|_1 = \|\mathbf{x}'\|_1$, then $h^*(\mathbf{x}) = h^*(\mathbf{x}') = h(\mathbf{x}) = h(\mathbf{x}')$.*

## IV. Proof of the Main Theorem

We now prove lower bounds on the rounds-of-adaptivity for algorithms which make $\leq N^c$ queries per round for some $1 \leq c \leq N^{1-\delta}$ where $\delta > 0$ is a constant. Let $n$ be an even integer and $g$ be an integer divisible by 4 such that $800\sqrt{cn \log n} \geq g \geq 200\sqrt{cn \log n}$. Let $r$ be the largest integer such that $gr \leq n$. Finally, let $N = nr$. Note that $g = \Theta(N^{1/3}(c \log N)^{2/3})$, $r = \Theta\left(\frac{N^{1/3}}{(c \log N)^{1/3}}\right)$, and $n = \Theta(N^{2/3}(c \log N)^{1/3})$. Since $c \leq N^{1-\delta}$, we get $n > cN^{2\delta/3} > c \log N$ and thus $g \geq 200c \log n$.

Let $P = (P_1, \ldots, P_r)$ be a random equipartition of a universe $U$ with $N$ elements into parts of size $n$. Given a subset $S$, let the $r$-dimensional vector $\mathbf{x}$ defined as $\mathbf{x}_i := |S \cap P_i|$ be the signature of $S$ with respect to $P$. We say a query $S$ is *$i$-balanced* with respect to $P$ if the associated signature $\mathbf{x}$ is $i$-balanced. We use the following simple property of a random equipartition.

**Lemma 9.** *For any integer $i \in [1, \ldots, (r-1)]$, let $P_1, P_2, ..., P_{i-1}$ be a sequence of $(i-1)$ sets each of size $n$ such that for $1 \leq j \leq (i-1)$, the set $P_j$ is generated by choosing uniformly at random $n$ elements from $U \setminus (P_1 \cup P_2 \cup ...P_{j-1})$. Let $S \subset U$ be any query that is chosen with possibly complete knowledge of $P_1, P_2, ..., P_{i-1}$. Then if we extend $P_1, P_2, ..., P_{i-1}$ to a uniformly at random equipartition $(P_1, ..., P_r)$ of $U$, with probability at least $1 - 1/n^{2c+3}$, the query $S$ is $i$-balanced with respect to the partition $(P_1, P_2, ..., P_r)$; here the probability is taken over the choice of $P_i, P_{i+1}, ..., P_r$.*

To prove Theorem 1, we use Yao's minimax lemma. The distribution over hard functions is as follows. First, we sample a random equipartition $P$ of the $U$ into $r$ parts each of size $n$. Given $P$ and a subset $S$, let $f_P(S) := h(\mathbf{x})$ and $f^*_P(S) := h^*(\mathbf{x})$, where $\mathbf{x}$ is the signature of $S$ with respect to $P$. Select one of $f_P$ and $f^*_P$ uniformly at random. This fixes the distribution over the functions, and this distribution is offered to a deterministic algorithm. We now prove that any $s$-round deterministic algorithm with $s < \frac{r}{2}$ fails to return the correct answer with probability $> 1/3$, and this would prove Theorem 1. In fact, we prove that with probability $\geq 1 - 1/n$, over the random equipartition $P$,

the deterministic algorithm cannot distinguish between $f_P$ and $f_P^*$, that is, the answers to all the queries made by the algorithm are the same on both functions. This means that the deterministic algorithm errs with probability $\geq \frac{1}{2} \cdot (1 - \frac{1}{n}) > \frac{1}{3}$. Indeed, since the minimum values of $f_P$ and $f_P^*$ are 0 and $-\frac{g}{2}$ for all $P$'s (by Lemma 6), and since $g = \Theta(N^{1/3}(c \log N)^{2/3})$, we also rule out an *additive* $O(N^{1/3})$-approximation. Since the range of $f_P$ and $f_P^*$ are integers in $[-N, +N]$, we also get an $\widetilde{\Omega}(\frac{1}{\varepsilon^{0.5}})$ lower bound on the rounds-of-adaptivity required to get an $\varepsilon$-additive approximation to SFM of functions in range $[-1, +1]$.

