Sparse Submodular Function Minimization

Andrei Graur
Management Science and Engineering
Stanford University
Stanford, USA
agraur@stanford.edu

Haotian Jiang
Microsoft Research
Redmond, USA
jhtdavid96@gmail.com

Aaron Sidford
Management Science and Engineering
Stanford University
Stanford, USA
sidford@stanford.edu

***Abstract*—In this paper we study the problem of minimizing a submodular function $f : 2^V \to \mathbb{R}$ that is guaranteed to have a k-sparse minimizer. We give a deterministic algorithm that computes an additive ϵ-approximate minimizer of such f in $\widetilde{O}(\mathsf{poly}(k)\log(|f|/\epsilon))$ parallel depth using a polynomial number of queries to an evaluation oracle of f, where $|f| = \max_{S\subseteq V}|f(S)|$. Further, we give a randomized algorithm that computes an exact minimizer of f with high probability using $\widetilde{O}(|V|\cdot\mathsf{poly}(k))$ queries and polynomial time. When $k = \widetilde{O}(1)$, our algorithms use either nearly-constant parallel depth or a nearly-linear number of evaluation oracle queries. All previous algorithms for this problem either use $\Omega(|V|)$ parallel depth or $\Omega(|V|^2)$ queries.**

In contrast to state-of-the-art weakly-polynomial and strongly-polynomial time algorithms for SFM, our algorithms use first-order optimization methods, e.g., mirror descent and follow the regularized leader. We introduce what we call *sparse dual certificates*, which encode information on the structure of sparse minimizers, and both our parallel and sequential algorithms provide new algorithmic tools for allowing first-order optimization methods to efficiently compute them. Correspondingly, our algorithm does not invoke fast matrix multiplication or general linear system solvers and in this sense is more combinatorial than previous state-of-the-art methods.

***Index Terms*—submodular function minimization, convex optimization, sparsity, query complexity, parallel complexity**

I. Introduction

Submodular function minimization (SFM) is a foundational problem in combinatorial optimization. Submodular functions encompass a wide range of functions that appear naturally in practical applications, including graph cut functions, matroid rank functions, set coverage functions, and utility functions from economics. Since seminal work of Edmonds in 1970 [Edm70], SFM has served as a central tool in many areas such as theoretical computer science, operations research, game theory, and recently, machine learning. We refer interested readers to surveys [McC05], [Iwa08] for a more comprehensive account of the rich history of SFM.

Throughout this paper we consider a standard setting for SFM. We are given a set function $f : 2^V \to \mathbb{R}$, where V is an n-element finite set, known as the *ground set*, and f is submodular, i.e.,

$$f(S \cup \{i\}) - f(S) \geq f(T \cup \{i\}) - f(T),$$

for all $S \subseteq T \subseteq V$ with $i \notin T$. Furthermore, we assume that f is accessed only through an *evaluation oracle* which when queried at any $S \subseteq V$ outputs $f(S)$ in time EO. We let $|f| \stackrel{\text{def}}{=} \max_{S\subseteq V}|f(S)|$ and $f^* \stackrel{\text{def}}{=} \min_{S\subseteq V} f(S)$ and consider the problem of computing an *ϵ-approximate minimizer*, i.e., $S \subseteq V$ with $f(S) \leq f^* + \epsilon$.

Since seminal work of Grötschel, Lovász, and Schrijver [GLS81] showed that SFM can be solved in polynomial time, there have been multiple advances in SFM over the last few decades [Sch00], [IFF01], [FI03], [Iwa03], [Vyg03], [Orl09], [IO09], [CJK14], [LJJ15], [CLSW17]. In this paper, we focus on algorithms that solve SFM to *high accuracy* with a *polynomial query complexity*, meaning that they solve the problem with a number of queries to an evaluation oracle that scale *weakly-polynomially* ($\mathsf{poly}(n, \log(|f|/\epsilon))$) [GLS81] or *strongly-polynomially* ($\mathsf{poly}(n)$) [GLS84], [GLS88].[1] Current state-of-the-art SFM algorithms in these regimes are weakly-polynomial $\widetilde{O}(n^2\log(n|f|/\epsilon))$-query, polynomial-time algorithms [KTE88], [NN89], [Vai89], [BV04], [LSW15], [JLSW20], strongly-polynomial $O(n^3)$-query, polynomial-time algorithms [LSW15], [DVZ21], [Jia22], and a strongly-polynomial

The full version of this paper can be found on arXiv at https://arxiv.org/abs/2309.16632.

[1] When f is integer valued, any $\epsilon < 1$ approximate solution is optimal; a variety of the prior work consider only this setting. Throughout the paper we do not distinguish between prior work which consider exactly solving SFM integer valued f (with a dependence on $|f|$) and those that work in the more general setting we consider in this paper.

$\widetilde{O}(n^2)$-query, exponential-time algorithm [Jia22] (see Section I-C for more details)[2].

On the hardness side, however, the current state-of-the-art lower bounds exclude algorithms making fewer than $\Omega(n \log n)$ queries in the strongly-polynomial regimes [CGJS22] and fewer than $\Omega(n)$ queries in the weakly-polynomial regime [Har08], [CLSW17]. Consequently, there are large, $\Omega(n)$ gaps, between these lower bounds and the best known upper bounds. Unfortunately, obtaining nearly-linear (or provably near-optimal) query complexity algorithms for SFM has been elusive.

In light of these developments, it is natural to ask, what additional structural assumptions may be needed to enable faster algorithms? One recent line of work has explored the complexity of *decomposable SFM* [JBS13], [NJJ14], [EN15], [ENV17], [KBP19], [AKM+21], [DJL+22], that is the special case where $f(S) = \sum_i f_i(S \cap T_i)$ for submodular f_i and sparse T_i given an oracle for evaluating the individual f_i over T_i. A different line of work [CLSW17], [ALS20] considers the complexity of *approximate* SFM when the minimizer is k-sparse, which we refer to as *k-sparse SFM* for brevity.[3] We refer to an SFM algorithm as approximate, if its query complexity is *pseudo-polynomial*, i.e., $O(\mathsf{poly}(n, |f|/\epsilon))$. The state-of-the-art approximate k-sparse SFM algorithm has a query complexity of $\widetilde{O}(k(|f|/\epsilon)^2)$, when f is integer valued and $\epsilon < 1$.

