

高速なパラメトリック劣モジュラ関数最小化とその応用

永野 清仁
東京大学

kiyohito_nagano@mist.i.u-tokyo.ac.jp

概要

本研究では強射列に対するパラメトリック劣モジュラ関数最小化を扱う。最近 Orlin により得られた劣モジュラ関数最小化アルゴリズムがこの枠組みに拡張可能であることを示して効率的アルゴリズムを与える。本手法は最小比問題や劣モジュラ制約凸最適化問題の高速解法への応用がある。また、今回新たに劣モジュラシステムのロバスト性評価がパラメトリック劣モジュラ関数最小化を用いて効率的に実行可能であることを示した。

A Faster Parametric Submodular Function Minimization Algorithm and Applications

Kiyohito Nagano
University of Tokyo

Abstract

This paper discusses the parametric submodular function minimization problem for strong map sequences. We show that the recent submodular function minimization algorithm of Orlin can be extended to this framework. Applications include faster algorithms for minimum ratio problems and convex optimization over submodular constraints. We also show that the robustness function of a submodular system can be computed efficiently via parametric submodular function minimization.

1 Introduction

In the areas of combinatorial optimization, game theory and other fields, submodular functions are recognized as fundamental functions and interesting subjects of research. They appear in the systems of networks and, at the same time, they naturally model economies of scale. Besides, submodular functions and convex functions are closely related: a set function f defined on the subsets of a finite ground set V is submodular if and only if the Lovász extension [16] of it is convex. By extending the theory of submodular functions, Murota [17] developed a theory of discrete convex analysis.

The first strongly polynomial algorithm for submodular function minimization (SFM) was described by Grötschel, Lovász and Schrijver [11], which relies on the ellipsoid method. The first combinatorial strongly polynomial algorithms for SFM were developed by Iwata, Fleischer, and Fujishige [13] and by Schrijver [20]. More recently, Orlin [19] developed a faster strongly polynomial algorithm for SFM. The running time of Orlin's algorithm is $O(n^5 \text{EO} + n^6)$, where n is the cardinality of the ground set and EO is the time of one function evaluation.

For parametric maximum flow problems in an important class, Gallo, Grigoriadis and Tarjan [9] proposed an efficient method by extending the push/relabel maximum flow algorithm devised by Goldberg and Tarjan [10]. Iwata, Murota and Shigeno [14] extended the result

in [9] to solve polymatroid intersection. Let f_1, \dots, f_k be submodular functions such that they have the same ground set V with $|V| = n$ and they form a strong map sequence [14, 22]. The parametric submodular function minimization is a problem of minimizing each function f_t ($1 \leq t \leq k$). In a similar way to [14], Fleischer and Iwata [4] gave an algorithm for parametric SFM which runs in $O((n^7 + kn^2)\text{EO} + n^8)$ time by extending their push/relabel $O(n^7 \text{EO} + n^8)$ SFM algorithm.

In this paper, we give a faster algorithm for parametric SFM by embedding Orlin's SFM algorithm [19] within the parametric optimization framework successfully. Since it runs in $O((n^5 + kn^3)\text{EO} + n^6)$ time, the asymptotic running time is the same as that of a single execution of Orlin's algorithm as long as $k = O(n^2)$. For each function f_t , by careful discussions on subsystems, our method provides a compact representation of all the minimizers. Faster algorithms for some optimization problems follow from our result. As stated in [4], the discrete Newton method for minimum ratio problems involving submodular functions can be implemented efficiently via parametric SFM. For example, such minimum ratio problems appear as subproblems in approximation algorithms for the set covering with submodular costs [12] and the prize collecting forest problems with submodular penalties [21]. In addition, from the framework provided in [18], various convex minimization problems over submodular constraints, including

the lexicographically optimal base problem [7] and the submodular utility allocation market problem [15], can be solved in $O(n^5 \text{EO} + n^6)$ time.

Furthermore, we deal with robustness functions of optimization problems. Frederickson and Solis-Oba gave an algorithm for computing the robustness functions of minimum spanning trees in [6], and later they extended this method to metroids in [5]. We consider the computation of the robustness function in submodular optimization, which generalizes the results in [5, 6], and showed that it can be solved efficiently by performing parametric SFM.