An $s$-round deterministic algorithm performs a collection of queries $\mathsf{Q}^{(\ell)}$ at every round $1 \leq \ell \leq s$ with $|\mathsf{Q}^{(\ell)}| \leq N^c \leq n^{2c}$. Let $\mathsf{Ans}^{(\ell)}$ denote the answers to the queries in $\mathsf{Q}^{(\ell)}$. The subsets queried in $\mathsf{Q}^{(\ell)}$ is a deterministic function of the answers given in $\mathsf{Ans}^{(1)}, \ldots, \mathsf{Ans}^{(\ell-1)}$. After receiving the answers to the $s$th round of queries, that is $\mathsf{Ans}^{(s)}$, the algorithm must return the minimizing set $S$. We now prove that when $P$ is a random equipartition of $U$, then with probability $1 - \frac{1}{n}$, the answers $\mathsf{Ans}^{(\ell)}$ given to $\mathsf{Q}^{(\ell)}$ are the same for $f_P$ and $f_P^*$, if $s < \frac{r}{2}$.

We view the process of generating the random equipartition as a game between an adversary and the algorithm where the adversary reveals the parts one-by-one. Specifically, the process of generating the random equipartition will be such that at the start of any round $\ell \in [1, \ldots, s]$, the adversary has only chosen and revealed to the algorithm the parts $P_1, P_2, ..., P_{\ell-1}$, and at this stage, $P_\ell, P_{\ell+1}, ..., P_r$ are equally likely to be any equipartition of $U \setminus (P_1 \cup P_2 \cup ... \cup P_{\ell-1})$ into $(r-\ell+1)$ parts. By the end of round $\ell$, the adversary has committed and revealed to the algorithm the part $P_\ell$, and the game continues with one caveat. In each round, there will be a small probability (at most $1/n^2$) with which the adversary may "fail". This occurs at a round $\ell$ if any query made by the algorithm *on or before round* $\ell$ turns out to be not $\ell$-balanced with respect to the sampled partition at round $\ell$. In that case, the adversary reveals all remaining parts to the algorithm (consistent with the answers given thus far), and the game terminates in the current round $\ell$ itself with the algorithm winning the game (that is, the algorithm can distinguish between $f_P$ and $f_P^*$). The probability of this failure event can be bound by $s/n^2 \leq 1/n$, summed over all rounds. In absence of this failure event, by Lemma 7, we know that the answers will be the same for $f_P$ and $f_P^*$ at the end of the algorithm, concluding the proof. We now formally describe this process.

At the start of round 1, the adversary samples a uniformly at random equipartition of $U$, say, $\Gamma^{(1)} = (P_1^{(1)}, P_2^{(1)}, ..., P_r^{(1)})$. The algorithm reveals its set of queries for round 1, namely, $\mathsf{Q}^{(1)}$. The adversary answers all queries in $\mathsf{Q}^{(1)}$ in accordance with the partition $\Gamma^{(1)}$. By Lemma 9, since $|\mathsf{Q}^{(1)}| \leq n^{2c}$, every query in $\mathsf{Q}^{(1)}$ is 1-balanced with respect to the partition $\Gamma^{(1)}$, with probability at least $1 - 1/n^3$. If this event occurs, the adversary reveals $P_1^{(1)}$ to the algorithm, and continues to the next round. Otherwise, the adversary reveals the entire partition $\Gamma^{(1)}$ to the algorithm and the game terminates.