In both of these cases, sparsity plays a prominent role. In the specific context of SFM, while various polyhedral and geometric properties of submodular functions have been extensively studied and heavily exploited since the 1970s [Edm70], these properties are mostly global, involving the entire set V altogether. On the other hand, assuming k-sparsity of the minimizer allows one to take a glimpse into local properties of submodularity, e.g., to understand the role a small number of elements play for the minimization of the function.

Moreover, sparsity of the minimizer is a natural assumption in convex optimization and submodular function minimization problems. In particular, sparsity arises in signal processing, feature selection, compressed sensing, etc. where the solution is often expected to be sparse, i.e., have a small number of non-zero elements [Don06], [MAH+12], [LWR20]. Sparsity is also common in cases where a regularizer is added to the objective function to encourage sparsity. One example of such a setup is the problem of finding an optimal dataset for speech recognition tasks [LB11]. This problem can be written as $f(S) + \lambda|S|$, where f is a submodular objective, and therefore it is expected that the size of the minimizing set is much smaller than the ground set for large values of the regularization coefficient λ. Consequently, understanding how the complexity of algorithms depends on the sparsity leads to better insight into more refined combinatorial and geometric structures of the problems. Therefore, the central question we ask in this paper is:

Can we leverage sparsity to improve upon state-of-the-art polynomial query complexities?

k-sparse SFM is also interesting in light of recent work [BS20] seeking to clarify the *parallel depth* of SFM, i.e., the number of parallel rounds of queries to the evaluation oracle required for a query-efficient algorithm. The state-of-the-art parallel depth lower bounds are $\Omega(n/\log n)$ in the strongly-polynomial regime [CGJS22], which matches the upper bound in [Jia22] up to a factor of $\log^2 n$, and $\widetilde{\Omega}(n^{1/3})$ in the weakly-polynomial regime [CCK21]. These polynomial parallel depth lower bounds crucially rely on the minimizers being dense for the constructed submodular functions, and highly parallel algorithms might be possible when the submodular function admits a sparse minimizer. Therefore, we also ask: *Can we improve the parallel complexities for k-sparse SFM?* Besides being interesting from an algorithmic perspective, obtaining improved parallel algorithms for k-sparse SFM could aid lower bound development by showing how hard-instances for lower bounds must have dense minimizers.

A. Challenges and Additional Motivations

Beyond intrinsic interest in improving the complexity of k-sparse SFM, this problem is also an interesting testbed for new techniques and a number of larger open problems on SFM. Here we briefly elaborate on these challenges and motivations for studying k-sparse SFM.

State-of-the-art SFM algorithms typically leverage the *Lovász extension* [Lov83] of f, a convex function $\hat{f} : [0,1]^V \to \mathbb{R}$ that agrees with f on the hypercube's vertices, i.e., $\hat{f}(\vec{1}_S) = f(S)$ for all $S \subseteq V$. It is known that $\hat{f}$ can be evaluated efficiently and minimizing $\hat{f}$ suffices for SFM. Consquently, SFM algorithms can be readily obtained by applying convex optimization methods to the Lovász extension. Indeed, state-of-the-art weakly-polynomial SFM algorithms [LSW15], [JLSW20] follow this approach by using *cutting plane methods*, a class of weakly-polynomial convex optimization methods, to obtain ϵ-approximate minimizers in $\widetilde{O}(n \log(1/\epsilon))$ parallel rounds of $\widetilde{O}(n)$ queries per

[2]Throughout the paper we use $\widetilde{O}(\cdot)$ to hide $O(\mathsf{poly}(\log n))$ factors.

[3]This problem is distinct from that of computing the minimum value k-sparse set for a submodular function.

round. State-of-the-art strongly-polynomial SFM algorithms [LSW15], [DVZ21], [Jia22] carefully apply these weakly-polynomial cutting plane methods iteratively.

With the k-sparsity assumption on the solutions, a natural approach would be to apply these continuous optimization methods to minimize $\hat{f}$ over $S_k^V \stackrel{\text{def}}{=} \Delta_k^V \cap [0,1]^V$, where $\Delta_k^V \stackrel{\text{def}}{=} \{x \in \mathbb{R}_{\geq 0}^V | \, \|x\|_1 \leq k\}$ is the interior of the simplex scaled up by k; this suffices for k-sparse SFM since $\vec{1}_{S^*} \in S_k^V$ for the k-sparse minimizer $S^* \subseteq V$. Unfortunately, while changing the domain from $[0,1]^V$ to S_k^V is known to improve the performance of certain pseudo-polynomial convex optimization methods (as in [CLSW17], [ALS20]), it is not known to improve the performance of weakly-polynomial convex optimization algorithms (e.g., state-of-the-art cutting plane method [JLSW20]) by more than logarithmic factors. Furthermore, without using more of the structure $\hat{f}$ it seems unlikely that this change of domain would affect the weakly-polynomial complexity by more than logarithmic factors, since one could scale a hard convex optimization problem to fit inside S_k^V without changing problem parameters by more than a polynomial factor.

These challenges call for the development of new optimization techniques that better utilize structures of the Lovász extension and sparsity of the domain, which might lead to applications for a broader range of open problems on SFM. We note several of these additional motivations below.

a) Strongly-polynomial time $O(n^{3-c})$-query algorithm for SFM: One of the most important motivations is towards improving strongly-polynomial time SFM algorithms. The current best query complexity here is $O(n^3 \log\log n / \log n)$ given in [Jia22], but this approach seems unlikely to provide further improvement given the stagnation of progress on obtaining a better approximation factor for the shortest vector problem, on which the algorithm in [Jia22] crucially relies.

