Geometric Rescaling Algorithms for Submodular Function Minimization

Daniel Dadush* dadush@cwi.nl
László A. Végh†‡ l.vegh@lse.ac.uk
Giacomo Zambelli ‡ g.zambelli@lse.ac.uk

Abstract

We present a new class of polynomial-time algorithms for submodular function minimization (SFM), as well as a unified framework to obtain strongly polynomial SFM algorithms. Our new algorithms are based on simple iterative methods for the minimum-norm problem, such as the conditional gradient and the Fujishige-Wolfe algorithms. We exhibit two techniques to turn simple iterative methods into polynomial-time algorithms.

Firstly, we use the geometric rescaling technique, which has recently gained attention in linear programming. We adapt this technique to SFM and obtain a weakly polynomial bound $O((n^4 \cdot \mathrm{EO} + n^5) \log(nL))$.

Secondly, we exhibit a general combinatorial black-box approach to turn any strongly polynomial εL-approximate SFM oracle into a strongly polynomial exact SFM algorithm. This framework can be applied to a wide range of combinatorial and continuous algorithms, including pseudo-polynomial ones. In particular, we can obtain strongly polynomial algorithms by a repeated application of the conditional gradient or of the Fujishige-Wolfe algorithm. Combined with the geometric rescaling technique, the black-box approach provides a $O((n^5 \cdot \mathrm{EO} + n^6) \log^2 n)$ algorithm.

Finally, we show that one of the techniques we develop in the paper, "sliding", can also be combined with the cutting-plane method of Lee, Sidford, and Wong [27], yielding a simplified variant of their $O(n^3 \log^2 n \cdot \mathrm{EO} + n^4 \log^{O(1)} n)$ algorithm.

*Centrum Wiskunde & Informatica. Supported by NWO Veni grant 639.071.510.
†London School of Economics.
‡Supported by EPSRC First Grant EP/M02797X/1.

1 Introduction

Given a finite set V, a function $f : 2^V \to \mathbb{Z}$ is *submodular* if

$$f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y) \quad \forall X, Y \subseteq V. \tag{1}$$

We denote $n := |V|$. Examples include the graph cut function, the coverage function, or the entropy function. Submodularity can be interpreted as a diminishing returns property and is therefore important in economics and game theory. Submodular optimization is widely applied in machine learning and computer vision (see e.g. [1]).

We will assume that the function f is given via an *evaluation oracle*: for every set $S \subseteq V$, we can query the value $f(S)$ in time EO. We will assume throughout that $f(\emptyset) = 0$; this is without loss of generality. In the *submodular function minimization (SFM)* problem, the objective is to find a minimizer of this function:

$$\min_{S \subseteq V} f(S). \tag{SFM}$$

The first polynomial-time – indeed, strongly polynomial – algorithm was given by Grötschel, Lovász, and Schrijver in 1981, using the ellipsoid method [19]. It remained an important goal to find a strongly polynomial combinatorial algorithm, which was resolved in 2000, independently by Schrijver [30], and by Iwata, Fleischer, and Fujishige [23]. The best current running time of a combinatorial algorithm is $O(n^5 \cdot \mathrm{EO} + n^6)$ by Orlin [28]. A recent breakthrough result by Lee, Sidford, and Wong [27] gave an improved variant of the ellipsoid method with running time $O(n^3 \log^2 n \cdot \mathrm{EO} + n^4 \log^{O(1)} n)$.

However, the above algorithms do not appear to work well for large scale instances that arise in applications such as speech recognition or image segmentation. A line of recent work has focused on exploiting special structure of submodular functions that arise in these applications, such as decomposability [13, 12, 26, 32]. But for general functions, simple iterative algorithms appear to outperform the provably polynomial algorithms [18]. In particular, the Fujishige-Wolfe minimum-norm point algorithm [15, 34] appears to be among the best ones in practice [1, 18], despite the fact that the first pseudo-polynomial running time bound was given as recently as 2014, by Chakrabarty et al. [5].

Our contributions This paper presents polynomial-time algorithms based on simple iterative methods such as the conditional gradient algorithm or the Fujishige-Wolfe algorithm. We exhibit two different techniques to improve the performance of these algorithms to polynomially bounded. The first technique uses *geometric rescaling*, whereas the second provides a *unified combinatorial framework for strongly polynomial SFM algorithms*. In what follows, we provide an overview of both techniques.

Geometric rescaling has recently gained attention for linear programming. We use the "Full Support Image Algorithm" from [8]; this was also obtained independently by Hoberg and Rothvoß [21]. This is a general algorithmic technique to turn simple iterative algorithms to polynomial-time algorithms for LP feasibility, by adaptively changing the scalar product. The first such algorithms were given by Betke [3], and by Dunagan and Vempala [10]; we refer the reader to [8] for an overview of the literature. The method is also applicable to conic problems in the oracle model [2, 7, 8, 29].

Geometric rescaling algorithms are inherently for feasibility problems. The immediate application of [8] to (SFM) would only provide the optimum value to (SFM) using binary search[1]. However, doing so would not provide us a primal optimal solution (that is, a minimizer set), nor a dual certificate of optimality (as in Theorem 2.1). We introduce new techniques to obtain both primal and dual optimal solutions. The *sliding technique* is used to obtain a primal optimal solution: we reduce the optimization problem (SFM) to a dynamically changing feasibility problem. In case of infeasibility, the geometric rescaling algorithms terminate when a certain number of iterations is reached, without providing a

[1]Indeed, any polynomial-time algorithm for conic feasibility can be turned into a weakly-polynomial algorithm for (SFM) using binary search. For example, in a recent note Fujishige [17] shows how an algorithm of Chubanov [7] can be used in this framework.

Farkas certificate of infeasibility. The *pull-back technique* enables to identify a dual optimality certificate (and more generally, an approximate dual solution). This technique is also applicable in the general LP setting.

Our geometric rescaling algorithm finds both primal and dual optimal solutions, in running time $O((n^4 \cdot \mathrm{EO} + n^5) \log(nL))$. Here, the complexity parameter L denotes the largest norm of a point in the base polytope. This matches the best weakly polynomial guarantees [22, 24] prior to [27].

Building on the geometric rescaling technique, we also obtain a *strongly polynomial* $O((n^5 \cdot \mathrm{EO} + n^6) \log^2 n)$. This is obtained from a *unified combinatorial framework* that can turn any strongly polynomial εL-approximate SFM-oracle into an exact strongly polynomial algorithm. In fact, pseudo-polynomial $\mathrm{poly}(n, 1/\varepsilon)$ running time suffices. Hence, somewhat surprisingly, we can even use the conditional gradient or the Fujishige-Wolfe algorithm to obtain strongly polynomial running times.

We can also apply this unified framework to the cutting plane method. Using the cutting plane technique by Lee, Sidford, and Wong [27], we obtain a much simpler SFM algorithm than the one described in their paper, with the same running time bound $O(n^3 \log^2 n \cdot \mathrm{EO} + n^4 \log^{O(1)} n)$. Interestingly, our variant based on cutting-planes does not rely on the Lovász extension, as is the case both for Lee, Sidford, and Wong, and for Grötschel, Lovász, and Schrijver. Rather, we apply the cutting plane method to the strict feasibility problem for a suitably defined convex set. This is made possible by the use of the same sliding technique developed for our geometric rescaling algorithm.

The general combinatorial framework is based on maintaining a *ring family* guaranteed to contain all minimizer sets, where the size of the family decreases through the algorithm until a minimizer is found. This technique was introduced by Iwata, Fleischer, and Fujishige [23], and used in multiple subsequent papers, such as Iwata and Orlin [24], and Lee, Sidford, and Wong [27]. We note that this technique traces back to strongly polynomial algorithms for minimum-cost flows, pioneered by Tardos [33]. Our implementation also adopts a simplified variant of the bucketing technique of [27] that leads to a factor n improvement in the running time as compared to the original framework of [23].

An advantage of our unified framework is that, unlike all previous papers where the combinatorial arguments on the ring-family are intertwined with the details of some "basic" algorithm – which can be combinatorial in nature as in [23] and [24] or continuous as in [27] – here we use a *black-box* approach, by explicitly formulating the approximate oracle requirement, and then showing that the "basic" routine fulfills those requirements.

The rest of the paper is structured as follows. Section 2 contains problem definitions and the necessary background, including an overview of the relevant iterative methods. Section 3 presents the weakly polynomial geometric rescaling algorithm to solve SFM. Section 4 presents the general framework for strongly polynomial algorithms. In Section 5, we describe the pull-back technique that enables the implementation of the approximate oracle using our geometric rescaling method. Finally, Section 6 shows how the cutting plane methods can be used in the strongly polynomial framework.

2 Preliminaries

We refer the reader to [31, Sections 44-45] on the basics of submodular optimization; this contains all definitions and basic results presented next. The survey [1] provides an overview of continuous algorithms for submodular function minimization.

For a vector $z \in \mathbb{R}^V$, we denote by $z(v)$ the component of z relative to v, and for a subset $S \subseteq V$ we use the notation $z(S) = \sum_{v \in S} z(v)$. For a number $a \in \mathbb{R}$, we let $a^+ = \max\{0, a\}$ and $a^- = \min\{0, a\}$; hence, $a = a^+ + a^-$. Similarly, given a vector $z \in \mathbb{R}^V$, we denote $z^+ = (z(v)^+)_{v \in V}$ and $z^- = (z(v)^-)_{v \in V}$.

The base polytope and the greedy algorithm The submodular base polytope $B(f)$ of a submodular function f is defined as

$$B(f) := \{x \in \mathbb{R}^V : x(S) \leq f(S)\ \forall S \subseteq V, x(V) = f(V)\}.$$

This polytope is non-empty for every submodular function f. The elements of $B(f)$ are called *bases*, and the vertices are the *extreme bases*. The extreme bases correspond to permutations of the ground set. More precisely, for any ordering $v_1, v_2, \ldots, v_n$ of V, the following point is a vertex of $B(f)$, and every vertex is of this form for some ordering:

$$\begin{aligned} x(v_1) &:= f(\{v_1\}), \\ x(v_i) &:= f(\{v_1, \ldots, v_i\}) - f(\{v_1, \ldots, v_{i-1}\}) \quad \forall i = 2, \ldots, n. \end{aligned} \tag{2}$$

Furthermore, given a weight function $w : V \to \mathbb{R}$, one can compute an extreme base minimizing $w^\top x$ by the greedy algorithm GreedyMin(f, w) as follows: order the vertices in V so that $w(v_1) \leq w(v_2) \leq \ldots \leq w(v_n)$, and output x defined by (2) as the optimal solution. The value of the minimum-cost is then given by

$$\min_{x \in B(f)} w^\top x = \sum_{i=1}^{n-1} f(\{v_1, \ldots, v_i\})(w(v_i) - w(v_{i+1})) + f(V)w(v_n). \tag{3}$$

The subroutine GreedyMin(f, w) requires $O(n \cdot \mathrm{EO} + n \log n)$ arithmetic operations. If w has several entries with the same value, then there are multiple ways to sort the elements of V in ascending value of w, each giving rise to a different optimal extreme base of $B(f)$. The extreme bases corresponding to the possible tie-breakings are the vertices of the face of $B(f)$ minimizing $w^\top x$.

If $v_1, \ldots, v_n$ is the ordering computed by GreedyMin(f, w), we define

$$\textsc{MinSet}(f, w) \stackrel{\text{def}}{=} \operatorname{argmin}\{f(S) : S = \{v_1, \ldots, v_i\} \exists i \in [n]\}. \tag{4}$$

A min-max characterization of (SFM) was given by Edmonds:

Theorem 2.1 (Edmonds [11])**.** *For a submodular function $f : 2^V \to \mathbb{R}$, we have*

$$\max\{x^-(V) : x \in B(f)\} = \min\{f(S) : S \subseteq V\}. \tag{5}$$

We will often use the following simple consequence. Assume that for some $x \in B(f)$, $S \subseteq V$, and $\varepsilon > 0$, we have $f(S) \leq x^-(V) + \varepsilon$. Then $f(S) \leq f(T) + \varepsilon$ for any $T \subseteq V$.

Complexity parameters There are multiple complexity parameters relevant for SFM.

$$L_f \stackrel{\text{def}}{=} \max\{\|z\|_1 : z \in B(f)\}, \quad L_{f,2} \stackrel{\text{def}}{=} \max\{\|z\|_2 : z \in B(f)\}, \quad F_f \stackrel{\text{def}}{=} \max\{|f(S)| : S \subseteq V\}.$$

That is, L_f and $L_{f,2}$ are the maximum 1 and 2-norms of the (extreme) bases of $B(f)$. Clearly, $L_{f,2} \leq L_f \leq \sqrt{n} L_{f,2}$. It is also well-known that $L_f = \Theta(F_f)$ (see e.g. [6, Lemma 5], and also [20, 25]).