At the start of round 2, the adversary samples another uniformly at random equipartition of $U$, say, $\Gamma^{(2)} = (P_1^{(2)}, P_2^{(2)}, ..., P_r^{(2)})$ subject to the constraint $P_1^{(2)} = P_1^{(1)}$. Note that $\Gamma^{(2)}$ is a uniformly at random equipartition of $U$ since $P_1^{(1)}$ was chosen uniformly at random. The algorithm reveals its set of queries for round 2, namely, $\mathsf{Q}^{(2)}$. Again by Lemma 9, we have that (i) every query in $\mathsf{Q}^{(1)}$ is 1-balanced with respect to the partition $\Gamma^{(2)}$, with probability at least $1 - 1/n^3$, and (ii) every query in $\mathsf{Q}^{(2)}$ is 2-balanced with respect to the partition $\Gamma^{(2)}$, with probability at least $1 - 1/n^3$. If this event occurs, the adversary answers all queries in $\mathsf{Q}^{(2)}$ in accordance with the partition $\Gamma^{(2)}$, and the game proceeds to the next round. The key insight here is that by Lemma 8, if a query $S \in \mathsf{Q}^{(i)}$ is $i$-balanced w.r.t. some partition $(P_1, ..., P_r)$, then the function value on the query $S$ is completely determined by $P_1, P_2, ..., P_i$ and $|S|$, and does not require knowledge of $P_{i+1}, ..., P_r$. Furthermore, the value of $f_P(S)$ *and* $f_P^*(S)$ are the same. In other words, the function value on query $S$ remains unchanged, for both $f$ and $f^*$, if we replace $P := (P_1, ..., P_i, P_{i+1}, ..., P_r)$ with any other partition $P' := (P_1, ..., P_i, P'_{i+1}, .., P'_r)$ such that $S$ remains $i$-balanced with respect to $P'$. So answers to all queries in $\mathsf{Q}^{(1)}$ are the same under both partitions $\Gamma^{(1)}$ and $\Gamma^{(2)}$. On the other hand, if either (i) or (ii) above does not occur, the adversary terminates the game and reveals the entire partition $\Gamma^{(1)}$ to the algorithm.

In general, if the game has successfully reached round $\ell \leq s$, then at the start of round $\ell$, the adversary samples a uniformly at random equipartition of $U$, say, $\Gamma^{(\ell)} = (P_1^{(\ell)}, P_2^{(\ell)}, ..., P_r^{(\ell)})$ subject to the constraints $P_1^{(\ell)} = P_1^{(1)}, P_2^{(\ell)} = P_2^{(2)}, ..., P_{\ell-1}^{(\ell)} = P_{\ell-1}^{(\ell-1)}$. Once again, note that $\Gamma^{(\ell)}$ is a uniformly at random equipartition of $U$ since $P_1^{(1)}$ was chosen uniformly at random, $P_2^{(2)}$ was chosen uniformly at random having fixed $P_1^{(1)}$, and so on. The algorithm now reveals its set of queries for round $\ell$, namely, $\mathsf{Q}^{(\ell)}$. By Lemma 9, we have that for any fixed $i \in [1, \ldots, \ell]$, all queries in $\mathsf{Q}^{(i)}$ are $i$-balanced with respect to the partition $\Gamma^{(\ell)}$

with probability at least $1 - 1/n^3$ each. Thus with probability at least $1 - \ell/n^3$, for every $i \in [1, \ldots, \ell]$, all queries in $\mathsf{Q}^{(i)}$ are $i$-balanced with respect to the partition $\Gamma^{(\ell)}$. If this event occurs, the adversary answers all queries in $\mathsf{Q}^{(\ell)}$ with respect to the partition $\Gamma^{(\ell)}$, and once again, by Lemma 8, answers to all queries in $\mathsf{Q}^{(1)}, \mathsf{Q}^{(2)}, ..., \mathsf{Q}^{(\ell-1)}$ remain unchanged if we answer them using the partition $\Gamma^{(\ell)}$. The game then continues to the next round. Otherwise, with probability at most $\ell/n^3 \leq 1/n^2$, the game terminates and the adversary reveals the entire partition $\Gamma^{(\ell-1)}$ to the algorithm.