Other state-of-the-art strongly-polynomial time SFM algorithms with $\widetilde{O}(n^3)$ query complexities in [LSW15], [DVZ21] learn precedence constraints of the form, if $p \in V$ is in a minimizer then so is q (e.g., [IFF01], [IO09], [LSW15], [DVZ21]). In the worst case, these algorithms might make $\widetilde{O}(n^2)$ queries to learn only a single coordinate that must be in a minimizer (or not), or for many coordinates $p \in V$ a single $q \in V$ that must be in any minimizer containing p. This worst-case behavior is a key barrier towards obtaining strongly-polynomial time algorithms with $O(n^{3-c})$ query complexities for constant $c > 0$. However, this worst-case behavior is sparse, and k-sparse SFM algorithms which better exploit local properties of submodular functions might be useful to get around the aforementioned barrier in this case and lead to a smaller query complexity.

b) SFM versus continuous optimization: Given the challenges of adapting weakly-polynomial convex optimization algorithms to leverage sparsity, obtaining weakly- and strongly-polynomial algorithms for k-sparse SFM could highlight differences between general convex optimization and SFM. Consequently, k-sparse SFM is a natural proving grounds for designing SFM algorithms that go beyond using the boundedness and convexity of the Lovász extension.

c) Combinatorial algorithms and iteration costs: The use of cutting plane methods in state-of-the-art SFM algorithms comes with certain inherent costs. Key among them is that all known cutting plane methods apply general linear system solvers or matrix multiplication methods, making these methods somewhat intrinsically non-combinatorial. This is inherent as ultimately the problems they solve are more general than that of solving arbitrary linear systems.

Since, as argued above, obtaining better query complexities for weakly- and strongly-polynomial k-sparse SFM suggests departing from cutting plane methods, the problem could be an interesting one to see where more combinatorial methods or ones with lower iteration costs can shine. State-of-the-art pseudo-polynomial SFM algorithms leverage optimization machinery which does not use linear system solves and correspondingly have runtimes that are within polylogarithmic factors of their query complexity [CLSW17], [ALS20]. Though there have been efforts in using alternative optimization methods to solve SFM, e.g., [DVZ21], the query complexities of such methods are much higher than the state-of-the-art. Correspondingly, k-sparse SFM is an interesting setting to see whether such methods can outperform cutting plane methods.

B. Our Results

Our main results include two algorithms which improve, respectively, the parallel depth and query complexities of polynomial-time k-sparse SFM algorithms.

a) Parallel depth for k-sparse SFM: In the parallel model for SFM (in the weakly-polynomial regime), the algorithm can submit up to $\mathsf{poly}(n, \log(|f|/\epsilon))$ parallel queries to the evaluation oracle in each round, and its parallel *depth* is defined to be the number of rounds needed to find the minimizer in the worst case. Our main result for this model is the following theorem.

Theorem I.1 (Parallel k-sparse SFM)**.** *There is a deterministic parallel algorithm for k-sparse SFM with parallel depth $\widetilde{O}(k^7 \cdot \log(|f|/\epsilon))$ and runtime $\widetilde{O}(n^2 \cdot k^7 \log(|f|/\epsilon) \cdot \mathsf{EO} + \mathsf{poly}(n) \cdot \log(|f|/\epsilon))$.*

When the sparsity $k = \widetilde{O}(1)$, the parallel depth in Theorem I.1 is $\widetilde{O}(1)$. To the best of our knowledge, this is the first nearly-constant parallel depth result for SFM, beyond the trivial n^k-query algorithm that queries all k-sparse sets in a single round (which does not have polynomial query complexity whenever $k = \omega(1)$).

Our result is in stark contrast to the best known weakly-polynomial parallel depth of $\widetilde{O}(n)$ for general SFM [LSW15]. It is important to emphasize here that $\widetilde{O}(1)$-sparsity is also *necessary* for obtaining a nearly-constant parallel depth. The work of [CCK21] implies that $\widetilde{\Omega}(k^{1/3})$ parallel depth is required for any weakly-polynomial algorithm for k-sparse SFM.

b) Query complexity for k-sparse SFM: While the algorithm in Theorem I.1 achieves a nearly-constant parallel depth when the sparsity is nearly-constant, even in this setting its query complexity is $\Omega(n^2)$. In light of the question of designing SFM algorithms with nearly-linear query complexity, our second main result is a pair of algorithms which improve the weakly- and strongly-polynomial query complexities for k-sparse SFM. (It remains open as to whether the parallel depth of strongly-polynomial k-sparse SFM can be similarly improved.)

Theorem I.2 (Weakly-polynomial k-sparse SFM)**.** *There is a randomized algorithm that outputs an ϵ-approximate minimizer for k-sparse SFM whp. in $\widetilde{O}((n \cdot \mathsf{poly}(k) \cdot \mathsf{EO} + \mathsf{poly}(n)) \log(|f|/\epsilon))$ time.*

Theorem I.3 (Strongly-polynomial k-sparse SFM)**.** *There is a randomized algorithm that outputs an exact minimizer for k-sparse SFM whp. in $\widetilde{O}(n \cdot \mathsf{poly}(k) \cdot \mathsf{EO} + \mathsf{poly}(n))$ time.*

We include both theorems above because the $\mathsf{poly}(k)$ in Theorem I.2 is slightly better than that in Theorem I.3 (see full version for more details). The algorithms in Theorems I.2 and I.3 have nearly-linear query complexities when the sparsity $k = \widetilde{O}(1)$. Previously, the only nearly-linear weakly-polynomial query complexity results for SFM were obtained when the submodular function f can be decomposed as $f(S) = \sum_i f_i(S)$ and each f_i depends only on $\widetilde{O}(1)$ coordinates [AKM+21], [DJL+22]. However, this is different and the techniques for solving it seem tailored to its structure.

Our algorithms for Theorems I.1-I.3 depart from the use of cutting plane methods and do not rely on linear system solves as a sub-procedure. In this sense, they are more combinatorial than state-of-the-art weakly-polynomial time [LSW15], [JLSW20] and strongly-polynomial time SFM algorithms [LSW15], [DVZ21], [Jia22]. Somewhat surprisingly, our algorithms combine first-order methods, which have been primarily used for pseudo-polynomial SFM algorithms (e.g., [CLSW17], [ALS20]), and arc finding, a technique central to many strongly-polynomial SFM algorithms (e.g., [LSW15], [DVZ21]), to obtain very efficient weakly- and strongly-polynomial time algorithms. Previous combination of these two techniques only appeared in [DVZ21], but the resulting algorithm has query complexity and parallel depth at least a factor of n^2 larger than the state-of-the-art algorithms based on cutting plane methods. The proofs of Theorems I.2 and I.3 additionally invoke various sampling techniques, which crucially allows us to save the additional factor of n from querying an entire subgradient of the Lovász extension in each iteration.