This paper is organized as follows. In Section 2, we review some basic facts in the theory of submodular functions. Section 3 describes a fast algorithm for parametric SFM. In Section 4, we will see applications of parametric SFM.

2 Preliminaries

A submodular function and its minimizers. Suppose V is a nonempty finite set. Let $f : 2^V \rightarrow \mathbb{R}$ be a submodular set function with $f(\emptyset) = 0$, that is, we have $f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$ for each $X, Y \subseteq V$. In this paper, we call the pair (V, f) a submodular system. A set function $g : 2^V \rightarrow \mathbb{R}$ is monotone if $g(X) \leq g(Y)$ for each $X, Y \subseteq V$ with $X \subseteq Y$, and g is supermodular if $-g$ is submodular. Let EO denote the time of one function evaluation of f . The submodular function minimization (SFM) is a problem of finding a subset $X \subseteq V$ with $f(X)$ minimum, or a structure of a collection of all the minimizers of f , $\arg \min f \subseteq 2^V$. Easily one can show that $\arg \min f$ forms a distributive lattice, i.e., subsets in $\arg \min f$ are closed under union and intersection. So there exist the unique minimal minimizer X^{\min} and the unique maximal minimizer X^{\max} . Furthermore, $\arg \min f$ has a compact directed graph representation. To be precise, there exist a finite set H , a collection $\mathcal{P} = \{P_h : h \in H\} \subseteq 2^V$ and a directed graph $G_f = (H, A)$ with vertices H and arcs A such that subsets in \mathcal{P} are pairwise disjoint, $X^{\max} \setminus X^{\min} = \bigcup_{h \in H} P_h$, and

$$\arg \min f = \{X^{\min} \cup P_{H'} : \text{no arc goes out from } H' \subseteq H \text{ in } G_f\},$$

where $P_{H'} = \cup\{P_h : h \in H'\}$ for $H' \subseteq H$.

Optimization over the base polytope. For a vector $x \in \mathbb{R}^V$, we denote the component of x on $v \in V$ by $x(v)$. The base polytope $\mathbf{B}(f)$ is defined by $\mathbf{B}(f) = \{x \in \mathbb{R}^V : x(X) \leq f(X) \forall X \subseteq V; x(V) = f(V)\}$ where $x(X) = \sum_{v \in X} x(v)$. A point in $\mathbf{B}(f)$ is called a base and an extreme point of $\mathbf{B}(f)$ is called an extreme base. For a base $x \in \mathbf{B}(f)$, we say $X \subseteq V$ is x -tight if $x(X) = f(X)$. Consider any total order \prec in V . The greedy algorithm [3] gives an extreme base $b^\prec \in \mathbb{R}^V$ by setting

$$(1) \quad b^\prec(v) = f(\{u : u \prec v\} \cup \{v\}) - f(\{u : u \prec v\})$$

for each $v \in V$. Thus, each inequality $x(X) \leq f(X)$ determines a supporting hyperplane of $\mathbf{B}(f)$. Conversely,

each extreme base can be obtained in this way. For $x \in \mathbb{R}^V$, define the vector $x^- \in \mathbb{R}^V$ by $x^-(v) = \min\{0, x(v)\}$ for each $v \in V$. For any base $x \in \mathbf{B}(f)$ and each $X \subseteq V$, we have $x^-(V) \leq x(X) \leq f(X)$. Besides, the result of Edmonds [3] immediately implies that

$$(2) \quad \begin{aligned} & \max_x \{x^-(V) : x \in \mathbf{B}(f)\}. \\ & = \min_X \{f(X) : X \subseteq V\} \end{aligned}$$

Hence we can immediately check the optimality of $X \subseteq V$ if we have an optimal solution x to (2). Moreover, using the information about such x , we can compute a compact representation of $\arg \min f$ via the algorithm of Bixby, Cunningham and Topkis [1]. See, e.g., [17, Note 10.11] for details.