Summing up over all rounds 1 through $s \leq \frac{r}{2} - 1$, the probability that the game reaches round $s$ is at least $1 - s/n^2 \geq 1 - 1/n$. This, in turn, implies that with probability $\geq 1 - \frac{1}{n}$, the random equipartition $P$ satisfies the following property : all the queries in $\mathsf{Q}^{(i)}$ are $i$-balanced with respect to $P$ for all $i \in [1..s]$. Now, since $s \leq \frac{r}{2}$, by Lemma 7 we get that the answers $\mathsf{Ans}^{(1)}, \ldots, \mathsf{Ans}^{(s)}$ given to these queries are the same for $f_P$ and $f_P^*$. Hence the algorithm cannot distinguish between these two cases. This completes the proof of Theorem 1.

## Acknowledgements

This work was supported in part by NSF awards CCF-1910534, CCF-1926872, CCF-2041920, and CCF-2045128.

## References

[1] https://www.cis.upenn.edu/~chenyu2/papers/parallel-sfm-lowerbound.pdf.

[2] https://arxiv.org/pdf/2111.07474.pdf.

[3] S. Assadi, Y. Chen, and S. Khanna. Polynomial pass lower bounds for graph streaming algorithms. In *Proc., ACM Symposium on Theory of Computing (STOC)*, pages 265–276, 2019.

[4] B. Axelrod, Y. P. Liu, and A. Sidford. Near-optimal approximate discrete and continuous submodular function minimization. In *Proc., ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 837–853, 2020.

[5] E. Balkanski, A. Rubinstein, and Y. Singer. An exponential speedup in parallel running time for submodular maximization without loss in approximation. *Proc., ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 283–302, 2019.

[6] E. Balkanski and Y. Singer. Minimizing a submodular function from samples. In *Adv. in Neu. Inf. Proc. Sys. (NeurIPS)*, pages 814–822, 2017.

[7] E. Balkanski and Y. Singer. The adaptive complexity of maximizing a submodular function. In *Proc., ACM Symposium on Theory of Computing (STOC)*, pages 1138–1151, 2018.

[8] E. Balkanski and Y. Singer. A lower bound for parallel submodular minimization. In *Proc., ACM Symposium on Theory of Computing (STOC)*, pages 130–139, 2020.

[9] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(9):1124 – 1137, 2004.

[10] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization viagraph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(11):1222 – 1239, 2001.

[11] S. Bubeck, Q. Jiang, Y.-T. Lee, Y. Li, and A. Sidford. Complexity of highly parallel non-smooth convex optimization. In *Adv. in Neu. Inf. Proc. Sys. (NeurIPS)*, pages 13900–13909, 2019.

[12] N. Buchbinder, M. Feldman, J. S. Naor, and R. Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM Journal on Computing (SICOMP)*, 44(5):1384–1402, 2015.

[13] D. Chakrabarty, P. Jain, and P. Kothari. Provable submodular minimization using Wolfe's algorithm. In *Adv. in Neu. Inf. Proc. Sys. (NeurIPS)*, pages 802–809, 2014.

[14] D. Chakrabarty, Y. T. Lee, A. Sidford, and S. C. Wong. Subquadratic submodular function minimization. In *Proc., ACM Symposium on Theory of Computing (STOC)*, pages 1220–1231, 2017.

[15] C. Chekuri and K. Quanrud. Parallelizing greedy for submodular set function maximization in matroids and beyond. In *Proc., ACM Symposium on Theory of Computing (STOC)*, pages 78–89, 2019.

[16] C. Chekuri and K. Quanrud. Submodular function maximization in parallel via the multilinear relaxation. In *Proc., ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 303–322, 2019.

[17] L. Chen, M. Feldman, and A. Karbasi. Unconstrained submodular maximization with constant adaptive complexity. In *Proc., ACM Symposium on Theory of Computing (STOC)*, pages 102–113, 2019.

[18] W. Cunningham. On submodular function minimization. *Combinatorica*, 5:185 – 192, 1985.