C. Related Work

SFM is a central combinatorial optimization problem with extensive applications. The problem of maximizing a submodular function has also been widely studied, but is very different and has seemingly different structure, algorithms, and history (see, e.g., [KG14] for a survey on this topic).

a) Strongly-, weakly-, and pseudo- polynomial algorithms for SFM: As discussed in the intro, a fundamental result for SFM is that it can be solved efficiently, in all three regimes of weakly-, strongly-, and pseudo-polynomial. The first weakly- and strongly-polynomial time SFM algorithms were given in the seminal work of Grötschel, Lovász, and Schrijver [GLS81], [GLS84], [GLS88]. The first pseudo-polynomial algorithm for SFM was given in a seminal work of Cunningham [Cun85]. Since then, there has been a long line of work on designing better algorithms for SFM in all three regimes [Sch00], [IFF01], [FI03], [Iwa03], [Vyg03], [Orl09], [IO09], [CJK14], [LJJ15], [LSW15], [CLSW17], [DVZ21]. The state-of-the-art algorithms for these regimes are shown in Table I.

b) Parallel SFM: For the parallel complexity of SFM discussed earlier in the intro, the current best weakly-polynomial algorithm has parallel depth $O(n \log nM)$ [LSW15] and the current best strongly-polynomial algorithms [Jia22] have parallel depth $O(n \log n)$ (with exponential runtime) or $O(n^2 \log\log n / \log n)$ (with polynomial runtime). In concurrent work [CGJS23], a superset of the authors give a $\widetilde{O}(n^{1/3}/\epsilon^{2/3})$-round $\mathsf{poly}(n)$-time algorithm for

Paper	Year	Running Times	Remarks
[JLSW20]	2020	$O(n^2 \log nM \cdot \mathsf{EO} + n^3 \log nM)$ $O(n^3 \log^2 n \cdot \mathsf{EO} + n^4 \log^2 n)$	current best weakly & strongly runtime
[ALS20]	2020	$\widetilde{O}(nM^2 \cdot \mathsf{EO} + \mathsf{poly}(n))$ $\widetilde{O}(kM^2 \cdot \mathsf{EO} + \mathsf{poly}(n))$	current best pseudo-poly current best sparse pseudo-poly
[Jia22]	2021	$O(n^3 \log \log n / \log n \cdot \mathsf{EO} + \mathsf{poly}(n))$ $O(n^2 \log n \cdot \mathsf{EO} + \exp(n))$	current best strongly query complexity

TABLE I: State-of-the-art weakly-, strongly-, and pseudo-polynomial algorithms for submodular function minimization. k is the sparsity and parameter $M = |f|/\epsilon$.

obtaining an ϵ-approximate minimizer, and a 2-round $n^{O(M)}$-time algorithm for computing an exact minimizer. As discussed in the intro, lower bounds for parallel SFM have also been studied recently (see Table II).

Paper	Year	Parallel Depth	Accuracy
[BS20]	2020	$\Omega(\log n / \log \log n)$	exact
[CCK21]	2021	$\widetilde{\Omega}(n^{1/3})$	$\|f\|/\mathsf{poly}(n)$
[CGJS22]	2022	$\Omega(n / \log n)$	exact

TABLE II: Parallel depth lower bounds for query-efficient SFM. In the "Accuracy" column, "exact" means the algorithm is required to compute an exact minimizer, and "$|f|/\mathsf{poly}(n)$" means the algorithm is allowed to output any approximate minimizer with an additive accuracy of $|f|/\mathsf{poly}(n)$.

c) Structured SFM: Given the aforementioned nearly n-factor gap between the state-of-the-art query complexity upper and lower bounds for SFM, there have been exciting recent results on improving the query complexity of SFM assuming more fine-grained structures of the submodular functions. In particular, for the problem of decomposable SFM discussed prior to Section I-A, it is known that f can be minimized in weakly-polynomial time using $\widetilde{O}(n)$ total queries to the evaluation oracles of each individual f_i [AKM+21], [DJL+22].

II. Our Approach

Here we provide an overview of our approach towards proving Theorems I.1-I.3. We first give some context and motivation, and then we cover the key components of our approach in Sections II-A-II-C.

To situate our approach, recall that previous state-of-the-art weakly- and strongly-polynomial time SFM algorithms all apply the general continuous optimization tool of *cutting plane methods* [Lev65], [New65], [Sho77], [YN76], [Kha80], [KTE88], [NN89], [Vai89], [BV04], [LSW15], [JLSW20]. Cutting plane methods are known to compute ϵ-approximate minimizers of bounded convex functions on $\mathbb{R}^n$ in $\widetilde{O}(n \log(1/\epsilon))$ iterations where each iteration consists of a subgradient computation, which typically takes $\widetilde{O}(1)$ depth, $O(n)$ queries to the evaluation oracle of f, and $\Omega(n^2)$ additional work [JLSW20] involving linear algebraic operations such as a linear system solve.

In this paper we seek to improve upon these methods for k-sparse SFM both in terms of performance and to avoid general linear algebraic primitives (to obtain, in some sense, a more combinatorial algorithm). However, as discussed in Section I-A, it is unclear how to substantially improve cutting plane methods just using the assumption that their is a sparse optimal solution.

Consequently, we depart from previous state-of-the-art weakly- and strongly-polynomial SFM algorithms and instead use first-order methods[4] such as mirror descent (see Section 4.2 in [B+15]) and (stochastic) follow-the-regularized-leader (referred to as "lazy mirror descent" in Section 4.4 of [B+15]) to minimize the Lovász extension. These methods have performance depending more on problem geometry, e.g., the domains B_∞^V versus S_k^V, than cutting plane methods. Also, implementing them often does not require linear system solves and therefore they typically have much smaller iteration costs.

Unfortunately, these desirable features of first-order methods have a cost. In contrast to cutting plane methods, when applied to non-smooth convex objectives like the Lovász extension, their convergence rate depends polynomially on the accuracy rather than polylogarithmically. Therefore, it is natural to use such methods for pseudo-polynomial SFM algorithms [CLSW17], [ALS20], but less clear how to leverage them to obtain improved weakly- or strongly- polynomial SFM algorithms.