Lemma 1 *If an optimal solution x to (2) is given as a convex combination of $O(n)$ extreme bases, a compact representation of $\arg \min f$ can be computed in $O(|V|^3 \text{EO})$ time.*

In many SFM algorithms, we keep a base $x \in \mathbf{B}(f)$ as a convex combination of extreme bases. In order to maximize the concave function $x^-(V)$, the base x will be updated iteratively like $x := x + \alpha x'$, where $\alpha \geq 0$ is a parameter and $x' \in \mathbb{R}^V$ is an update direction. For a base $x \in \mathbf{B}(f)$, we have $\|x\|_1 := \sum_{v \in V} |x(v)| = f(V) - 2x^-(V)$. So, a base is optimal for (2) if and only if it minimizes $\|x\|_1$ over $\mathbf{B}(f)$. Thus, staying 0 seems comfortable for each component $x(v)$. The following property is helpful to understand the algorithm in §3.

Proposition 2 *For any base $x \in \mathbf{B}(f)$, there exists an optimal solution y to (2) such that $\{v \in V : x(v) = 0\} \subseteq \{v \in V : y(v) = 0\}$.*

So, it is natural to say that $x' \in \mathbb{R}^V$ is good if $x'(V) = 0$, $x'(v) = 0 \forall v \in V^0$, $x'(V^+) \leq x'(V^-)$ where $V^0 = \{v : x(v) = 0\}$, $V^+ = \{v : x(v) > 0\}$, $V^- = \{v : x(v) < 0\}$.

Restrictions and contractions. With respect to a subset $S \subseteq V$, the restriction of f , denoted as f^S , is a function defined by $f^S(X) = f(X)$ ($X \subseteq S$), and the contraction of f , denoted as f_S , is a function defined by $f_S(X) = f(X \cup S) - f(S)$ ($X \subseteq V \setminus S$). Clearly, functions $f^S : 2^S \rightarrow \mathbb{R}$ and $f_S : 2^{V \setminus S} \rightarrow \mathbb{R}$ are submodular and satisfy $f^S(\emptyset) = f_S(\emptyset) = 0$. For $S \subseteq V$ and $x \in \mathbb{R}^V$, we denote the subvector $(x(v) : v \in S) \in \mathbb{R}^S$ by x^S . For two disjoint finite sets U_1, U_2 and two vectors $x_1 \in \mathbb{R}^{U_1}$, $x_2 \in \mathbb{R}^{U_2}$, the direct sum $x = x_1 \oplus x_2 \in \mathbb{R}^{U_1 \cup U_2}$ is defined by $x(v) = x_1(v)$ if $v \in U_1$ and $x_2(v)$ if $v \in U_2$. For $A_1 \subseteq \mathbb{R}^{U_1}$ and $A_2 \subseteq \mathbb{R}^{U_2}$, define $A_1 \oplus A_2 = \{x_1 \oplus x_2 : x_1 \in A_1; x_2 \in A_2\}$.

Lemma 3 (see [8]) *Suppose $S \subseteq V$. If $x_1 \in \mathbf{B}(f^S)$ and $x_2 \in \mathbf{B}(f_S)$, $x_1 \oplus x_2 \in \mathbf{B}(f)$ and so $\mathbf{B}(f^S) \oplus \mathbf{B}(f_S) \subseteq \mathbf{B}(f)$. Conversely, if $x \in \mathbf{B}(f)$ and S is x -tight, then $x^S \in \mathbf{B}(f^S)$ and $x^{V \setminus S} \in \mathbf{B}(f_S)$.*

A strong map and parametric submodular function minimization. For submodular functions f_1, f_2 defined on 2^V , if $Y \supseteq X$ implies $f_1(Y) - f_1(X) \leq$

$f_2(Y) - f_2(X)$, we write $f_1 \leftarrow f_2$ or $f_2 \rightarrow f_1$ and we call this relation a strong map [14, 22]. We briefly review basic properties. For $t = 1, 2$, let us denote the minimal minimizer of f_t by X_t^{\min} and the maximal minimizer of f_t by X_t^{\max} . Since $f_1(X \cup Y) - f_1(X) \leq f_2(X \cup Y) - f_2(X) \leq f_2(Y) - f_2(X \cap Y)$ for each $X, Y \subseteq V$, we obtain $X_1^{\min} \supseteq X_2^{\min}$ by setting $X = X_1^{\min}$ and $X_1^{\max} \supseteq X_2^{\max}$ by setting $Y = X_2^{\max}$. For $t = 1, 2$, let b_t^{\leftarrow} be the extreme base of $\mathbf{B}(f_t)$ that is generated by a total order \prec in V via the greedy algorithm. In view of (1), we have $b_1^{\leftarrow} \geq b_2^{\leftarrow}$. Given a bunch of submodular functions that form a strong map sequence, the parametric submodular function minimization is a problem of finding a minimizer of every function. In §3, we give an efficient algorithm for parametric SFM.