Some of our algorithms require the explicit knowledge of a complexity parameter. We can use the following upper-bounds. Let $\alpha(v) = \max\{f(\{v\}), |f(V) - f(V \setminus \{v\})|\}$. Then, for every $z \in B(f)$, $|z(v)| \leq \alpha(v)$ (see [16, Section 3.3]). Hence, we can upper bound $F_f \leq L_f \leq \alpha(V)$ and $L_{f,2} \leq \sum_{v \in V} \alpha(v)^2$. On the other hand, $\alpha(v) \leq F_f$ for all $v \in V$.

To summarize, $\log(nZ)$ is within a constant factor of the same value for any choice of $Z \in \{F_f, L_f, L_{f,2}, \alpha(V), \sum_v \alpha(v)^2\}$. Since our running time bounds will contain such terms, the choice of the specific complexity parameter does not matter.

The minimum-norm point problem Fujishige [15] showed a reduction of (SFM) to the following convex quadratic optimization problem.

Theorem 2.2 (Fujishige [15])**.** *Let z be the unique optimal solution to*

$$\min\left\{\frac{1}{2}\|x\|_2^2 \colon x \in B(f)\right\}. \tag{6}$$

Then, the set $S^* = \{v \in V \colon z(v) < 0\}$ is a minimizer of (SFM)*. Furthermore, $|f(S^*)| \leq \sqrt{n}\|z\|_2$.*

We remark that the set S^* in the above claim is in fact the inclusion-wise minimal minimizer to (SFM) [15]. Note that in case of $f(V) = 0$, Theorems 2.1 and 2.2 imply that the minimizer of the 2-norm also minimizes the 1-norm in $B(f)$. An approximate optimal solution to (6) can be converted to an approximate optimal solution to (5), as stated below (see [5, Theorem 5]).

Theorem 2.3. *Assume that $z \in B(f)$ satisfies that $\|z\|_2^2 \leq z^\top x + \delta^2$ for any $x \in B(f)$. Let $S = \textsc{MinSet}(f, z)$. Then, $f(S) \leq z^-(V) + 2n\delta$. Consequently, $f(S) \leq f(T) + 2n\delta$ for any $T \subseteq V$.*

2.1 Iterative methods for SFM

Convex optimization algorithms can be naturally applied to SFM, either by solving the quadratic formulation (6), or by minimizing the so-called *Lovász-extension*, which we do not discuss here. We refer the reader to [1] for an overview of such algorithms. Here, we briefly outline two important algorithms based on (6).

The conditional gradient algorithm The conditional gradient, or Frank-Wolfe algorithm maintains a point $y \in B(f)$, represented as a convex combination $y = \sum_{i=1}^k \lambda_i g_i$ of extreme bases. It is initialized with $y = g$ for an arbitrary extreme base g. Every iteration runs $\textsc{GreedyMin}(f, y)$ to obtain an extreme base g'. If $y^\top g' \geq \|y\|_2^2$, then y is the minimum-norm point in $B(f)$, and the algorithm terminates. Otherwise, y is replaced by the minimum-norm point y' on the line segment $[y, g']$. The standard convergence analysis of the conditional gradient algorithm, together with Theorem 2.3 provide the following convergence bound (see e.g. [1, Sec 10.8]).

Theorem 2.4. *For any $\delta > 0$, within $O(n/\delta^2)$ iterations of the conditional gradient algorithm, we obtain a $y \in B(f)$ such that for $S = \textsc{MinSet}(f, y)$, we have $f(S) \leq y^-(V) + O(\delta L_{f,2})$. The total running time is $O((n^2 \cdot \mathrm{EO} + n^2 \log n)/\delta^2)$.*

The Fujishige-Wolfe algorithm Wolfe [34] gave a finite algorithm for finding the minimum-norm point in a polytope given by its vertices; his algorithm can also be interpreted as an active set method [1]. Fujishige adapted Wolfe's algorithm to SFM [15, 18]. We now give a brief sketch of the algorithm; for a more detailed description, see [5, 18, 34].

An affinely independent set of points $X \in \mathbb{R}^n$ is called a *corral* if the orthogonal projection of 0 to the affine hull of X is in the relative interior of the convex hull of X. In particular, the optimal solution to the minimum-norm point problem can be obtained by a corral, comprising vertices of the face of a polytope containing the minimum-norm point.

Every *major cycle* of the Fujishige-Wolfe algorithm starts and ends with a corral formed by extreme bases in $B(f)$. The algorithm is initialized with an arbitrary extreme base (note that every singleton set is a corral). Let X be the corral at the beginning of a major cycle, and let y be the projection of 0 to the affine hull of X; this can be obtained by a closed-form formula. Let us run $\textsc{GreedyMin}(f, y)$ to obtain an extreme base g'. If $y^\top g' \geq \|y\|_2^2$, then the algorithm terminates with y as the minimum-norm point in $B(f)$. Otherwise, we consider $X' = X \cup \{g'\}$, which is also affinely independent. We set $\bar{x} = y$, and compute y' as the projection of 0 to the affine hull of X'. If y' is in the relative interior of $\mathrm{conv}(X')$, the major cycle terminates with the new corral X'. Otherwise, we start a *minor cycle*: we replace X' by the extreme points of the face of the $\mathrm{conv}(X')$ that contains the intersection point $[\bar{x}, y'] \cap \mathrm{conv}(X')$; the new $\bar{x}$ is defined to be this intersection point. Minor cycles are repeated until a corral is obtained. Finite convergence is guaranteed since $\|\bar{x}\|_2$ decreases in every major and minor cycle, and the number of corrals is finite. However, a bound on the convergence rate was only recently given in [5].

Theorem 2.5 (Chakrabarty et al. [5])**.** *For any $\delta > 0$, within $O(n^2/\delta^2)$ iterations (major and minor cycles) of Wolfe's algorithm, we obtain a $y \in B(f)$ such that for $S = \textsc{MinSet}(f, y)$, we have $f(S) \leq y^-(V) + O(\delta L_{f,2})$. The total running time is $O((n^3 \cdot \mathrm{EO} + n^5)/\delta^2)$.*

The line-Fujishige-Wolfe algorithm There is a natural way to speed up the convergence of the Fujishige-Wolfe algorithm, by combining it with the conditional gradient step. For the minimum-norm point algorithm, Betke [3, Algorithm 2.8] proposed such a variant; the authors are not aware of this algorithm having been used in the submodular context. The only change compared to the Fujishige-Wolfe algorithm is that at the beginning of every major cycle, $\bar{x}$ is set to be the minimum-norm point on the line segment $[y, g']$ instead of y. This is the same as the optimal line search in the conditional gradient method. Consequently, in every major cycle we make at least as much progress as in the conditional gradient algorithm. It is easy to see that in the Fujishige-Wolfe algorithm the total number of iterations is at most twice the total number of major cycles. The iteration bound in Theorem 2.5 can be improved to $O(n/\delta^2)$, and the total running time to $O((n^2 \cdot \mathrm{EO} + n^4)/\delta^2)$.

3 Weakly polynomial algorithm via rescaling

The geometric rescaling algorithm The Full Support Image Algorithm in [8, Section 3.2] is applicable to the following oracle setting. Let $\Sigma \subseteq \mathbb{R}^n$ be non-empty, full dimensional cone; our aim is to find a feasible point in the interior. We are given a separation oracle for $\mathrm{int}(\Sigma)$; that is, for any vector w, the oracle decides whether $w \in \mathrm{int}(\Sigma)$, and if not, it returns a vector z such that $z^\top w \leq 0$ but $z^\top y > 0$ for all $y \in \mathrm{int}(\Sigma)$. Then the algorithm finds a point in $\mathrm{int}(\Sigma)$ in $O(n^3 \log \hat{\omega}^{-1})$ calls to the separation oracle, where $\hat{\omega}$ is a condition number which we will define in Section 3.3. The parameter $\hat{\omega}$ can be lower bounded by the *width of the cone* Σ, defined as the radius of the largest ball contained in Σ and centered on the surface of the unit sphere.

Consider now a submodular function f with $f(V) = 0$. Assume we want to decide whether $f(S) \geq 0$ for all $S \subseteq V$, that is, if $S = \emptyset$ is an optimal solution to (SFM). This is equivalent to $0 \in B(f)$ (note that $f(V) = 0$ is needed for this equivalence). Consider now the cone

$$\Sigma = \{w \in \mathbb{R}^n \,:\, w^\top y \geq 0 \quad \forall y \in B(f)\}.$$

This cone has a non-empty interior whenever (6) is different from 0, or equivalently, if $0 \notin B(f)$. A separation oracle for Σ is provided by GreedyMin(f, w). Consequently, if the algorithm does not terminate in the required running time bound, we can conclude that $f(S) \geq 0$ for all $S \subseteq V$. We can use this algorithm in a binary search framework to solve (SFM). When querying $\min_{S \subseteq V} f(S) \geq -\mu$ for a $\mu > 0$, we shift $f(S)$ by $f(S) + \mu$ for every $S \subsetneq V$, $S \neq \emptyset$.

The main drawback of this algorithm is that it only provides the optimum value, but does not give either an optimal set S, nor a dual certificate as in Theorem 2.1. Also, the binary search leads to an additional factor $\log F_f$ in the running time.

In this section, we describe a variant of this algorithm, which provides a primal optimal solution, and does not require binary search. This will be achieved by dynamically shifting or "sliding" the function f throughout the algorithm, as explained below. However, the algorithm does not directly return a dual certificate of optimality. This can be obtained using the pull-back technique introduced in Section 5; see also the remark after Theorem 4.1.

We start by describing the sliding framework. Besides the geometric rescaling algorithm described next, this technique will also be useful for devising simple cutting plane algorithms for SFM in Section 6.

Sliding the function Throughout the algorithm, we maintain a value $\mu \in \mathbb{Z}_+$, along with a set W, such that $f(W) = -\mu$. We initialize $\mu = \max\{0, -f(V)\}$, and set $W = \emptyset$ or $W = V$ accordingly. Hence $-\mu$ gives an upper bound on $\min_{S \subseteq V} f(S)$. The algorithm terminates once it concludes that $f(W) = \min_{S \subseteq V} f(S)$ for the current W. We define the function $f_\mu : 2^V \to \mathbb{Z}$ as

$$f_\mu(S) \stackrel{\mathrm{def}}{=} \begin{cases} 0, & \text{if } S = \emptyset \text{ or } S = V, \\ f(S) + \mu, & \text{otherwise.} \end{cases} \tag{7}$$

This operation is known as the μ-enlargement of the function f (see Fujishige [16, Section 3.1(d)]).

Lemma 3.1. *For a submodular function f and a value $\mu \geq \max\{0, -f(V)\}$, the function f_μ is submodular. If $0 \in B(f_\mu)$, then $-\mu \leq f(S)$ for every $S \subseteq V$. Furthermore, $B(f_\mu) \subseteq B(f_{\mu'})$ whenever $\mu \leq \mu'$.*

Proof. The function $f'(S) = f(S) + \mu$ is clearly submodular. We obtain f_μ from f' by decreasing the value of $f'(\emptyset)$ and $f'(V)$; note that the bound on μ guarantees that these are both nonnegative. This maintains submodularity, since for any choice of X and Y, the RHS in (1) decreases by at least as much as the LHS when replacing f' by f_μ. If $0 \in B(f_\mu)$, then $0 \leq f_\mu(S)$ for any $S \subseteq V$. If $S \notin \{\emptyset, V\}$, then this gives $f(S) \geq -\mu$; the choice of μ guarantees the same for $S = \emptyset$ and $S = V$. For $\mu' \geq \mu$, the containment $B(f_\mu) \subseteq B(f_{\mu'})$ follows, since the constraints $x(S) \leq f_{\mu'}(S)$ are implied by the constraints $x(S) \leq f_\mu(S)$. □

The following Lemma will be used to update the value of μ.