Fortunately, recent advances in weakly- and strongly-polynomial SFM algorithms provide hope for overcoming this limitation. Work of [LSW15], [DVZ21] provide different ways to incorporate learned *precedence* con-

[4]Technically speaking, cutting-plane methods also only use first-order information. However, following the conventions of the optimization literature, we do not refer to cutting plane methods as first-order methods.

straints, i.e., if an element is in a minimizer then what other elements must also be in that minimizer, to reduce the scale of the problem. For example, [DVZ21] showed that it suffices to solve SFM approximately to a relative accuracy of $O(1/n^3)$, in a primal-dual sense, repeatedly to obtain a strongly-polynomial algorithm for SFM.

Despite the above hope for improving k-sparse SFM via first-order methods, there are a number of natural hurdles in the way. For example, the $O(1/n^3)$-error requirement in [DVZ21] is prohibitively expensive for first-order methods to outperform cutting plane methods. Additionally, learning and updating precedence constraints need to be made sufficiently efficient.

Nevertheless, we are able to follow this broad approach by introducing and leveraging a central concept of this paper we call *sparse dual certificates* (see Section 4.3 in the full version for more details). In particular, we demonstrate how to carefully apply first-order methods to $1/\mathsf{poly}(k)$-accuracy to compute sparse dual certificates and, building upon [DVZ21], how these certificates can be used to efficiently deduce precedence constraints. Our parallel and sequential algorithms differ in their specific implementations of these strategies. We believe the notion of sparse dual certificates and our algorithmic techniques for computing and using them for k-sparse SFM might have broader applications to improving weakly- or strongly- polynomial time SFM algorithms.

a) Section organization: To illustrate our approach and our key insights, we subdivide the remainder of this section. In Section II-A, we provide the general framework we use to iteratively decrease the scale of the k-sparse SFM problem. Section II-B and Section II-C, we provide the key ideas in our parallel and sequential algorithms respectively.

A. Framework

Building on a long line of work [IFF01], [IO09], [LSW15] (and in particular, [DVZ21]), our algorithms for minimizing a submodular function $f : 2^V \to \mathbb{R}$ works by maintaining a set of precedence constraints indicating elements that must or must not be in any k-sparse minimizer, as well as for each $p \in V$ a set of elements S_p that must be in any k-sparse minimizer S^* containing p. We call these precedence constraints *arc constraints*[5] and their collection a *ring family*.

Given these arc constraints, we consider an induced *submodular extension* $f^\sharp$ consistent with the ring family. $f^\sharp$ is essentially the complement of a submodular extension studied in [DVZ21]; it is crucial that we work with $f^\sharp$ since sparsity is not preserved under complements. The extension $f^\sharp$ has many desirable properties. For example, minimizing $f^\sharp$ suffices for minimizing f and any arc constraints learned for $f^\sharp$ apply to f. Beyond consistency and submodularity, the key property we use about $f^\sharp$ is that the marginal vector[6] $u \in \mathbb{R}^V$ defined as $u_p \stackrel{\text{def}}{=} f^\sharp(\{p\}) - f^\sharp(\emptyset)$ for any coordinate $p \in V$ does not increase as we add arc constraints.

By maintaining the ring family and the extension $f^\sharp$, and leveraging their properties, k-sparse SFM reduces to the problem of learning new arc constraints so that we can either

1) decrease the scale of $\|u\|_\infty$ by more than a constant factor, or
2) learn enough arc constraints so that the k-sparse minimizer is clear.

In particular, if $\|u\|_\infty \le \varepsilon/|V|$, then due to submodularity the largest set consistent with every arc constraint will be an ε-approximate minimizer for the original submodular function. Note how k-sparsity helps for our purposes: if the set of elements S_p that must be in every k-sparse minimizer containing p has more than k elements, then p cannot be in any k-sparse minimizer and can therefore be discarded. This allows us to maintain at most k arc constraints from any element $p \in V$, which significantly decreases the cost of manipulating the arc constraints and the submodular extension $f^\sharp$.

In both our parallel and sequential settings, we use $\|u\|_\infty$ as a potential function and design efficient parallel and sequential subprocedures to find arc constraints to decrease $\|u\|_\infty$ by a constant factor. Each setting has its distinct challenges and our techniques differ correspondingly. However, there is one common technique underlying these two different implementations, based on the notion of a *sparse dual certificate* (see Section 4.3 in the full version for details).

a) Sparse dual certificates: Sparse dual certificates are generalizations of standard dual solutions to SFM [Edm70] that better capture the sparsity assumptions on the minimizers of the submodular function. In our framework, sparse dual certificates bridge the gap between the task of finding arc constraints and the pseudo-polynomial convergence rate of first-order methods. In particular, we show how to use sparse dual certificates to deduce arc constraints (see Section 4.3 in the full version). We also

[5]Our definition of arc constraints is only with respect to k-sparse minimizers and is therefore different from the standard one in the literature. See Section 4.1 in the full version for more details.

[6]The formal notation we define and use for the marginal vector in Section 3 of the full version is $u_{f^\sharp}$. Here we drop the subscript and use u instead for simplicity.

develop various algorithmic techniques to compute these certificates by running first-order methods up to only $1/\mathsf{poly}(k)$ accuracy (see Sections 5 and 6 respectively in the full version for our parallel and sequential algorithms for computing sparse dual certificates).

B. Parallel Algorithm

To motivate our parallel algorithm, consider minimizing the Lovász extension $\hat{f}$ of the induced function $f^\sharp : 2^V \to \mathbb{R}$ with a k-sparse minimizer and let $f^* \stackrel{\text{def}}{=} \min_{S \subseteq V} f^\sharp(S) = \min_{S \subseteq V} f(S)$. As discussed above, it suffices to learn arc constraints so that we can decrease $\|u\|_\infty$ by a constant factor after updating the ring family. We may assume that $\min_{p \in V} u_p \geq 0$ as by submodularity adding any p with $u_p < 0$ to any set decreases its value and therefore p must be in every minimizer.

As a warm-up for this goal, perhaps the first natural question is: under these assumptions how efficiently can we compute a $\delta \|u\|_\infty$-approximate minimizer for a given $\delta = 1/\mathsf{poly}(k)$? The question of deducing arc constraints seems harder than this problem since it involves proving something about all k-sparse minimizers at that accuracy threshold. For this warm-up question, let us even assume for now that $f^* \geq -\Omega(\|u\|_\infty)$, as the problem is in some sense easier otherwise and will be addressed towards the end of this subsection.