[19] D. Dadush, L. A. Végh, and G. Zambelli. Geometric rescaling algorithms for submodular function minimization. In *Proc., ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 832–848, 2018.

[20] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

[21] A. Ene and H. L. Nguyen. Submodular maximization with nearly-optimal approximation and adaptivity in nearly-linear time. *Proc., ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 274–282, 2019.

[22] A. Ene, H. L. Nguyen, and A. Vladu. Submodular maximization with matroid and packing constraints in parallel. In *Proc., ACM Symposium on Theory of Computing (STOC)*, pages 90–101, 2019.

[23] M. Fahrbach, V. Mirrokni, and M. Zadimoghaddam. Submodular maximization with nearly optimal approximation, adaptivity and query complexity. In *Proc., ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 255–273, 2019.

[24] U. Feige, V. Mirrokni, and J. Vondrak. Maximizing non-monotone submodular functions. *SIAM Journal on Computing (SICOMP)*, 40(4):1133 – 1153, 2011.

[25] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[26] R. Gurjar and R. Rathi. Linearly representable submodular functions: An algebraic algorithm for minimization. In *Proc., International Colloquium on Automata, Languages and Programming (ICALP)*, pages 61:1–61:15, 2020.

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

[28] S. Iwata and J. B. Orlin. A simple combinatorial algorithm for submodular function minimization. In *Proc., ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1230–1237, 2009.

[29] R. Iyer and J. A. Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. *Adv. in Neu. Inf. Proc. Sys. (NeurIPS)*, 2013.

[30] R. Iyer, S. Jegelka, and J. A. Bilmes. Fast semidifferential-based submodular function optimization. In *ICML (3)*, pages 855–863, 2013.

[31] H. Jiang. Minimizing convex functions with integral minimizers. In *Proc., ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 976–985, 2021.

[32] D. R. Karger and R. Motwani. An NC algorithm for minimum cuts. *SIAM J. Comput.*, 26(1):255–272, 1997.

[33] R. M. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in random NC. *Combinatorica*, 6(1):35–48, 1986.

[34] R. M. Karp, E. Upfal, and A. Wigderson. The complexity of parallel search. *J. Comput. System Sci.*, 36(2):225–253, 1988.

[35] P. Kohli, M. P. Kumar, and P. H. S. Torr. P3 and beyond: Move making algorithms for solving higher order functions. *IEEE Trans. Pattern Anal. and Machine Learning*, 31:1–8, 2008.

[36] S. Lacoste-Julien and M. Jaggi. On the global linear convergence of Frank-Wolfe optimization variants. In *Adv. in Neu. Inf. Proc. Sys. (NeurIPS)*, 2015.

[37] Y. T. Lee, A. Sidford, and S. C.-W. Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. *Proc., IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1049–1065, 2015.

[38] W. Li, P. Liu, and J. Vondrák. A polynomial lower bound on adaptive complexity of submodular maximization. In *Proc., ACM Symposium on Theory of Computing (STOC)*, pages 140–152, 2020.

[39] H. Narayanan, H. Saran, and V. V. Vazirani. Randomized parallel algorithms for matroid union and intersection, with applications to arborescences and edge-disjoint spanning trees. *SIAM J. Comput.*, 23(2):387–397, 1994.

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

[41] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions – I. *Math. Programming*, 14(1):265–294, 1978.

[42] A. Nemirovski. On parallel complexity of nonsmooth convex optimization. *Journal of Complexity*, 10(4):451–463, 1994.

[43] J. B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Math. Programming*, 118(2):237–251, 2009.

[44] A. Rubinstein, T. Schramm, and S. M. Weinberg. Computing exact minimum cuts without knowing the graph. In *Proc., Innovations in Theoretical Computer Science (ITCS)*, pages 39:1–39:16, 2018.

[45] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Combin. Theory Ser. B*, 80(2):346–355, 2000.

[46] J. Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM J. Comput.*, 42(1):265–304, 2013.