Lemma 3.2. *Consider a value $\mu \geq \max\{0, -f(V)\}$, and let $w : V \to \mathbb{R}$ be a cost function such that $\min\{w^\top x \colon x \in B(f_\mu)\} > 0$. For $S = \textsc{MinSet}(f_\mu, w)$,we have $f(S) < -\mu$.*

Proof. Let $v_1, \ldots, v_n$ be the ordering of V returned by $\textsc{GreedyMin}(f, w)$. Recall that $w(v_1) \leq w(v_2) \leq \ldots \leq w(v_n)$. From (3) we see that the maximum value of $w^\top x$ over $B(f_\mu)$ can be written as

$$w^\top x = \sum_{i=1}^{n-1} (f(\{v_1, \ldots, v_i\}) + \mu)(w(v_i) - w(v_{i+1})).$$

Since $w^\top x > 0$ and $w(v_i) - w(v_{i+1}) \leq 0$ for $i = 1, \ldots, n-1$, it follows that $f(\{v_1, \ldots, v_i\}) < -\mu$ for some value of i, implying the claim. □

Lemma 3.3. *Consider a value $\mu \geq \max\{0, -f(V)\}$ such that $\mu = -f(W)$ for some $W \subseteq V$. Then, $L_{f_\mu} \leq 4L_f$.*

Proof. For any permutation of the ground set, let g and g' be the corresponding extreme bases in $B(f)$ and in $B(f_\mu)$, respectively. These only differ in the first and last elements: respectively by $+\mu$, and by $-\mu - f(V)$. Hence, $\|g'\|_1 \leq \|g\|_1 + 2\mu + |f(V)|$. Note that $\mu \leq L_f$; this is because $\mu = -f(W)$ for a certain set W, and therefore any permutation that starts with the elements of W will give an extreme base of 1-norm at least $|f(W)|$. Also, $|f(V)| \leq L_f$. The claim follows. □

3.1 The sliding von Neumann algorithm

The Full Support Image Algorithm uses the von Neumann algorithm as the basic subroutine. The von Neumann algorithm was described in [9] to find a feasible solution to the system $A^\top y > 0$ for a matrix $A \in \mathbb{R}^{n \times p}$. It can be seen as a variant of the conditional gradient algorithm for minimizing $\frac{1}{2}\|y\|^2$ over $y = Ax$, $\sum x_i = 1$, $x \geq 0$. The main difference between the conditional gradient and the von Neumann algorithm is that the latter one only needs to decide whether the optimum value is positive. As a consequence, it does not require a minimization oracle for $y^\top z$ over the convex hull of the columns. Instead, one only needs to decide whether this minimum is positive, and if not, find a column a_i such that $y^\top a_i < 0$. This will be important for SFM, since we have to run the von Neumann algorithm not over $B(f)$, but over the convex hull of the normalized extreme bases (with respect to a certain norm).

When applied to SFM, the von Neumann algorithm would only be able to decide whether $0 \in B(f)$, or equivalently, if $f(S) \geq 0$ (assuming $f(V) = 0$). Our *sliding von Neumann algorithm* (Algorithm 1) works directly for SFM, using the adaptive shifting f_μ described above. At any point when the algorithm would conclude $0 \notin B(f_\mu)$, the value of μ is increased, and the algorithm continues with the modified problem. This technique is analogous to the *sliding objective function method* when applying the ellipsoid algorithm to optimization problems, see e.g. [4]. However, we do not change a single constraint (corresponding to the objective function), but modify almost every constraint in the feasible region $B(f)$.

The key feature of geometric rescaling algorithms is that the scalar product changes from the standard Euclidean one. The input includes a positive semidefinite matrix $Q \in \mathbb{R}^{n \times n}$,

Algorithm 1 The sliding von Neumann algorithm

Input: A submodular function $f : 2^V \to \mathbb{Z}$, a value $\mu \geq \max\{0, -f(V)\}$, a set $W \subseteq V$ with $f(W) = -\mu$, a positive definite matrix $Q \in \mathbb{R}^{n \times n}$, and an $\varepsilon > 0$.

Output:

- A value $\mu' \geq \mu$ and a set $W' \subseteq V$ with $f(W') = -\mu'$,
- bases $g_1, \ldots, g_k \in B(f_{\mu'})$, $x \in \mathbb{R}^k$, $y \in \mathbb{R}^n$ such that $y = \sum_{i=1}^k x_i g_i / \|g_i\|_Q$, $\vec{e}^\top x = 1$, $x \geq 0$, and $\|y\|_Q \leq \varepsilon$.

1: Pick g_1 as an arbitrary vertex of $B(f_\mu)$. Set $x_1 := 1$, $y := g_1/\|g_1\|_Q$.
2: Let $k := 2$.
3: **while** $\|y\|_Q > \varepsilon$ **do**
4: Let $g_k \leftarrow \textsc{GreedyMin}(f_\mu, Qy)$.
5: **if** $y^\top Q g_k > 0$ **then** ▷ sliding
6: $W :=\textsc{MinSet}(f_\mu, Qy)$; $\delta := -f_\mu(W)$; $\mu := -f(W)$;
7: Set v_1 and v_n to be the first and last elements of V in increasing order by the weight vector Qy.
8: $g_k(v_1) := g_k(v_1) + \delta$; $g_k(v_n) := g_k(v_1) - \delta$.
9: **end if**
10:

$$\lambda := \frac{\left\langle y - \frac{g_k}{\|g_k\|_Q}, y \right\rangle_Q}{\left\| y - \frac{g_k}{\|g_k\|_Q} \right\|_Q^2};$$

11: $y := (1-\lambda)y + \lambda g_k/\|g_k\|_Q$; ▷ min Q-norm point on $[y, g_k/\|g_k\|_Q]$
12: $x_k := \lambda$;
13: **for** $i = 1, \ldots, k-1$ **do** $x_i := (1-\lambda)x_i$
14: $k := k + 1$
return μ, W, the vectors $g_1, \ldots, g_k$, x, and y.

and we use the scalar product $\langle x, y \rangle_Q \stackrel{\text{def}}{=} x^\top Q y$; this induces the norm $\|x\|_Q \stackrel{\text{def}}{=} \langle x, x \rangle_Q$. The overall algorithm in Section 3.2 runs the sliding von Neumann algorithm several times, each time with a different scalar product Q.

Let us now give an overview of Algorithm 1. We initialize the parameter $\mu = \max\{0, -f(V)\}$, and work with f_μ; μ may increase during the algorithm. We maintain a vector y, which is a convex combination of vectors in $B(f_\mu)$, divided by their Q-norms. At every iteration, we call $\textsc{GreedyMin}(f_\mu, Qy)$ to obtain an extreme base $g_k \in B(f_\mu)$ minimizing $y^\top Q x$ over $B(f_\mu)$. If $y^\top Q g_k \leq 0$, then we update y to the minimum Q-norm point on the line segment $\left[y, \frac{g_k}{\|g_k\|_Q}\right]$ (which is given by the choice of λ in line 10).

Consider now the case $y^\top Q g_k > 0$. This means that Qy is the normal vector of a hyperplane separating $B(f_\mu)$ from 0. In particular, this implies that $\min_{S \subseteq V} f_\mu(S) < 0$, that is, $\min_{S \subseteq V} f(S) < -\mu$. In this case, we "slide" the function, by updating μ to a larger value as follows. We update $W := \textsc{MinSet}(f_\mu, Qy)$ and $\mu := -f(W)$. Lemma 3.2 guarantees that this strictly increases the value of μ by a positive δ. We change g_k to represent the output of $\textsc{GreedyMin}(f_\mu, Qy)$ for the new value of μ; this can be obtained by changing the first and last components of g_k in the decreasing order of the elements v of V with respect to the weight function Qy.

Lemma 3.4. *Algorithm 1 terminates in $\lceil 1/\varepsilon^2 \rceil$ iterations. At any point of the algorithm $y/\gamma \in B(f)$, where $\gamma = \sum_{i=1}^k \lambda_i / \|g_i\|_Q$.*

Proof. According to Lemma 3.2, whenever we change μ, we set a larger value, and $y^\top Q g_k \leq 0$

after the change. According to Lemma 3.1, the polytope $B(f_\mu)$ becomes larger at this change; hence all previous g_i's will bases in $B(f_\mu)$, although they are not extreme bases (vertices) anymore. This implies the second claim. The iteration bound follows by the standard argument for von Neumann's algorithm [9]: $1/\|y\|_Q^2$ increases by at least 1 at every update. □

Similarly to Algorithm 1, one can adapt the Fujishige-Wolfe or the line-Fujishige-Wolfe algorithm to this setting, that is, with sliding the value μ, and using GreedyMin(f_μ, Qy) instead of GreedyMin(f, y).

3.2 Geometric rescaling algorithm for SFM

Algorithm 2 Rescaling-SFM

Input: A submodular function $f : 2^V \to \mathbb{Z}$.
Output: A set W minimizing $f(W)$.

1: Set $Q := I_n$, $R := I_n$.
2: Set $\mu := \max\{0, -f(V)\}$.
3: **if** $f(V) < 0$ **then** $W := V$, **else** $W := \emptyset$.
4: **for** $i = 1, \dots, T$ **do**
5: Call Sliding von Neumann$(f, \mu, W, Q, \varepsilon)$ to obtain the new values of μ and W, and vectors $g_1, \dots, g_k, x, y$.
6: **If** $y = 0$, then **stop**; **return** W
7: **rescale**

$$R := \frac{1}{(1+\varepsilon)^2}\left(R + \sum_{i=1}^{k} \frac{x_i}{\|g_i\|_Q^2} g_i g_i^\top\right); \quad Q := R^{-1}. \tag{8}$$

return W.

Algorithm Rescaling-SFM is shown in Algorithm 2. It is the adaptation of the Full Support Image Algorithm to our submodular setting, using the sliding von Neumann algorithm. We need to modify the algorithm and its analysis to reflect that the feasible region keeps changing due to the updates to the value of μ. We use the parameters

$$\varepsilon \stackrel{\text{def}}{=} \frac{1}{20n}, \qquad T \stackrel{\text{def}}{=} 5n \log(nL_{f,2}).$$

The value μ keeps increasing during the algorithm; it is updated within the sliding von Neumann subroutine. We also maintain a set W with $f(W) = -\mu$. The algorithm stops after T rescalings. At this point, we conclude from a volumetric argument that the current W is the minimizer of f. We show the following running time bound.

Theorem 3.5. *Algorithm* Rescaling-SFM *finds an optimal solution to* (SFM) *in time* $O((n^4 \cdot \mathrm{EO} + n^5) \log(nL_{f,2}))$.

Note that, the definition of T requires knowing the value of $L_{f,2}$; we can replace it by the bound $\sum_{v \in V} \alpha(v)^2$ as in Section 2. As noted there, this changes the overall running time bound only by a constant factor. We also note that the rescaling formula (8) uses the denominator $(1+\varepsilon)^2$ instead of $1+\varepsilon$ as in [8]. This is needed in the proof of Lemma 5.2 in Section 5. Nevertheless, the analysis in [8] remains valid by choosing, as we did here, ε smaller by a constant factor.

Let us also note that Hoberg and Rothvoß [21, Section 2.1] present an alternative rescaling method, which uses only rank-1 rescaling in an appropriately chosen random direction; the algorithm admits the same complexity bounds. This variant can also be adapted to the SFM setting.

3.3 Analysis

Let us define the ellipsoid

$$E(R) \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n : x^\top R x \leq 1\}.$$

Further, let

$$\Sigma_\mu \stackrel{\text{def}}{=} \{w \in \mathbb{R}^n \colon w^\top x \geq 0 \quad \forall x \in B(f_\mu)\}, \qquad F_\mu \stackrel{\text{def}}{=} \Sigma_\mu \cap \mathbb{B}^n. \tag{9}$$

Σ_μ is the set of normal vectors of hyperplanes that weakly separate 0 from $B(f_\mu)$. A vector in the interior of Σ_μ gives a strong separation, and verifies that $0 \notin B(f_\mu)$. This in turn implies that $f_\mu(S) < 0$ for some set $S \subseteq V$, and thus the minimum value of f is strictly less than the current estimate $-\mu$.

The main ideas of the analysis are showing that *(a)* the ellipsoid $E(R)$ contains the set F_μ at every iteration (Lemma 3.7), and that *(b)* the volume of $E(R)$ keeps decreasing by a constant factor at every rescaling (Lemma 3.9). For an integer valued f, one can lower bound the volume in terms of n and $L_{f,2}$, assuming that F_μ has a nonempty interior. Hence, at termination one can conclude that the interior of F_μ is empty, which implies that $f_\mu \geq 0$, or equivalently, the minimum value of the function is $-\mu$ for the current μ.

The analysis below provides a slightly different argument than the volume analysis, by bounding the Q-norm of the bases used during the algorithm. This will be needed for the "pull-back" argument for finding a dual certificate of optimality in Section 5.

Clearly, GreedyMin(f_μ, w) can be used as a separation oracle for Σ_μ. Further, Lemma 3.2 implies that if $\mu' \geq \mu$, then $\Sigma_{\mu'} \subseteq \Sigma_\mu$ and $F_{\mu'} \subseteq F_\mu$.

As in [8], for a convex set $X \subset \mathbb{R}^n$ and a vector $a \in \mathbb{R}^n$, we define the width

$$\text{width}_X(a) \stackrel{\text{def}}{=} \max\{a^\top z \colon z \in X\}.$$

Further, we define the condition number

$$\hat{\omega}_\mu \stackrel{\text{def}}{=} \min_{x \in B(f_\mu) \setminus \{0\}} \frac{\text{width}_{F_\mu}(x)}{\|x\|_2}.$$

A key estimate for the running time analysis is the following.