A natural approach to this warm-up problem, as alluded to earlier, is to apply standard first-order methods such as mirror descent to the Lovász extension $\hat{f}$ of $f^\sharp$ over the domain S_k^V. By submodularity, u entrywise upper bounds the subgradients of $\hat{f}$. If somehow the subgradients were also entrywise lower bounded by $-\|u\|_\infty$, then standard analysis of mirror descent with an entropy regularizer (see Theorem 4.2 of [B+15]) applied to $\hat{f}$ over S_k^V would compute an $\delta \|u\|_\infty$-approximate minimizer in $\widetilde{O}(\delta^{-2})$ iterations. Furthermore, since each iteration of this method can be implemented in $O(1)$ depth, this would yield a $\widetilde{O}(\delta^{-2})$ depth algorithm as desired.

Unfortunately, it is not necessarily the case that every subgradient of $\hat{f}$ is entrywise lower bounded in magnitude by $-\|u\|_\infty$. In fact, its most negative entry can be as negative as $f^* - (n-1)\|u\|_\infty$, ruling out showing that mirror descent converges in $\widetilde{O}(\mathsf{poly}(k, \delta^{-1}))$ iterations.

To overcome this issue, we show that the structure of k-sparse solutions allows us to *truncate* subgradients. We prove that if we run mirror descent methods with every subgradient coordinate of value $\leq f^* - k\|u\|_\infty$ set to $f^* - k\|u\|_\infty$, then this still approximately minimizes the Lovász extension $\hat{f}$ and computes sparse dual certificates. Running mirror descent with these truncated subgradients yields a deterministic algorithm which computes a $\delta\|u\|_\infty$-approximate minimizer in $\widetilde{O}(\mathsf{poly}(k)/\delta^2)$ depth and $\widetilde{O}(n \cdot \mathsf{poly}(k)/\delta^2)$ evaluation oracle queries.

The solution to this warm-up problem is the key ingredient in our parallel algorithm. In particular, assuming $f^* \leq -\|u\|_\infty$, we show that the sparse dual certificate obtained by running the warm-up algorithm over S_V^{k+1} with accuracy $O(\|u\|_\infty/k)$ suffices to conclude an element that must be in every k-sparse minimizer, i.e., a dimensionality reduction. As dimensionality reduction can occur at most k times, this gives a $\widetilde{O}(\mathsf{poly}(k))$-depth $\widetilde{O}(n \cdot \mathsf{poly}(k))$-query algorithm.

On the other hand, when $f^* \geq -\|u\|_\infty$, then we consider each of the induced submodular functions f_p where an element p is always included and run the same algorithm on each such function. Note that each f_p, once shifted to evaluate 0 at the new emptyset (or the singleton $\{p\}$), has minimum value $-\Omega(u_p)$. Consequently, when this is done for p with u_p near $\|u\|_\infty$, the procedure finds an element which must be in any k-sparse minimizer containing p. Importantly, this can be done in parallel for each individual p! This conveys the main ideas behind the parallel algorithm.

C. Sequential Algorithm

In the previous section we outlined the main ideas of our parallel algorithm in Theorem I.1. Unfortunately, that algorithm has a rather high query complexity. In every round of decreasing $\|u\|_\infty$, the algorithm might solve n different induced SFM problems, corresponding to the inclusion of each element $p \in V$, causing the query complexity to scale quadratic in n instead of linear.

In the literature on using ring families for weakly- and strongly-polynomial SFM, there is a standard technique for alleviating the need to apply the algorithm to n different SFM problems to deduce arcs. In [LSW15], [DVZ21] and early SFM algorithms (see [LSW15] for a discussion of the history), the algorithm obtains a suitable dual certificate for the original function. This dual certificate is then modified by moving individual elements to the start of each permutation, and it is argued that this modification can be used to deduce arcs. In other words, rather than running n optimization methods to deduce n dual certificates, these methods deduce one set of dual certificates and consider n different modifications of it.

In our sequential algorithm we follow a similar approach, but it brings about a variety of challenges, each of which requires algorithmic and analytic insights to

overcome. The first challenge is that truncated subgradients, which are used in our parallel algorithm, do not seem amenable to this technique; it is unclear how to deduce arcs just by moving elements to the front after the truncation, which may lose critical information that makes this work. [LSW15], [DVZ21] consider the elements of the dual certificate that decrease significantly when moving a certain element to the start of each permutation. However, truncation does not seem to allow for a similar approach, as all components that are negative past a threshold are truncated to the same value.

To overcome this challenge, we provide a first-order method for computing ϵ-approximate minimizers (and their associated sparse dual certificates) using true (rather than truncated) subgradients. As discussed in Section II-B, this is difficult as the entries of the subgradient can vary by $\Omega(n \|u\|_\infty)$. Correspondingly, standard analysis of iterative first-order methods, e.g., mirror descent and FTRL (Follow-the-Regularized-Leader), require $\Omega(n^2)$ iterations, which would naively make a prohibitive $\Omega(n^3)$ queries! It is therefore imperative that we use a different technique (other than truncation as in the parallel setting) to either reduce the number of iterations or the cost per iteration; we do both. In particular, we use stochastic FTRL[7] (Follow-the-Regularized-Leader) where in each iteration we sample a random 1-sparse unbiased estimator of the subgradient. We show how this can be implemented using $\widetilde{O}(1)$ evaluation queries per iteration and that the total number of iterations is suitably bounded.

Making the above approach work requires a number of insights. First, the number of iterations of stochastic FTRL is straightforwardly boundable in terms of the square of the ℓ_∞ norm of the stochastic estimates of the subgradient (analogous to as it was done for mirror descent in the parallel setting). However, unfortunately any sampling scheme in the worst case could have an ℓ_∞-norm of $\Omega(n \|u\|_\infty)$, again leading to $\Omega(n^2)$ iterations. To get around this, we instead perform a more fine-grained analysis of the convergence of FTRL in terms of "local norms" (see Section 6.1 in the full version for details). This is a known optimization method analysis technique and our analysis is inspired from and perhaps most closely resembles [CJST19]; this technique was not used in previous work on SFM that uses sampling [CLSW17], [HRRS19], [ALS20].