The dual of a submodular function. Function $f^\# : 2^V \rightarrow \mathbb{R}$, the dual of the submodular function f , is defined by $f^\#(X) = f(V) - f(V \setminus X)$ ($X \subseteq V$). Clearly, $f^\#$ is supermodular. The characteristic vector $\chi_X \in \mathbb{R}^V$ of $X \subseteq V$ is defined by $\chi_X(v) = 1$ if $v \in X$ and 0 otherwise. Define $\varphi_1(p) = \max_{x \in \mathbf{B}(f)} \langle p, x \rangle$ and $\varphi_2(p) = \min_{x \in \mathbf{B}(f)} \langle p, x \rangle$ ($p \in \mathbb{R}^V$), where $\langle p, x \rangle = \sum_{v \in V} p(v)x(v)$. It is known that $\varphi_1(\chi_X) = f(X)$ and $\varphi_2(\chi_X) = f^\#(X)$ for each $X \subseteq V$.

3 Parametric SFM

We consider the efficient computation of a minimizer of every function in a strong map sequence of submodular functions $f_t : 2^{\{1, \dots, n\}} \rightarrow \mathbb{R}$ ($t = 1, \dots, k$) such that $f_1 \leftarrow \dots \leftarrow f_k$ and $f_t(\emptyset) = 0$ for each t . We assume that the time of function evaluation of each f_t is bounded by EO. Our algorithm **P-SFM** for parametric SFM, $\min_X \{f_t(X) : X \subseteq \{1, \dots, n\}\}$ for $t = 1, \dots, k$, runs in $O(n^5 \text{EO} + n^6 + kn^3 \text{EO})$ time. If $k = 1$ it corresponds to Orlin's algorithm for SFM [19], which runs in $O(n^5 \text{EO} + n^6)$ time. Remark that our algorithm finds a compact representation of all the minimizers of each function f_t .

Throughout this section, using some t and subset $V \subseteq \{1, \dots, n\}$, let $f = f_t^V$ be a restriction of f_t by V . Define $X_t^{\min} := \cap \{X : X \in \arg \min f_t\}$, $X_t^{\max} := \cup \{X : X \in \arg \min f_t\}$ for each $t = 1, \dots, k$. For a total order \prec in V and $1 \leq t \leq k$, we denote by b_t^{\leftarrow} the extreme base of $\mathbf{B}(f_t^V)$ generated by \prec . We have the following relations.

Lemma 4 $X_1^{\min} \supseteq \dots \supseteq X_k^{\min}$, $X_1^{\max} \supseteq \dots \supseteq X_k^{\max}$ and $b_1^{\leftarrow} \leq \dots \leq b_k^{\leftarrow}$.

Our algorithm **P-SFM** initializes $V := V^i = \{1, \dots, n\}$ and repeatedly restrict V . It minimizes f_1, \dots, f_k in this order, and always maintains the function $f = f_t^V$ satisfying

- R1. V is nonincreasing and t is nondecreasing, i.e., if $f = f_t^{V'}$ at some iteration and $f = f_{t''}^{V''}$ after that, $V' \supseteq V''$ and $t' \leq t''$;
- R2. Relation $X_t^{\min} \subseteq V$ holds while minimizing f_t^V .

Notice that the properties R1 and R2 are consistent with

Lemma 4. We call a set of iterations of **P-SFM** trying to minimize f_t a t -phase. To give the algorithm in §3.3, roughly speaking, we explain inner operations of **P-SFM** in §3.1 and outer operations in §3.2.

3.1 Preparations from Orlin's result [19]

We make some preparations to give an iterative local search algorithm. They are from Orlin's paper [19]. In each t -phase, the algorithm **P-SFM** tries to find a minimizer of $f = f_t^V$ by updating the base $x \in \mathbf{B}(f)$. Now we fix t with $1 \leq t \leq k$.