Lemma 3.6. *Assume* $\min_{S \subseteq V} f(S) < -\mu \leq \min\{0, f(V)\}$. *Then*

$$\hat{\omega}_\mu \geq \frac{1}{4\sqrt{n} L_{f,2}}.$$

Proof. Lemma 3.3 asserts $L_{f_\mu,2} \leq 4L_{f,2}$, that is, $\|x\|_2 \leq 4L_{f,2}$ for every $x \in B(f_\mu)$. The claim follows by showing

$$\text{width}_{F_\mu}(x) \geq 1/\sqrt{n}. \tag{10}$$

To prove this, we note that the assumption of the lemma implies $0 \notin B(f_\mu)$. Let z denote the minimum norm point in $B(f_\mu)$, and let $\hat{z} = z/\|z\|_2$. Then for every $x \in B(f_\mu)$,

$$\hat{z}^\top x \geq \|z\|_2.$$

By Theorem 2.2, if S is the minimizer of f_μ, then $1 \leq |f_\mu(S)| \leq \sqrt{n}\|z\|_2$. Thus $\hat{z}^\top x \geq 1/\sqrt{n}$. Since $\hat{z} \in F_\mu$, this provides the bound on $\text{width}_{F_\mu}(x)$ for every $x \in B(f_\mu)$. □

We will use the following results from [8].

Lemma 3.7 ([8, Lemma 3.6])**.** *Throughout the algorithm,* $F_\mu \subseteq E(R)$ *holds.*

Proof. The main part of the proof in [8] is showing that, the said property is maintained at every rescaling. A new phenomenon in the submodular setting is that the set F_μ also changes when μ increases in the sliding von Neumann algorithm. But as noted above, F_μ only decreases in these iterations, hence the property is maintained. □

Lemma 3.8 ([8, Lemma 3.7])**.** *Throughout the algorithm,* $\|x\|_Q \geq \hat{\omega}_\mu \|x\|_2$ *must hold for every* $x \in B(f)$.

Proof. This follows by [8, Lemma 2.15], asserting that $\|x\|_Q = \mathrm{width}_{E(R)}(x)$, and from the definition of $\hat{\omega}_\mu$. □

Lemma 3.9 ([8, Lemma 3.8])**.** *The determinant of R increases at least by a factor $16/9$ at every rescaling.*

Lemma 3.10 ([8, Lemma 3.9])**.** *At any stage of the algorithm, there exists a point $g_k \in B(f_\mu)$ used during one of the previous sliding von Neumann iterations with*

$$\|g_k\|_Q \leq \frac{\|g_k\|_2}{\sqrt{\det(R)^{1/n} - 1}}.$$

Proof of Theorem 3.5. The algorithm performs $T = 5n \log(nL_{f,2})$ rescalings. Lemma 3.9 shows that after T rescalings, $\det(R) \geq (16/9)^T$. Then Lemma 3.10 implies that, after T rescalings, there exists a point $g_k \in B(f_\mu)$ with $\|g_k\|_Q < \|g_k\|_2/(4nL_{f,2})$. Now Lemma 3.8 would contradict Lemma 3.6 if the assumption $\min_{S \subseteq V} f(S) < -\mu$ were true. Since the algorithm maintains a set W with $f(W) = -\mu$, we can conclude that $f(W) = -\mu = \min_{S \subseteq V} f(S)$. This shows that the algorithm correctly terminates.

The algorithm calls the sliding von Neumann subroutine $T = O(n \log(nL_{f,2}))$ times; each call takes at most $\lceil 1/\varepsilon^2 \rceil = O(n^2)$ iterations. At the kth iteration of von Neumann, it takes time time $O(n \cdot \mathrm{EO} + n \log n)$ to run GreedyMin and time $O(k)$ to update the coefficients $x_1, \ldots, x_k$. These give a bound of $O(n^3 \cdot \mathrm{EO} + n^4)$ for each sliding von Neumann subroutine.

Further, every rescaling has to compute $O(n^2)$ outer products $g_i g_i^\top$, add their weighted sum to R, and compute $Q = R^{-1}$. The computation is dominated by computing the outer products, which take altogether $O(n^4)$ time. Hence the iterations between two subsequent rescalings take time $O(n^3 \cdot \mathrm{EO} + n^4)$, yielding the claimed complexity bound. □

4 Strongly polynomial algorithms

In this section, we provide a general scheme to convert an approximate SFM algorithm to a strongly polynomial one. We assume that the SFM algorithm is provided via the following oracle.

Oracle Approx-SFM
Input: A submodular function $f : 2^V \to \mathbb{Z}$ and $\delta > 0$.
Output: A set $W \subseteq V$, and a vector $y \in B(f)$ such that

$$f(W) \leq y^-(V) + \delta L_f.$$

Further, assume y is given as a convex combination of bases of $B(f)$.

The set W returned by the oracle is clearly within δL_f from the optimal solution to (SFM). In particular, if $\delta < 1/L_f$, then W is optimal.

Let $\mathrm{AO}(f, \delta)$ denote the running time of the oracle. We assume that the oracle makes at least one call to the greedy algorithm, which implies that $AO(f, \delta)$ is at least $n \cdot \mathrm{EO}$. Various algorithms in the literature provide implementations of the approximation oracle. Among them:

- the conditional gradient method, in time $O((n^2 \cdot \mathrm{EO} + n^2 \log n)\delta^{-2})$ (Theorem 2.4);
- the Fujishige-Wolfe algorithm in $O((n^3 \cdot \mathrm{EO} + n^5)\delta^{-2})$ (Theorem 2.5);
- the Iwata-Orlin weakly polynomial algorithm [24], in time $O((n^4 \cdot \mathrm{EO} + n^5) \log(n\delta^{-1}))$;[2]
- the Sidford-Lee-Wong cutting plane method in $O(n^2 \cdot \mathrm{EO} \log(n\delta^{-1}) + n^3 \log^{O(1)}(n\delta^{-1}))$ (see Section 6).

The following theorem shows how the oracle can be implemented using Rescaling-SFM. This will be proved in Section 5.

[2] This is not explicitly stated in [24], however their analysis shows that, in time $O((n^4 \cdot \mathrm{EO} + n^5) \log(n\delta^{-1}))$, they obtain a set $W \subseteq V$ and a point $x \in B(f)$ such that $x(W) = f(W)$, $x(v) \geq 0$ for all $v \in V \setminus W$, and $\Phi(x) := \sum_{v \in W} (x^+(v))^2 \leq \delta^2 L_f^2/n$. This implies that $f(W) = x^-(W) + x^+(W) \leq x^-(V) + \sqrt{n\Phi(x)} \leq x^-(V) + \delta L_f$.

Theorem 4.1. *Setting $T = O(n \log(n\delta^{-1}))$ in Algorithm 2, from its output one can compute a set $W \subseteq V$ and a point $y \in B(f)$, expressed as a convex combination of $O(n^3 \log(n\delta^{-1}))$ extreme bases of $B(f)$, such that $f(W) \leq y^-(V) + \delta L_f$. The running time is $O((n^4 \cdot \mathrm{EO} + n^5) \log(n\delta^{-1}))$.*

Finding a dual certificate in Rescaling-SFM For an integer valued f, a pair W and y satisfying the requirements of Approx-SFM$(f, 1/L_f)$ are an optimal pair of primal and dual solutions as in Theorem 2.1. Hence the algorithm of Theorem 4.1 for $\delta = 1/L_f$ provides a dual certificate of optimality in time $O((n^4 \cdot \mathrm{EO} + n^5) \log(nL_f))$, the same as the complexity bound as in Theorem 3.5 (using that $L_f \leq \sqrt{n} L_{f,2}$).

Identifying the structure of optimal solutions The following lemma provides a simple way to identify sets that must be contained in every optimal solution.

Lemma 4.2. *Let y and W denote the output of* Approx-SFM(f, δ). *If $y(v) < -\delta L_f$, then v must be contained in every minimizer of f.*

Proof. Let $S \subseteq V \setminus \{v\}$. Then $f(S) \geq y(S) \geq y^-(V \setminus \{v\}) \geq f(W) - y(v) - \delta L_f > f(W)$. This shows that S cannot be an optimal solution to (SFM). □

Once we find such an element v, minimizing f can be reduced to minimizing the contraction $f' : 2^{V \setminus \{v\}} \to \mathbb{Z}$, defined as $f'(S) \stackrel{\text{def}}{=} f(S \cup \{v\}) - f(\{v\})$. Our other main tool to identify structural properties of optimal solutions is the following.

Lemma 4.3. *Let $y \in B(f)$, $U \subseteq V$, and $v \in V \setminus U$. Assume that $y(v) > -y^-(V \setminus U)$. Then any minimizer to* (SFM) *that contains v must contain some element of U.*

Proof. Let $S \subseteq V \setminus U$, $v \in S$. Then $f(\emptyset) = 0 < y(v) + y^-(V \setminus U) \leq f(S)$, hence S cannot be a minimizer. □

4.1 Ring families

A set family $\mathcal{F} \subseteq 2^V$ is called a *ring family*, if $X, Y \in \mathcal{F}$ implies $X \cap Y, X \cup Y \in \mathcal{F}$. The function $f : \mathcal{F} \to \mathbb{Z}$ is a submodular function over the ring family $\mathcal{F}$, if (1) holds for any $X, Y \in \mathcal{F}$. Submodular function minimization over ring families has been well-studied and can be reduced to standard submodular function minimization [31, Chapter 49]. This is the underlying framework of the strongly polynomial SFM algorithm by Iwata, Fleischer, and Fujishige [23], and has been subsequently used in several other algorithms, e.g. in [24, 27]. Starting with the entire ring family $\mathcal{F} = 2^V$, these algorithms make progress by gradually restricting the function to a smaller ring family that must contain all minimizers. Our algorithm follows the same overall scheme.

A compact representation of a ring family can be obtained via a directed graph (V, F) such that $X \in \mathcal{F}$ if and only if $\delta_F^+(X) = 0$, that is, no arc in F leaves X. In what follows, let us assume that F is an acyclic graph. This is without loss of generality, since strongly connected components can be contracted to single vertices; indeed, given the set of elements C defining a strongly connected component of F, any minimizer of (SFM) must either contain C or be disjoint from C.

The acyclic graph $D = (V, F)$ defines a partial order $\preceq_F$. We have $u \preceq_F v$ if there exists a directed path in F from v to u. In other words, $u \preceq_F v$ if and only if u is contained in every $X \in \mathcal{F}$ that contains v. We say that an ordering of the vertices is *consistent* with the graph F, if u is ordered before v whenever $u \preceq_F v$.

The following definitions and results are similar to those in [31, Section 49.3]. For a set $X \subseteq V$, let

$$X^{\downarrow} \stackrel{\text{def}}{=} \{u \in V : \exists v \in X, u \preceq v\}, \quad X^{\uparrow} \stackrel{\text{def}}{=} \{u \in V : \exists v \in X, v \preceq u\}$$
$$x^{\downarrow} \stackrel{\text{def}}{=} \{x\}^{\downarrow}, \quad x^{\uparrow} \stackrel{\text{def}}{=} \{x\}^{\uparrow}.$$

Thus, $X^{\downarrow}$ is the unique minimal element of $\mathcal{F}$ containing X. We define

$$\ell(v) \stackrel{\text{def}}{=} f((V \setminus v^{\uparrow}) \cup \{v\}) - f(V \setminus v^{\uparrow}).$$

Let us define $f^{\downarrow} : 2^V \to \mathbb{Z}$ by

$$f^{\downarrow}(X) \stackrel{\text{def}}{=} f(X^{\downarrow}) - \ell^-(X^{\downarrow} \setminus X).$$

Lemma 4.4. *The function $f^{\downarrow}$ is submodular on 2^V with $f^{\downarrow}(S) \geq f(S^{\downarrow})$ for all $S \subseteq V$ and $f^{\downarrow}(S) = f(S)$ for every $S \in \mathcal{F}$. Consequently, minimizing f on the ring family $\mathcal{F}$ is equivalent to minimizing $f^{\downarrow}$ on 2^V. The complexity of* GreedyMin$(f^{\downarrow}, w)$ *can be bounded by $O(n \cdot \mathrm{EO} + n^2)$, where* EO *is the complexity to evaluating f.*

Proof. We need the following.