The next challenge is to actually implement sampling using $\widetilde{O}(1)$ queries per iteration so that the local norms of the samples are suitably small. Sampling $i \in V$ with probability proportional to $|(g_{x_t})_i|$ and then outputting $\mathsf{sign}(g_{x_t})_i \cdot \|g_{x_t}\|_1$ would have the desired local norm bound. Additionally, sampling by this is essentially what is done in some of the sampling-based SFM methods [CLSW17], [HRRS19], [ALS20] (albeit for a different norm analysis particularly relevant for pseudopolynomial SFM algorithms). However, these papers implement this sampling by somewhat complex dynamic data structure which could be challenging to analyze in our setting. Instead, we provide a simple straightforward sampling procedure which we call vSampling. This sampling scheme picks $i \in V$ proportional to an upper bound v_i for $|(g_{x_t})_i|$ so that $\sum_{i \in I} v_i$ for consecutive coordinates I can be evaluated using only $O(1)$ queries. This sampling can be implemented using $O(\log n)$ queries by a simple (static) binary tree data structure and we prove it has the desired expected local-norm bounds.

Another challenge we face is that our stochastic FTRL analysis merely yields a subgradient y that is a suitable dual certificate *in expectation*, whereas we need the guarantee to hold with high probability in order to correctly deduce arc-constraints. To this end, we show that $\|y\|_\infty$ is small with high probability[8], and apply the Azuma-Hoeffding concentration inequality for martingales to show that averaging over $\mathsf{poly}(k)$ such subgradients yields a suitable dual certificate with high probability. Showing that no entry of y is too negative carries much of the difficulty in the analysis. For this step, we apply a novel analysis of our optimization method, which uses submodularity structure and couples the iterates of FTRL with iterates of an instantiation of the multiplicative weights algorithm. For details see Section 6.1 of the full version.

The above method computes an implicit representation of $\widetilde{O}(n \cdot \mathsf{poly}(k))$ permutations such that the average of the subgradients they induce is a dual certificate of SFM from which either arcs or coordinates in the minimizer can be deduced. However, naively writing down the gradients that average out to the certificate would require $\Omega(n^2)$-queries. Furthermore, deducing an arc for a single coordinate, through the operation of moving a set of elements to the beginning of each permutation (which we often refer to as move-to-front for simplicity), would also naively require $\Omega(n^2)$-queries, which is prohibitively expensive. To overcome this limitation, we provide efficient methods to sample from these permutations and their associated subgradients. More specifically, we design a

[7]We choose stochastic FTRL rather than stochastic mirror descent to facilitate the attainment of with high probability success guarantees.

[8]Throughout this paper, with high probability means that the probability is $1 - n^{-C}$ for some constant $C > 0$.

method which first draws $\widetilde{O}(n \cdot \mathsf{poly}(k))$ samples from the subgradients as a preprocessing step, and then uses the samples to deduce several arcs. The preprocessing step enables an efficient implementation of the move-to-front operations due through an importance sampling technique. Each arc deduced requires $\widetilde{O}(\mathsf{poly}(k))$ additional samples. For more details, see Section 6.3 of the full version.

This summarizes the main ingredients for obtaining our $\widetilde{O}(n \cdot \mathsf{poly}(k) \log(|f|/\epsilon))$-query result. Somewhat surprisingly, a more careful amortized cost analysis reveals that this algorithm is in fact a strongly-polynomial time algorithm that makes $\widetilde{O}(n \cdot \mathsf{poly}(k))$ queries. This stems, partially, from a more fine-grained analysis of the size of subgradients and how many arcs are deduced each time we compute an ϵ-approximate minimizer (and its corresponding dual certificate).

This discussion omits a variety of details which are deferred to the full version. The use of randomness and the loss of parallelism in this sequential algorithm is interesting and we leave it as an open problem to see to what degree a deterministic $\widetilde{O}(\mathsf{poly}(k) \log(1/\epsilon))$-depth and $\widetilde{O}(n \cdot \mathsf{poly}(k) \log(1/\epsilon))$-work weakly-polynomial time algorithm (and a strongly-polynomial time analog) can be achieved.

ACKNOWLEDGEMENTS

We thank Deeparnab Chakrabarty for helpful discussions and we thank the anonymous reviewers for helpful feedback. Andrei Graur was supported in part by the Nakagawa departmental fellowship award from the Management Science and Engineering Department at Stanford University, NSF CAREER Award CCF-1844855, and NSF Grant CCF-1955039. Part of work was done while Haotian Jiang was a Ph.D. student at the Unviersity of Washington and supported by a Packard fellowship. Aaron Sidford was supported in part by a Microsoft Research Faculty Fellowship, NSF CAREER Award CCF-1844855, NSF Grant CCF-1955039, a PayPal research award, and a Sloan Research Fellowship.

REFERENCES

[AKM+21] Kyriakos Axiotis, Adam Karczmarz, Anish Mukherjee, Piotr Sankowski, and Adrian Vladu. Decomposable submodular function minimization via maximum flow. In *International Conference on Machine Learning*, pages 446–456. PMLR, 2021.

[ALS20] Brian Axelrod, Yang P Liu, and Aaron Sidford. Near-optimal approximate discrete and continuous submodular function minimization. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 837–853. SIAM, 2020.

[B+15] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[BS20] Eric Balkanski and Yaron Singer. A lower bound for parallel submodular minimization. In *Proceedings of the 52nd annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 130–139, 2020.

[BV04] Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. *Journal of the ACM (JACM)*, 51(4):540–556, 2004.

[CCK21] Deeparnab Chakrabarty, Yu Chen, and Sanjeev Khanna. A polynomial lower bound on the number of rounds for parallel submodular function minimization and matroid intersection. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2021.

[CGJS22] Deeparnab Chakrabarty, Andrei Graur, Haotian Jiang, and Aaron Sidford. Improved lower bounds for submodular function minimization. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2022.

[CGJS23] Deeparnab Chakrabarty, Andrei Graur, Haotian Jiang, and Aaron Sidford. Parallel submodular function minimization. *arXiv preprint arXiv:2309.04643*, 2023.

[CJK14] Deeparnab Chakrabarty, Prateek Jain, and Pravesh Kothari. Provable submodular minimization using wolfe's algorithm. *Advances in Neural Information Processing Systems*, 27, 2014.

[CJST19] Yair Carmon, Yujia Jin, Aaron Sidford, and Kevin Tian. Variance reduction for matrix games. *Advances in Neural Information Processing Systems*, 32, 2019.

[CLSW17] Deeparnab Chakrabarty, Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. Subquadratic submodular function minimization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1220–1231, 2017.

[Cun85] William H Cunningham. On submodular function minimization. *Combinatorica*, 5(3):185–192, 1985.

[DJL+22] Sally Dong, Haotian Jiang, Yin Tat Lee, Swati Padmanabhan, and Guanghao Ye. Decomposable non-smooth convex optimization with nearly-linear gradient oracle complexity. *Advances in Neural Information Processing Systems*, 2022.

[Don06] David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[DVZ21] Daniel Dadush, László A. Végh, and Giacomo Zambelli. Geometric rescaling algorithms for submodular function minimization. *Mathematics of Operations Research*, 46(3):1081–1108, 2021.

[Edm70] Jack Edmonds. Submodular functions, matroids, and certain polyhedra. *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen*, page 11, 1970.

[EN15] Alina Ene and Huy Nguyen. Random coordinate descent methods for minimizing decomposable submodular functions. In *International Conference on Machine Learning*, pages 787–795. PMLR, 2015.

[ENV17] Alina Ene, Huy Nguyen, and László A Végh. Decomposable submodular function minimization: discrete and continuous. *Advances in Neural Information Processing Systems*, 30, 2017.

[FI03] Lisa Fleischer and Satoru Iwata. A push-relabel framework for submodular function minimization and applications to parametric optimization. *Discrete Applied Mathematics*, 131(2):311–322, 2003.

[GLS81] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[GLS84] Martin Grötschel, László Lovász, and Alexander Schrijver. Geometric methods in combinatorial optimization. In *Progress in combinatorial optimization*, pages 167–183. Elsevier, 1984.

[GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*. Springer, 1988.

[Har08] Nicholas James Alexander Harvey. *Matchings, matroids and submodular functions*. PhD thesis, Massachusetts Institute of Technology, 2008.

[HRRS19] Yassine Hamoudi, Patrick Rebentrost, Ansis Rosmanis, and Miklos Santha. Quantum and classical algorithms for approximate submodular function minimization. *arXiv preprint arXiv:1907.05378*, 2019.

[IFF01] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48(4):761–777, 2001.

[IO09] Satoru Iwata and James B Orlin. A simple combinatorial algorithm for submodular function minimization. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 1230–1237. SIAM, 2009.

[Iwa03] Satoru Iwata. A faster scaling algorithm for minimizing submodular functions. *SIAM Journal on Computing*, 32(4):833–840, 2003.

[Iwa08] Satoru Iwata. Submodular function minimization. *Mathematical Programming*, 112(1):45, 2008.

[JBS13] Stefanie Jegelka, Francis Bach, and Suvrit Sra. Reflection methods for user-friendly submodular optimization. *Advances in Neural Information Processing Systems*, 26, 2013.

[Jia22] Haotian Jiang. Minimizing convex functions with rational minimizers. *ACM Journal of the ACM (JACM)*, 2022.

[JLSW20] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games, and its applications. In *Proceedings of the 52nd annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 944–953, 2020.

[KBP19] Senanayak Sesh Kumar Karri, Francis Bach, and Thomas Pock. Fast decomposable submodular function minimization using constrained total variation. *Advances in Neural Information Processing Systems*, 32, 2019.

[KG14] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability*, 3:71–104, 2014.

[Kha80] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[KTE88] Leonid G Khachiyan, Sergei Pavlovich Tarasov, and I. I. Erlikh. The method of inscribed ellipsoids. In *Soviet Math. Dokl*, volume 37, pages 226–230, 1988.

[LB11] Hui Lin and Jeff Bilmes. Optimal selection of limited vocabulary speech corpora. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[Lev65] Anatoly Yur'evich Levin. An algorithm for minimizing convex functions. In *Doklady Akademii Nauk*, volume 160, pages 1244–1247. Russian Academy of Sciences, 1965.

[LJJ15] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. *Advances in neural information processing systems*, 28, 2015.

[Lov83] László Lovász. Submodular functions and convexity. In *Mathematical programming the state of the art*, pages 235–257. Springer, 1983.

[LSW15] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1049–1065. IEEE, 2015.

[LWR20] Xiaoping Li, Yadi Wang, and Rubén Ruiz. A survey on sparse learning models for feature selection. *IEEE transactions on cybernetics*, 52(3):1642–1660, 2020.

[MAH+12] Farokh Marvasti, Arash Amini, Farzan Haddadi, Mahdi Soltanolkotabi, Babak Hossein Khalaj, Akram Aldroubi, Saeid Sanei, and Janathon Chambers. A unified approach to sparse signal processing. *EURASIP journal on advances in signal processing*, 2012(1):1–45, 2012.

[McC05] S Thomas McCormick. Submodular function minimization. *Discrete Optimization*, 12:321–391, 2005.

[New65] Donald J Newman. Location of the maximum on unimodal surfaces. *Journal of the ACM (JACM)*, 12(3):395–398, 1965.

[NJJ14] Robert Nishihara, Stefanie Jegelka, and Michael I Jordan. On the convergence rate of decomposable submodular function minimization. *Advances in Neural Information Processing Systems*, 27, 2014.

[NN89] YE Nesterov and AS Nemirovskii. Self-concordant functions and polynomial time methods in convex programming. preprint, central economic & mathematical institute, ussr acad. *Sci. Moscow, USSR*, 1989.

[Orl09] James B Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009.

[Sch00] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[Sho77] Naum Z Shor. Cut-off method with space extension in convex programming problems. *Cybernetics and systems analysis*, 13(1):94–96, 1977.

[Vai89] Pravin M Vaidya. A new algorithm for minimizing convex functions over convex sets. In *30th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 338–343, 1989.

[Vyg03] Jens Vygen. A note on Schrijver's submodular function minimization algorithm. *Journal of Combinatorial Theory, Series B*, 88(2):399–402, 2003.

[YN76] David B Yudin and Arkadii S Nemirovski. Evaluation of the information complexity of mathematical programming problems. *Ekonomika i Matematicheskie Metody*, 12:128–142, 1976.