Claim 4.5. *For every $X, Y \in \mathcal{F}$ with $X \subseteq Y$, we have $\ell(Y \setminus X) + f(X) \leq f(Y)$.*

Proof. Let us take the elements of $Y \setminus X$ in a consistent order with $\preceq$ as $z_1, \ldots, z_r$. Then, $Z_i = X \cup \{z_1, \ldots, z_i\} \in \mathcal{F}$ for each $i \in [r]$, and $Z_i \subseteq (V \setminus z_i^{\uparrow}) \cup \{z_i\}$. Submodularity implies $f(Z_i) - f(Z_{i-1}) \geq \ell(z_i)$. The claim follows by adding up all these inequalities. □

To prove that $f^{\downarrow}$ is submodular, we show that $f^{\downarrow} = b$ where $b : 2^V \to \mathbb{Z}$ is defined by

$$b(X) \stackrel{\text{def}}{=} \min\{f(Y) - \ell^-(Y \setminus X) \,:\, X \subseteq Y, Y \in \mathcal{F}\}.$$

The submodularity of b follows by [14, Theorem 14.3.4A], using that $\ell^-(Y) \leq f(Y)$ for every $Y \in \mathcal{F}$ by Claim 4.5. Let us show that $b = f^{\downarrow}$. By definition, $b(X) \leq f^{\downarrow}(X)$ for any $X \subseteq V$. Consider now $Y \in \mathcal{F}$ such that $Y \supsetneq X^{\downarrow}$. Again by Claim 4.5, we have $f(Y) \geq f(X^{\downarrow}) + \ell^-(Y \setminus X^{\downarrow})$; this implies that $X^{\downarrow}$ is the minimizer in the definition of b, therefore $f = b$.

The nonpositivity of ℓ^- gives that $f^{\downarrow}(S) \geq f(S^{\downarrow})$ for all $S \subsetneq V$; it is clear that $f^{\downarrow}(S) = f(S)$ for all $S \in \mathcal{F}$, $S \neq V$. Regarding the complexity of GreedyMin, one needs to compute the values of $f^{\downarrow}(\{v_1, \ldots, v_i\})$ for every $i \in [n]$ for a given order of the vertices; thus, we need to find the sets $S_i = \{v_1, \ldots, v_i\}^{\downarrow}$. We can assume that F is maintained as a transitive graph. When moving from i to $i+1$, we need to compute $S_{i+1} = S_i \cup v_i^{\downarrow}$, which can be done in $O(n)$ time. Adding the ℓ^- values also take $O(n)$ time for each set. Hence, we obtain an overhead $O(n^2)$ over the $O(n \cdot \mathrm{EO})$ oracle queries and $O(n \log n)$ time for sorting the ground set.

□

Claim 4.6. *For every $v \in V$, $\ell(v) = f^{\downarrow}(V) - f^{\downarrow}(V \setminus \{v\})$. In particular, $y(v) \geq \ell(v)$ for every $y \in B(f^{\downarrow})$.*

Proof. If $V \setminus \{v\} \notin \mathcal{F}$, then $V \setminus \{v\}^{\downarrow} = V$, therefore $f^{\downarrow}(V) - f^{\downarrow}(V \setminus \{v\}) = f(V) - f(V) + \ell(v)$ by definition of $f^{\downarrow}$. If $V \setminus \{v\} \in \mathcal{F}$, then $v^{\uparrow} = \{v\}$, therefore $f^{\downarrow}(V) - f^{\downarrow}(V \setminus \{v\}) = f(V) - f(V \setminus \{v\}) = f((V \setminus v^{\uparrow}) \cup \{v\}) - f(V \setminus v^{\uparrow}) = \ell(v)$. For the last part, note that for any extreme base g of $B(f^{\downarrow})$, $g(v) = f^{\downarrow}(S) - f^{\downarrow}(S \setminus \{v\})$ for some $S \subseteq V$ containing v, and by submodularity $f^{\downarrow}(S) - f^{\downarrow}(S \setminus \{v\}) \geq f^{\downarrow}(V) - f^{\downarrow}(V \setminus \{v\}) = \ell(v)$. □

We will use the following bound on the complexity parameter of $f^{\downarrow}$.

Claim 4.7. *Assuming that $f(V) \leq 0$, we have $|\ell^-(V)|/|V| \leq L_{f^{\downarrow}} \leq 2|\ell^-(V)|$.*

Proof. By Claim 4.6 $y(v) \geq \ell(v)$ for all $v \in V$ and for every $y \in B(f^{\downarrow})$, therefore $y^-(V) \geq \ell^-(V)$. This shows that $\|y\|_1 = f^{\downarrow}(V) - 2y^-(V) \leq -2\ell^-(V)$, using also that $f^{\downarrow}(V) \leq 0$.

For the lower bound, let us choose $v \in V$ with lowest value of $\ell^-(v)$. Thus, $|\ell^-(v)| \geq |\ell^-(V)|/|V|$. Consider any extreme base g of $B(f^{\downarrow})$ from an order where v comes last. Then by the first part of Claim 4.6 $g(v) = \ell(v)$, hence $L_{f^{\downarrow}} \geq \|g\|_1 \geq |g(v)| = |\ell(v)| \geq |\ell^-(v)| \geq |\ell^-(V)|/|V|$. □

Algorithm 3 The basic strongly polynomial algorithm

Input: A submodular function $f : 2^V \to \mathbb{Z}$ with $f(V) \leq 0$, and $\delta > 0$.
Output: An optimal solution to (SFM)

1: Initialize $F := \emptyset$, $T := \emptyset$.
2: **while** $\ell^-(V) < 0$ **do**
3: Call Approx-SFM$(f^\downarrow, \delta)$ to obtain W and $y \in B(f^\downarrow)$, represented as a convex combination $y = \sum_{i=1}^k x_i g_i$.
4: **for** $z \in V$ such that $f(V \setminus z^\uparrow) > -|V| \cdot y^-(V \setminus z^\uparrow)$ **do**
5: Compute $y' = \sum_{i=1}^k x_i g_i'$ by bringing all elements of $z^\uparrow$ backward in the order defining g_i.
6: **for** $v \in V \setminus z^\uparrow$ such that $y'(v) > -y'^-(V \setminus z^\uparrow)$ **do**
7: add arc (v, z) to F.
8: **for** $v \in V$ such that $(y(v) < 2\ell^-(V)\delta)$ **do** ▷ contraction
9: Replace f by $f(S \cup v^\downarrow) - f(v^\downarrow)$ on the ground set $V := V \setminus v^\downarrow$.
10: Set $f(V) := \min\{0, f(V)\}$.
11: Set $T := T \cup v^\downarrow$.
12: Contract all strongly connected components of F to single nodes.
return the pre-image of T in the original ground set.

4.2 The basic strongly polynomial scheme

Algoritm 3 builds a ring family $\mathcal{F}$ represented by a directed graph F with the property that $\mathcal{F}$ contains all optimal solutions to (SFM); thus, minimizing f is equivalent to minimizing the modified function $f^\downarrow$. We formulate the algorithm with a general value of δ, and show that it terminates within n^2 iterations for the choice $\delta = 1/(3n^3)$. In particular, we show the following running time bound. We denote by $\mathrm{AO}^\downarrow(f, \delta)$ the maximum $AO(f^\downarrow, \delta)$, where $f^\downarrow$ ranges over all possible choices of ring families $\mathcal{F}$ containing all optimal solutions to (SFM).

We note that, since as in Lemma 4.4, Greedy-Min$(f^\downarrow, w)$ uses time $O(n^2 \cdot \mathrm{EO} + n^2)$ instead of $O(n^2 \cdot \mathrm{EO} + n \log n)$, thus $\mathrm{AO}^\downarrow(f, \delta)$ is upper bounded by the worst case running time bound on $AO(f, \delta)$ plus $n/\log(n)$ times the worst case bound on the number of calls to the greedy algorithm of $AO(f, \delta)$.

Theorem 4.8. *Using $\delta = 1/(3n^3)$, Algoritm 3 finds the optimal solution to* (SFM) *in time $O(n^2 \mathrm{AO}^\downarrow(f, 1/(3n^3)) + n^3 k \cdot \mathrm{EO} + n^4 k)$, where k is an upper bound on the number of extreme bases in the convex combination returned by* Approx-SFM.

Using the bounds from Theorems 2.4 and 2.5, we obtain $O(n^{10} \cdot \mathrm{EO} + n^{11})$ using the conditional gradient algorithm, and $O(n^{11} \cdot \mathrm{EO} + n^{12})$ using the Fujishige-Wolfe algorithm. Note that, $k = O(n/\delta^2) = O(n^7)$ for conditional gradient, whereas $k = O(n)$ for Fujishige-Wolfe. While these running times are high polynomials, we emphasize that they can be obtained by repeated applications of simple iterative methods, without using any form of scaling.

Theorem 4.1 gives a running time $O((n^6 \cdot \mathrm{EO} + n^7) \log n)$ using the Rescaling-SFM algorithm. In Section 4.3, we give an enhanced version of the algorithm with running time $O((n^5 \cdot \mathrm{EO} + n^6) \log^2 n)$.

Let us now give an overview of Algoritm 3. Each main iteration calls the oracle Approx-SFM$(f^\downarrow, \delta)$. Two types of contractions are used. All cycles in F can be contracted to single elements, since an optimal solution can contain either all or no element of a cycle (line 12). The other type of contraction (in line 9) reduces the size of the ground set by eliminating elements that must be contained in every optimal solution. The set T represents the set of elements eliminated by contractions. Thus, the submodular function at the current stage will be defined as $f(S \cup T) - f(T)$ for the original input function f, with the possible exception of $f(V)$. Therefore, the complexity of evaluating the current f is still EO. We will use n below to denote the size of the original ground set V.

The other main step of the algorithm is adding new arcs to F. The following lemma shows the validity of these steps and that either of these operations should occur in every iteration.

Lemma 4.9. *Every $v \in V$ contracted in line 9 must be contained in all minimizers of* (SFM)*, and every arc (v, z) added to F in line 7 satisfies the property that every minimizer that contains v must also contain z. If $\delta \leq 1/(3n^3)$, then every iteration either contracts an element or adds a new arc to F.*

Proof. In line 6, Lemma 4.3 implies that every minimizer of $f^{\downarrow}$ that contains v, must also contain some element of $z^{\uparrow}$. By definition, if a minimizer contains an element of $z^{\uparrow}$, then it must contain z. It follows that every minimizer containing v must also contain z, therefore the new arc (v, z) is valid.

Consider a v such that $y(v) < 2\ell^-(V)\delta$ in line 9. Lemma 4.2 and Claim 4.7 imply that v is contained in every minimizer of $f^{\downarrow}$, and so must be also all elements of $v^{\downarrow}$. By induction, we must have that $v^{\downarrow}$ is contained in every minimizer of f.

Finally, we need to show that if $\delta \leq 1/(3n^3)$, then every iteration adds some arc to F in line 6 or contracts some element in line 9. Note that, if the algorithm enters the while loop when $|V| = 1$, say $V = \{v\}$, then $y(v) = f(V) = \ell^-(V) < 2\ell^-(V)\delta$, so the algorithm contracts v in line 9, and subsequently terminates. Assume that $|V| \geq 2$ and that no element is contracted in line 9. Then $y(v) \geq 2\ell^-(V)\delta$ for all $v \in V$, and thus

$$f^{\downarrow}(S) \geq y^-(V) \geq 2|V| \cdot \ell^-(V)\delta \quad \forall S \subseteq V. \tag{11}$$

Since $f^{\downarrow}(S) = f(S)$ for $S \in \mathcal{F}$, and $\mathcal{F}$ contains all minimizers of f, we have that

$$f(S) \geq 2|V| \cdot \ell^-(V)\delta \quad \forall S \subseteq V. \tag{12}$$

Note that, by construction and from the fact that $V \setminus z^{\uparrow} \in \mathcal{F}$, for every $i \in [k]$ we have $g_i'(V \setminus z^{\uparrow}) = f^{\downarrow}(V \setminus z^{\uparrow}) = f(V \setminus z^{\uparrow})$, and $g_i'(u) \geq g_i(u)$ for every $u \in V \setminus z^{\uparrow}$. It follows that $y'(V \setminus z^{\uparrow}) = f(V \setminus z^{\uparrow})$ and $y'(u) \geq y(u)$ for all $u \in V \setminus z^{\uparrow}$.

Assume that $f(V \setminus z^{\uparrow}) > -|V| \cdot y^-(V \setminus z^{\uparrow})$, as in the condition in line 4. It follows that

$$y'(V \setminus z^{\uparrow}) = f(V \setminus z^{\uparrow}) > -|V| \cdot y^-(V \setminus z^{\uparrow}) \geq -|V| \cdot y'^-(V \setminus z^{\uparrow}).$$

This in turn implies the existence of $v \in V \setminus z^{\uparrow}$ such that $y'(v) > -y'^-(V \setminus z^{\uparrow})$ in line 6.

Finally, we show that if (12) holds, then at least one $z \in V$ satisfies

$$f(V \setminus z^{\uparrow}) > |V| \cdot |y^-(V)|, \tag{13}$$

a bound which is slightly stronger than the condition $f(V \setminus z^{\uparrow}) \geq -|V| \cdot y^-(V \setminus z^{\uparrow})$ in line 4. Hence, at least on new arc will be added to F. We choose $z \in V$ such that $\ell(z)$ is the most negative possible. In particular, $\ell(z) \leq \ell^-(V)/|V|$. By (12) we have

$$\frac{\ell^-(V)}{|V|} \geq \ell(z) = f((V \setminus z^{\uparrow}) \cup \{z\}) - f(V \setminus z^{\uparrow}) \geq 2|V| \cdot \ell^-(V)\delta - f(V \setminus z^{\uparrow}). \tag{14}$$

Consequently,

$$f(V \setminus z^{\uparrow}) \geq |V| \cdot |\ell^-(V)| \cdot \left(\frac{1}{|V|^2} - 2\delta \right).$$

From the assumption $\delta \leq 1/(3n^3) \leq 1/(3|V|^3)$ we obtain $1/|V|^2 - 2\delta \geq 2|V|\delta$ since $|V| \geq 2$. Therefore (13) follows since

$$f(V \setminus z^{\uparrow}) \geq 2|V|^2 \cdot |\ell^-(V)|\delta \geq |V| \cdot |y^-(V)|.$$

The final inequality follows using (11). □

Proof of Theorem 4.8. Lemma 4.9 justifies the contraction steps and the addition of new arcs to F, and shows that the number of main iterations is at most n^2. Let us also note that after every contraction, we decrease the value of $f(V)$ if it becomes positive (that is, if $f(V) > f(v^{\uparrow})$ before the contraction of v). This operation clearly maintains submodularity.

It is also safe in the sense that it may not lead to an incorrect output with respect to the original function. Indeed, note that at termination the algorithm returns the current set of T, which are elements that must be contained in every minimizer of the original function. Hence, the algorithm outputs the unique minimal solution to (SFM). On the other hand, if $f(V)$ was ever decreased, then we decrease it to the same value as $f(\emptyset)$. Therefore it can never become the unique minimizer. If the algorithm terminates with the entire ground set V, then it follows that $f(V)$ was never decreased during the algorithm.

Let us now estimate the running time. Besides the calls to Approx-SFM, the running time is dominated by computing the g_i' bases in line 5, which altogether require $O(nk \cdot \mathrm{EO} + n^2k)$ for every iteration, and this is required $O(n^2)$ times. □

4.3 Speeding up the algorithm

The algorithm described in the previous section needs to identify $O(n^2)$ arcs in F. In the worse case, each iteration may only identify a single arc, resulting in $O(n^2)$ calls to Approx-SFM.

On the other hand, if we were able to guarantee that $|\ell^-(z)|$ is within a factor $O(n^b)$ from $|\ell^-(V)|$ for a constant fraction of all $z \in V$ for some constant $b \geq 1$, the analysis in the proof of Lemma 4.9 implies that for $\delta = 1/O(n^{b+2})$ we would guarantee $f(V \setminus z^\uparrow) \geq -ny^-(V \setminus z^\uparrow)$ for all such $z \in V$. Thus, after running Approx-SFM$(f^\downarrow, 1/O(n^{b+2}))$, we could extend F by $\Theta(n)$ new arcs.

If this property held in all iterations, then $O(n)$ calls to Approx-SFM would suffice. However, the the number of $z \in V$ with $|\ell^-(z)|$ value "close" to $|\ell^-(V)|$ can be $o(n)$. To deal with this situation, we apply the *"bucketing"* technique of Lee, Sidford, and Wong [27]. Instead of the entire V, we restrict our function in every iteration to a suitably chosen $\bar{V} \subseteq V$, and run Approx-SFM restricted to this set with $\delta = n^{-O(\log n)}$. We will obtain $\theta(\bar{V})$ new arcs in this iteration. Thus, if Approx-SFM has running time $O((|\bar{V}|^4 \cdot \mathrm{EO} + |\bar{V}|^5) \log^2 n)$, then the amortized cost of extending F by an arc will be $O((n^3 \cdot \mathrm{EO} + n^4) \log^2 n)$.

We note that this improvement is only applicable if $\mathrm{AO}(f, \delta)$ depends logarithmically on $1/\delta$. Since δ can be quasi-polynomial, the conditional gradient or Fujishige-Wolfe methods would not even be polynomial in this framework.

The following lemma adapts the argument in Section 15.4.1 in [27].

Lemma 4.10. *Let $f : 2^V \to \mathbb{Z}$ be a submodular function, $\mathcal{F}$ a ring family containing all minimizers of f, and $f^\downarrow : 2^V \to \mathbb{Z}$ be the corresponding function defined by f and $\mathcal{F}$. Then in $O(n \cdot \mathrm{EO})$ time we can find a nonempty subset $\bar{V} \subseteq V$ and a positive integer $b = O(\log n)$, such that*

- *For every $z \in V \setminus \bar{V}$, we have $\ell(z) > 2\ell^-(V)/(2n)^{4b}$.*
- *There exist at least $|\bar{V}|/2$ distinct $z \in \bar{V}$ such that $\ell(z) \leq 2\ell^-(V)/(2n)^{4b-4}$.*

Proof. Let us define $V^t \stackrel{\mathrm{def}}{=} \{z \in V : \ell(z) \leq 2\ell^-(V)/(2n)^{4t}\}$ for $t = 1, 2, \ldots$. Clearly, $V^1 \neq \emptyset$, as it contains z with the smallest $\ell(z)$ value. Let b be the smallest value such that $|V^b| \leq 2|V^{b-1}|$. Clearly, $b = O(\log n)$, and choosing $\bar{V} = V^b$ satisfies both requirements. □

For the set $\bar{V}$ and value b as in the lemma, let $\bar{f} : 2^{\bar{V}} \to \mathbb{Z}$ denote the restriction of $f^\downarrow$ to the ground set $\bar{V}$, and let us set

$$\bar{\delta} \stackrel{\mathrm{def}}{=} \frac{1}{(2n)^{4b}}, \quad \delta \stackrel{\mathrm{def}}{=} \frac{2n^2+1}{(2n)^{4b}}, \tag{15}$$

Let us call Approx-SFM$(\bar{f}, \bar{\delta})$ to obtain the vector $\bar{y} \in B(\bar{f})$ defined as a convex combination of extreme bases $\bar{g}_1, \ldots, \bar{g}_k \in B(\bar{f})$, and a set $W \subseteq \bar{V}$ such that $\bar{f}(W) \leq \bar{y}^-(\bar{V}) + \bar{\delta} L_{\bar{f}}$.

Let us now extend $\bar{y} \in \mathbb{R}^{\bar{V}}$ to $y \in \mathbb{R}^V$ as follows. For $v \in \bar{V}$, we let $y(v) = \bar{y}(v)$. Then, consider an arbitrary order $v_1, \ldots, v_{n-|\bar{V}|}$ of $V \setminus \bar{V}$, and set $y(v_j) := f^\downarrow(\bar{V} \cup \{v_1, \ldots, v_j\}) - f^\downarrow(\bar{V} \cup \{v_1, \ldots, v_{j-1}\})$. Let us also define $g_1, \ldots, g_k \in \mathbb{R}^V$, by $g_i(v) = \bar{g}_i(v)$ for $v \in \bar{V}$, $g_i(v) = y(v)$ for $v \in V \setminus \bar{V}$ $(i = 1, \ldots, k)$. Note that, by definition, $g_1, \ldots, g_k$ are extreme bases of $B(f^\downarrow)$, and y is a convex combination of $g_1, \ldots, g_k$.

Lemma 4.11. *For the vector y and set W as above, we have that $y \in \mathbb{B}(f^\downarrow)$, and $f^\downarrow(W) \leq y^-(V) + \delta L_{f^\downarrow}$.*

Proof. By definition $f^\downarrow(W) = \bar{f}(W)$ and $L_{\bar{f}} \leq L_{f^\downarrow}$, because $\bar{f}$ is a restriction of $f^\downarrow$. Therefore,

$$f^\downarrow(W) \leq \bar{y}^-(\bar{V}) + \bar{\delta} L_{f^\downarrow}.$$

Claim 4.12. *$y(v) \geq \ell^-(v)$ for every $v \in V \setminus \bar{V}$.*

Proof. If $v = v_j$, then $y(v) = f^\downarrow(\bar{V} \cup \{v_1, \ldots, v_j\}) - f^\downarrow(\bar{V} \cup \{v_1, \ldots, v_{j-1}\}) \geq f^\downarrow(V) - f^\downarrow(V \setminus \{v\})$ by submodularity. Further, $f^\downarrow(V) - f^\downarrow(V \setminus \{v\}) \geq \ell^-(v)$ by Claim 4.6. □

We have $y^-(V) = \bar{y}^-(\bar{V}) + y^-(V \setminus \bar{V})$. By the choice of $\bar{V}$, we have $\ell^-(v) \geq 2\ell^-(V)/(2n)^{4b} = 2\ell^-(V)\bar{\delta}$ for every $v \in V \setminus \bar{V}$. Using the claim above, we get $y^-(V \setminus \bar{V}) \geq 2n\ell^-(V)\bar{\delta}$. Thus, $y^-(V) \geq \bar{y}^-(\bar{V}) + 2n\ell^-(V)\bar{\delta} \geq \bar{y}^-(\bar{V}) - 2n^2\bar{\delta}L_{f^\downarrow}$. Here, the last inequality used the lower bound in Claim 4.7. Consequently,

$$f^\downarrow(W) \leq y^-(V) + (2n^2+1)\bar{\delta}L_{f^\downarrow} = y^-(V) + \delta L_{f^\downarrow}.$$

□

This proof shows that we can implement Approx-SFM(f, δ) by calling Approx-SFM$(\bar{f}, \bar{\delta})$, and adding the remaining $V \setminus \bar{V}$ elements by $O(n)$ value oracle queries, which is time $O(n \cdot \mathrm{EO} + n^2)$ for the function $f^\downarrow$.

Let us modify Algorithm 3 as follows. In every iteration, we compute $\bar{V}$ and b as in Lemma 4.10, and use this modified implementation of Approx-SFM with δ as defined in (15).

Theorem 4.13. *The above described modification of Algorithm 3 finds an optimal solution to* (SFM) *in time $O(n \cdot \mathrm{AO}^\downarrow(f, n^{-O(\log n)}) + n^3 k \cdot \mathrm{EO} + n^4 k)$, where k is an upper bound on the number of extreme bases returned by* Approx-SFM. *Using the implementation with* Rescaling-SFM, *the running time is $O((n^5 \cdot \mathrm{EO} + n^6)\log^2 n)$.*

Proof. They key observation is that, at every call of the approximation oracle, if no nodes are contracted at line 9, then at least $\frac{1}{2}|\bar{V}|$ new arcs are added to F.

This follows by showing that $f(V \setminus z^\uparrow) > -|V| \cdot y^-(V \setminus z^\uparrow)$ holds for at least half of the elements z of $\bar{V}$. Indeed, as in the proof of Lemma 4.9, we can assume that $y(v) \geq 2\ell^-(V)\delta$ for all $v \in V$ and that (12) holds.

By Lemma 4.10 and our choice of $\bar{V}$, half of the elements of $\bar{V}$ satisfy $\ell(z) \leq 2\ell^-(V)/(2n)^{3b-3} = 2\ell^-(V)\delta(2n)^4/(2n^2+1)$. Hence, as in (14), the assumption (12) implies that

$$f(V \setminus z^\uparrow) \geq 2|V| \cdot \ell^-(V)\delta - \ell(z) \geq 2|V|^2 \cdot |\ell^-(V)|\delta \left(\frac{(2n)^4}{|V|^2(2n^2+1)} - \frac{1}{|V|} \right) > |V| \cdot |y^-(V)|,$$

where the last inequality uses that $|y^-(V)| \leq 2|V| \cdot |\ell^-(V)|\delta$ because of the assumption $y(v) \geq 2\ell^-(V)\delta$ for all $v \in V$, and that the expression in the brackets is ≥ 1 for $n \geq 2$.

The running time of Approx-SFM$(\bar{f}, \bar{\delta})$ is AO$(\bar{f}, n^{-O(\log n)})$. Consequently, the amortized cost of an oracle call per new arc is AO$(\bar{f}, n^{-O(\log n)})/|\bar{V}|$. Since AO depends at least linearly on $|\bar{V}|$, this can be upper bounded by AO$(f, n^{-O(\log n)})/|V|$. Hence, the total time of the oracle calls is $O(n \cdot \mathrm{AO}(f, n^{-O(\log n)})$, which is $O((n^5 \cdot \mathrm{EO} + n^6)\log^2 n)$ for Rescaled-SFM. We also have to recompute the convex combinations in line 5. For every new arc, this requires recomputing k extreme bases, in total time $O(n^3 k \cdot \mathrm{EO} + n^4 k)$. □

5 The pull-back technique for Rescaling-SFM

The main purpose of this section is to prove Theorem 4.1, that is implement Approx-SFM using Rescaling-SFM. We will use a *"pull-back"* technique. Recall that in Rescaling-SFM, we keep modifying the matrix Q defining the scalar product. Lemmas 3.9 and 3.10 guarantee that after t rescalings, there we can identify a vector $g \in B(f_\mu)$ that has a small Q-norm for the current Q, and the bound decreases geometrically with t. Our key technical claim, Lemma 5.2, shows a constructive way to identify a vector $v \in B(f_\mu)$ with

$\|v\|_2 \leq \|g\|_Q$. Provided a vector v with small 2-norm (and thus small 1-norm), we can easily satisfy the requirements of the Approx-SFM, using the following lemma.

Lemma 5.1. *Let $\mu \geq \max\{0, -f(V)\}$ and $W \subseteq V$ such that $f(W) = -\mu$. Let $\mu_1, \mu_2, \cdots, \mu_h \in [0, \mu]$, and for $i = 1, \ldots, h$ let g_i be a basis of the base polytope $B(f_{\mu_i})$. Given $v = \sum_{i=1}^{h} \lambda_i g_i$ where $\lambda \geq 0$ and $\sum_{i=1}^{h} \lambda_i = 1$, in time $O(nh)$ we can compute $y \in B(f)$, given as a convex combination of h extreme bases of $B(f)$, such that*

$$f(W) \leq y^-(V) + \frac{\|v\|_1}{2}.$$

Proof. For $i = 1, \ldots, h$, let $\bar{g}_i$ be the basis of $B(f)$ defined by the same ordering which defined the basis g_i of $B(f_{\mu_i})$. Define $y := \sum_{i=1}^{h} \lambda_i \bar{g}_i$.

Observe that, given $i \in [h]$, if $v_1, \ldots, v_n$ is the ordering defining g_i, then $\bar{g}_i(v_1) = g_i(v_1) - \mu_i$, $\bar{g}_i(v_j) = g_i(v_j)$ for $j = 2, \ldots, n-1$, and $\bar{g}_i(v_n) = g_i(v_n) + \mu_i + f(V)$. Thus, computing $\bar{g}_1, \ldots, \bar{g}_h$ requires time $O(h)$ and computing y requires time $O(nh)$. Furthermore, we have that $\|\bar{g}_i\|_1 \leq \|g_i\|_1 + 2\mu + f(V)$. This implies that

$$\|y\|_1 \leq \|v\|_1 + 2\mu + f(V) = \|v\|_1 - 2f(W) + f(V).$$

Since $\|y\|_1 = f(V) - 2y^-(V)$, the above implies

$$f(W) \leq \frac{\|v\|_1 + f(V) - \|y\|_1}{2} = y^-(V) + \frac{\|v\|_1}{2}.$$

□

Our next Lemma enables pulling back a vector with small Q-norm to a vector with no larger 2-norm. This is done gradually, by pulling back at each rescaling of Rescaling-SFM. The columns of the matrix A will be the bases used in the current iteration of the sliding von Neumann algorithm. We also note that this technique is applicable to the general Full Support Image Algorithm in [8], enabling to find approximate solutions as well as dual certificates of infeasibility.

Lemma 5.2. *Let $A \in \mathbb{R}^{n \times p}$, $R \in \mathbb{S}^n_+$, and $Q = R^{-1}$. Let $x \in \mathbb{R}^p_+$ such that $y := \sum_{i=1}^{p} x_i \frac{a_i}{\|a_i\|_Q}$ satisfies $\|y\|_Q \leq \varepsilon$. Define*

$$R' \stackrel{\text{def}}{=} \frac{1}{(1+\varepsilon)^2} \left(R + \sum_{i=1}^{p} \frac{x_i}{\|a_i\|_Q^2} a_i a_i^\mathsf{T} \right), \tag{16}$$

and $Q' \stackrel{\text{def}}{=} (R')^{-1}$. For every $v \in \mathbb{R}^n$, there exists $\mu \in \mathbb{R}^p_+$ such that $\|v + A\mu\|_Q \leq \|v\|_{Q'}$. Moreover, such a vector μ can be computed in time $O(n^2 p)$.

Proof. For the given $v \in \mathbb{R}^n$, we define $u \stackrel{\text{def}}{=} \frac{1}{(1+\varepsilon)^2} RQ'v$ and let

$$\beta \stackrel{\text{def}}{=} \max_{i \in [p]} \frac{\langle a_i, u \rangle_Q}{\|a_i\|_Q}, \quad \mu_i \stackrel{\text{def}}{=} \frac{x_i}{\|a_i\|_Q} \left(\beta - \frac{\langle a_i, u \rangle_Q}{\|a_i\|_Q} \right) \quad \text{for } i \in [p] \tag{17}$$

We will show that the statement is satisfied by the choice of $\mu \in \mathbb{R}^p_+$ defined above. These values can be clearly computed in $O(n^2 p)$ time.

First, we observe that, by substituting the definitions of R' and u, we obtain

$$v = R'QRQ'v = u + \sum_{i=1}^{p} x_i \frac{\langle a_i, u \rangle_Q}{\|a_i\|_Q} \frac{a_i}{\|a_i\|_Q},$$

which, from the definition of μ and β, implies that

$$v + A\mu = u + \beta y. \tag{18}$$

Next, notice that

$$\begin{aligned}\|v\|_{Q'} &= \sqrt{v^\mathsf{T} Q'R'Q'v} = \frac{1}{(1+\varepsilon)}\left((v^\mathsf{T}Q')R(Q'v) + v^\mathsf{T}Q'\left(\sum_{i=1}^{p}\frac{x_i}{\|a_i\|_Q^2}a_i a_i^\mathsf{T}\right)Q'v\right)^{\frac{1}{2}} \\ &\geq \frac{1}{(1+\varepsilon)}\left((v^\mathsf{T}Q'R)Q(RQ'v)\right)^{\frac{1}{2}} = (1+\varepsilon)\|u\|_Q.\end{aligned}$$

From the above and observing that $|\beta| \leq \|u\|_Q$, from the definition of β, we have

$$\|v + A\mu\|_Q \leq \|u\|_Q + |\beta|\|y\|_Q \leq (1+\varepsilon)\|u\|_Q \leq \|v\|_{Q'},$$

where the first inequality follows from (18) and the triangle inequality. □

We are ready to prove Theorem 4.1, showing how Approx-SFM can be implemented using Rescaling-SFM.

Proof of Theorem 4.1. Run algorithm Rescaling-SFM(f), setting the limit on the number of rescalings to a number $T = cn\log(n\delta^{-1})$ for some constant c to be specified later. At the end of the execution, we identified a value μ and a set $W \subseteq V$ such that $f(W) = -\mu$. Let $g_1, \ldots, g_h$ be all the points in $B(f_\mu)$ used in the sliding von Neumann iterations during the execution of the algorithm.

By Lemma 3.10, for an appropriate choice of c, after T rescalings there exists $k \in [h]$ such that

$$\|g_k\|_Q \leq \frac{2\delta}{3\sqrt{n}}\|g_k\|_2.$$

Let $\hat{g}_k = g_k/\|g_k\|_2$. The running time of Rescaling-SFM(f) with the above choice of T is $O((n^4\mathrm{EO} + n^5)\log(n\delta^{-1}))$. Note also that $h \in O(n^3\log(n\delta^{-1}))$, thus finding k requires time $O(n^5\log(n\delta^{-1}))$ to compute the Q-norms of $g_1, \ldots, g_h$.

By applying Lemma 5.2 for T times (considering the rescaling matrices used in the algorithm in reverse order), we can find a vector $\mu \in \mathbb{R}_+^h$ such that $\|\hat{g}_k + \sum_{i=1}^h \mu_i g_i\|_2 \leq \|\hat{g}_k\|_Q$. Recall that each rescaling matrix is defined by at most n^2 vectors among $g_1, \ldots, g_h$, therefore each application of Lemma 5.2 requires time $O(n^4)$ (assuming that the matrices Q and R used at every rescaling are saved in memory so we do not need to recompute them). Thus, overall, the time required to compute μ is $O(n^5\log(n\delta^{-1}))$.

Define $\alpha = 1 + \|g_k\|_2 \cdot \sum_{i=1}^h \mu_i$, and $\lambda \in \mathbb{R}_+^h$ by

$$\lambda_i = \begin{cases} \frac{\|g_k\|_2\mu_i}{\alpha} & i \in [h]\setminus\{k\} \\ \frac{1+\|g_k\|_2\mu_k}{\alpha} & i = k \end{cases}$$

Define $v := \sum_{i=1}^h \lambda_i g_i$. Observe that $\sum_{i=1}^h \lambda_i = 1$, thus $v \in B_{f_\mu}$. Computing v requires time $O(n^4\log(n\delta^{-1}))$, since we need to sum h n-dimensional vectors.

Furthermore,

$$\|v\|_1 \leq \sqrt{n}\|v\|_2 = \sqrt{n}\frac{\|g_k\|_2}{\alpha}\left\|\hat{g}_k + \sum_{i=1}^h \mu_i g_i\right\|_2 \leq \sqrt{n}L_{f_\mu,2}\|\hat{g}_k\|_Q \leq 2\delta L_f,$$

where the last inequality follows from the fact that $L_{f_\mu,2} \leq L_{f_\mu} \leq 3L_f$ by Lemma 3.3. By Lemma 5.1, in time $O(n^4\log(n\delta^{-1}))$ we can compute $y \in B(f)$ satisfying $f(W) \leq y^-(V) + \frac{\|v\|_1}{2} \leq y^-(V) + \delta L_f$. □

Remark 5.3. The bound $O(n^5\log(n\delta^{-1}))$ for computing μ in the above proof was assuming $O(n^2)$ time for computing Q-scalar products $\langle g, u\rangle_Q$. We note that this can be easily improved by a factor n: we can assume that Qg was precomputed and stored during Rescaling-SFM for all bases g used during the sequence of rescalings. Indeed, it was necessary to compute the norms $\|g\|_Q$ in the sliding von Neumann algorithm. Thus, the bound improves to $O(n^4\log(n\delta^{-1}))$; however, this does not change the overall running time estimate.

6 Cutting plane method

The current best cutting plane method for finding a point in a convex set provided by a separation oracle is due to Lee, Sidford, and Wong [27]. If κ is n times the ratio of the radius of an initial ball containing the convex feasible region and the radius of a ball contained inside, then their algorithm finds a feasible point in $O(n \cdot \mathrm{SO} \log \kappa + n^3 \log^{O(1)} \kappa)$, where SO is the complexity of an (exact) separation oracle. In Part III of their paper, they apply this algorithm for submodular function minimization, and obtain the current best running time bound, $O(n^3 \log^2 n \cdot \mathrm{EO} + n^4 \log^{O(1)} n)$ (see [27, Section 15.4]). This is obtained by combining their cutting plane algorithm with an improved version of the combinatorial framework of ring families; one of their important new contributions is the bucketing technique we also use in Section 4.

In this section, we present an alternative way of applying their cutting plane method to SFM. We prove the same running time bound in a substantially simplified way. Firstly, instead of using the Lovász extension as in [19] and in [27], we apply the cutting plane method to find a feasible solution in F_μ, as defined in (9). We use the sliding technique as in Section 3 for the cutting plane algorithm. Secondly, we employ the combinatorial framework in a black-box manner, by implementing Approx-SFM via the Lee-Sidford-Wong algorithm. The combinatorial interpretation of the certificate returned by the cutting plane method turns out to be much easier than in [27].

Weakly polynomial algorithm Let us start by exhibiting a weakly polynomial $O(n^2 \log(nL_{f,2}) \cdot \mathrm{EO} + n^3 \log^{O(1)}(nL_{f,2}))$ algorithm for SFM, which is the same as the running time in [27]. We use a slight modification of the cutting plane algorithm [27, Section 6.4, Algorithm 2].

We start with $\mu = \max\{0, -f(V)\}$, and maintain a set W with $f(W) = -\mu$ throughout. For the current iterate $x^{(k)}$, GreedyMin$(f_\mu, x^{(k)})$ is used as the separation oracle for $\mathrm{int}(F_\mu)$, which returns an extreme base g of $B(f_\mu)$. If $g^\top x^{(k)} > 0$, then $x^{(k)} \in \mathrm{int}(F_\mu)$, thus, $x^{(k)}$ is feasible. In this case, instead of terminating, we modify the value of μ as in the sliding von Neumann algorithm. That is, we set $W = \textsc{MinSet}(f_\mu, x^{(k)})$, and set the new value $\mu' = -f(W)$. From Lemma 3.2, we see that $x^{(k)} \notin \mathrm{int}(F_{\mu'})$. Thus, we can continue with adding a new cutting plane. Note that $F_{\mu'} \subseteq F_\mu$ if $\mu' > \mu$. Hence, all previous separations remain valid. (Again, this is similar to the sliding objective technique, although we are changing all constraints of the polytope simultaneously.) When $-\mu$ is the minimum value of f, L_μ has no points in the interior, therefore we stop when the volume of the current relaxation becomes too small.

In this setting, we have $\mathrm{SO} = n \cdot \mathrm{EO} + n \log n$. For every value of μ, $F_\mu \subseteq \mathbb{B}^n$ by definition, and Lemma 3.6 implies that, as long as $\min_{S \subseteq V} f(S) < -\mu$, F_μ contains a ball of radius $1/(4\sqrt{n} L_{f,2})$. Hence, $\kappa = O(\sqrt{n} L_{f,2})$, giving the desired running time bound.

Let us note that, even using the original ellipsoid method, as in Grötschel, Lovász, and Schrijver [19], one can obtain $O((n^3 \cdot \mathrm{EO} + n^4) \log(nL_{f,2}))$, since the original ellipsoid algorithm finds a feasible point in time $O((n^2 \cdot \mathrm{SO} + n^4) \log \kappa)$ in the oracle model. Interestingly, even such a simple and direct use of the standard ellipsoid method, compared to the usual approach of minimizing the Lovász extension, provides a running time that is a factor n lower than any weakly-polynomial SFM-algorithm known prior to the work of Lee-Sidford-Wong [27].

Strongly polynomial algorithm Let us now show an $O((n^4 \cdot \mathrm{EO} + n^5) \log(n\delta^{-1}))$ implementation of Approx-SFM(f, δ) using the Lee-Sidford-Wong cutting plane method. We use Theorem 31 from [27]. For $K = F_\mu$ (for any value of μ), by definition $F_\mu \subseteq \mathbb{B}^n \subseteq \mathbb{B}^n_\infty(1)$, that is, $R = 1$. Due to the sliding, the algorithm cannot find a feasible solution, and thus it always returns a thin direction as follows.

Theorem 6.1 ([27, Theorem 31])**.** *For any $\varepsilon \in [0, 1]$, in expected time $O(n \log(n/\varepsilon)) \cdot \mathrm{SO} + n^3 \log^{O(1)}(n/\varepsilon))$, the (sliding) cutting plane method returns a value μ, and constraints $a_i^\top x \geq b_i$ for $i \in [h]$, where $h = O(n)$, $\|a_i\|_2 = 1$, which are all valid for F_μ. Each of these constraint is either an original box constraint, that is $x_j \geq -1$ or $-x_j \geq -1$, or an inequality returned by the separation oracle. Let P denote the intersection of these hyperplanes.*

Further, we obtain non-negative numbers $t_1, t_2, t_3, \ldots, t_h$ with $t_1 = 1$, and a point $x^* \in P$, which satisfy the following:

(a) $\|x^*\|_2 \le 3\sqrt{n}$,

(b) $\left\|\sum_{i=1}^h t_i a_i\right\|_2 = O(\sqrt{n}\varepsilon \log(1/\varepsilon))$,

(c) $a_i^\top x^* - b_i \le \varepsilon$,

(d) $\left(\sum_{i=1}^h t_i a_i\right)^\top x^* - \sum_{i=1}^h t_i b_i \le O(\sqrt{n}\varepsilon \log(1/\varepsilon))$.

The output certifies that the region $P \cap \mathbb{B}_\infty^n(1)$ has small width in the direction of a_1. In fact, for $\bar{a} = \sum_{i=1}^h t_i a_i$ and $\bar{b} = \sum_{i=1}^h t_i b_i$, the valid inequality $\bar{a}^\top x \ge \bar{b}$ is "close" to $-a_1^\top x \ge -b_1$. Indeed, from (b) we see that $a_1^\top x + \bar{a}^\top x = O(n\varepsilon \log(1/\varepsilon))$ for every $x \in \mathbb{B}_\infty^n(1)$. Then, (c) and (d) imply that $b_1 + \bar{b} = O(n\varepsilon \log(1/\varepsilon))$.

We show that for an appropriately chosen ε, this can be used to implement Approx-SFM(f, δ).

Lemma 6.2. *For an appropriate ε such that $\delta = \Omega(n^{3/2}\varepsilon \log(1/\varepsilon))$, from the output of the cutting plane method we can obtain W and y as required for* Approx-SFM(f, δ)*, that is, $f(W) \le y^-(V) + \delta L_f$.*

Proof. Let $[h] = I_b \cup I_s$, where I_b is the set of indices i such that $a_i^\top x_i \ge b_i$ is a box constraint, and I_s is the set of indices corresponding to constraints from the separation oracle. Each constraint in I_s is of the form $a_i = g_i/\|g_i\|_2$ and $b_i = 0$, where g_i is an extreme base of $B(f_{\mu_i})$, where $\mu_i \le \mu$ was the value of μ at the time this cutting plane was added. The lemma will easily follow from the next claim.

Claim 6.3. *The index 1 is in I_s, and $\left\|\sum_{i \in I_s} t_i a_i\right\|_2 = O(n\varepsilon \log(1/\varepsilon))$.*

Proof. First, we show that $1 \in I_s$. For a contradiction, assume that $1 \in I_b$, that is, $a_1 = e_j$ or $a_1 = -e_j$ for some $j \in [n]$ and $b_1 = -1$. As noted above, $b_1 + \bar{b} = O(n\varepsilon \log(1/\varepsilon))$; hence, $\bar{b} > 0$ follows (for small enough ε). This is a contradiction, since $b_i = -1$ for all $i \in I_b$, and $b_i = 0$ for all $i \in I_s$.

Thus, $1 \in I_b$, and therefore $b_1 = 0$. Thus, $\bar{b} = O(n\varepsilon \log(1/\varepsilon))$. Again, this implies that $\sum_{i \in I_b} t_i = O(n\varepsilon \log(1/\varepsilon))$. Together with $\left\|\sum_{i \in I_b} t_i a_i + \sum_{i \in I_s} t_i a_i\right\|_2 = O(\sqrt{n}\varepsilon \log(1/\varepsilon))$ from (b), we get that $\left\|\sum_{i \in I_s} t_i a_i\right\|_2 = O(n\varepsilon \log(1/\varepsilon))$, as required. □

Let $v = \left(\sum_{i \in I_s} \frac{t_i}{\|g_i\|_2} g_i\right) / \left(\sum_{i \in I_s} \frac{t_i}{\|g_i\|_2}\right)$. Since $1 \in I_s$, we have we $\sum_{i \in I_s} \frac{t_i}{\|g_i\|_2} \ge \frac{1}{L_{f,2}} \ge \frac{1}{L_f}$. Hence, it follows that

$$\|v\|_1 \le \sqrt{n}\|v\|_2 \le L_f \sqrt{n} \left\|\sum_{i \in I_s} t_i a_i\right\|_2 = O(L_f n^{3/2} \varepsilon \log(1/\varepsilon)) \le 2\delta L_f.$$

Then, Lemma 5.1 is applicable to provide the certificate for Approx-SFM(f, δ). Note that the set W with $f(W) = -\mu$ has been maintained during the cutting plane algorithm. □

Combining with Theorem 4.13, the total complexity of the oracle calls is $O((n^3 \cdot \mathrm{EO} + n^4) \log(nL_{f,2}))$. However, the total time for recomputing the extreme bases as in line 5 in Algorithm 3 would consume time $O(n^4 \cdot EO + n^5)$, since $k = O(n)$. To decrease this term by a factor n, we can adapt the same trick as in the proof of Lemma 79 in [27]. At the expense of selecting δ to be smaller by a factor n, it suffices to recompute only one of the g_i's, instead of the entire combination.

Comparison to the Lee-Sidford-Wong SFM algorithm Let us now compare our approach to the SFM algorithm described in [27, Part III]. We employ the same cutting plane method, and a common framework is using ring families; our bucketing argument has been adapted from [27].

Their combinatorial framework is more complex than ours: upper bounds analogous to the lower bounds $\ell(z)$ are needed, and accordingly, their algorithm identifies both outgoing and incoming arcs, as well as removes elements which cannot be contained in any minimizer. The simple trick that enables us to work only with lower bounds, and identify only incoming arcs is repeatedly truncating the value of $f(V)$; thus, we can bound $L_{f^\downarrow}$ in terms of $\ell^-(V)$, as in Claim 4.7.

Our black-box approach clearly separates the combinatorial argument from the cutting plane method, which is used only inside the oracle. In contrast, these two ingredients cannot be clearly separated in [27]. They use the cutting plane method for the formulation using the Lovász extension, and they transform the cutting plane certificate to identify a small norm convex combination in the base polytope. This is analogous to, but substantially more complicated than, our Lemma 6.2. In particular, it is not always possible to identify such a combination, since the constraints of the feasible region can have large coefficients. In such cases, these large coefficients can be used to fix some of the variables to 0 and 1, and hence make progress in terms of the ring family. In contrast, the certificate from our sliding cutting plane algorithm on F_μ can be straightforwardly translated in Lemma 6.2 to satisfy the requirements of the approximate oracle.

References

[1] F. Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends in Machine Learning*, 6(2-3):145–373, 2013.

[2] A. Belloni, R. M. Freund, and S. Vempala. An efficient rescaled perceptron algorithm for conic systems. *Mathematics of Operations Research*, 34(3):621–641, 2009.

[3] U. Betke. Relaxation, new combinatorial and polynomial algorithms for the linear feasibility problem. *Discrete & Computational Geometry*, 32(3):317–338, 2004.

[4] R. G. Bland, D. Goldfarb, and M. J. Todd. The ellipsoid method: A survey. *Operations research*, 29(6):1039–1091, 1981.

[5] D. Chakrabarty, P. Jain, and P. Kothari. Provable submodular minimization using Wolfe's algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, pages 802–809, 2014.

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

[7] S. Chubanov. A polynomial algorithm for linear feasibility problems given by separation oracles. http://www.optimization-online.org/DB_HTML/2017/01/5838.html, 2017.

[8] D. Dadush, L. A. Végh, and G. Zambelli. Rescaling algorithms for linear programming. Part I: Conic feasibility. *arXiv preprint arXiv:1611.06427*, 2016.

[9] G. B. Dantzig. Converting a converging algorithm into a polynomially bounded algorithm. Technical report, Stanford University, 1991.

[10] J. Dunagan and S. Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. *Mathematical Programming*, 114(1):101–114, 2008.

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

[12] A. R. Ene and H. L. Nguyen. Random coordinate descent methods for minimizing decomposable submodular functions. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

[13] A. R. Ene, H. L. Nguyen, and L. A. Végh. Decomposable submodular function minimization: Discrete and continuous. *arXiv preprint arXiv:1703.01830*, 2017.

[14] A. Frank. *Connections in combinatorial optimization*. Number 38 in Oxford lecture series in mathematics and its applications. Oxford Univ Pr, 2011.

[15] S. Fujishige. Lexicographically optimal base of a polymatroid with respect to a weight vector. *Mathematics of Operations Research*, 5(2):186–196, 1980.

[16] S. Fujishige. *Submodular functions and optimization*, volume 58. Elsevier, 2005.

[17] S. Fujishige. A note on submodular function minimization by Chubanov's LP algorithm. http://www.optimization-online.org/DB_HTML/2017/09/6217.html, 2017.

[18] S. Fujishige and S. Isotani. A submodular function minimization algorithm based on the minimum-norm base. *Pacific Journal of Optimization*, 7(1):3–17, 2011.

[19] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[20] E. Hazan and S. Kale. Online submodular minimization. *Journal of Machine Learning Research*, 13(Oct):2903–2922, 2012.

[21] R. Hoberg and T. Rothvoß. An improved deterministic rescaling for linear programming algorithms. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 267–278, 2017.

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

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

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

[25] S. Jegelka and J. A. Bilmes. Online submodular minimization for combinatorial structures. In *Proceedings of the 28th International Conference on Machine Learning (ICML11)*, pages 345–352, 2011.

[26] S. Jegelka, H. Lin, and J. A. Bilmes. On fast approximate submodular minimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 460–468, 2011.

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

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

[29] J. Peña and N. Soheili. A deterministic rescaled perceptron algorithm. *Mathematical Programming*, 155(1-2):497–510, 2016.

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

[31] A. Schrijver. *Combinatorial optimization - Polyhedra and Efficiency*. Springer, 2003.

[32] P. Stobbe and A. Krause. Efficient minimization of decomposable submodular functions. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[33] É. Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255, 1985.

[34] P. Wolfe. Finding the nearest point in a polytope. *Mathematical Programming*, 11(1):128–149, 1976.