Valid distance functions. A distance function is a nonnegative integer vector $d \in \mathbb{Z}_{\geq 0}^V$ and each d induces a total order \prec_d in $V \subseteq \{1, \dots, n\}$:

$$u \prec_d v \text{ if } d(u) < d(v) \text{ or } d(u) = d(v) \text{ and } u < v.$$

Let $b_d \in \mathbf{B}(f)$ be the extreme base generated by \prec_d . The algorithm keeps

- distance functions D with $|D| = O(n)$;
- a vector $\lambda \in \mathbb{R}^D$ s.t. $\lambda(d) > 0$ for each $d \in D$ and $\sum_{d \in D} \lambda(d) = 1$;
- a submodular function $f = f_t^V$ with $f(\emptyset) = 0$.

Let $D_{\min}(v) = \min\{d(v) : d \in D\}$ for each $v \in V$. The triple (D, λ, f) , which generates a base $x = \sum_{d \in D} \lambda(d) \cdot b_d \in \mathbf{B}(f)$, is valid if

- V1. $d(v) = 0$ for all $v \in \{u \in V : x(u) < 0\}$;
- V2. $d \in D$, and $D_{\min}(v) \leq d(v) \leq D_{\min}(v) + 1$ for all $v \in V$ and $d \in D$.

If the triple is valid, we also say D is valid. In each t -phase, starting with some valid triple, the algorithm maintains valid (D, λ, f) and a base $x = \sum_d \lambda(d) \cdot b_d$, and it updates the triple until x becomes optimal for (2). Let $V^0 = \{v \in V : x(v) = 0\}$, $V^+ = \{v \in V : x(v) > 0\}$, and $V^- = \{v \in V : x(v) < 0\}$.

Primary and secondary distance functions. For a distance function d and an element $v \in V$, let $d' = \text{INC}(d, v)$ be the distance function such that $d'(u) = d(v) + 1$ if $u = v$ and $d'(u) = d(u)$ otherwise. By submodularity,

$$(3) \quad b_{d'}(v) - b_d(v) \leq 0 \ \& \ b_{d'}(u) - b_d(u) \geq 0 \text{ if } u \neq v.$$

We maintain collection D of distance functions in non-decreasing order of $d(V) = \sum_v d(v)$. Given D , for each $v \in V$, let the primary distance function $p(v) \in D$ be the first distance function in D such that $d(V) = D_{\min}(v)$ and let secondary distance function $s(v) \in \mathbb{Z}_{\geq 0}^V$ be $\text{INC}(p(v), v)$, which is not necessarily in D .

The auxiliary matrix and the update. Given a base x generated by a valid triple (D, λ, f) , let us see how x will be updated. Let $c_v \in \mathbb{R}^V$ be a vector defined by $c_v = b_{s(v)} - b_{p(v)}$ for each $v \in V$. Define the auxiliary matrix $A^* \in \mathbb{R}^{V^0 \times V^0}$ by $A^*(u, v) = c_v(u)$ for $u, v \in V^0$. In view of (3), we have that $A^*(u, v) \leq 0$ if $u = v$ and $A^*(u, v) \geq 0$ if $u \neq v$. Furthermore, equality $c_v(V) = 0$ implies that each column sum $\sum_{u \in V^0} A^*(u, v) = c_v(V^0) \leq 0$.

Theorem 5 ([19]) *If A^* is singular, there is a vector $\gamma \in \mathbb{R}^{V^0}$ s.t. $\gamma \geq \mathbf{0}$ and $A^*\gamma = \mathbf{0}$. If A^* is nonsingular, each component of $(A^*)^{-1}$ is nonpositive and so the solution γ to $A^*\gamma = -c$ is nonnegative whenever $c \geq \mathbf{0}$.*

If $V^+ = \emptyset$, x is optimal for (2) and V is a minimizer of f because $x^-(V) = f(V)$. So we only consider the case where $V^+ \neq \emptyset$. To compute a good update direction $x' \in \mathbb{R}^V$, we choose some $v^* \in V^+$ and compute a vector $\tilde{\gamma} \in \mathbb{R}^{V^0 \cup \{v^*\}}$ such that

$$(4) \quad \tilde{\gamma} \neq \mathbf{0}, \quad \tilde{\gamma} \geq \mathbf{0} \quad \text{and} \quad A^*\gamma + c\eta = \mathbf{0},$$

where $\gamma = \tilde{\gamma}^{V^0} \in \mathbb{R}^{V^0}$, $\eta = \tilde{\gamma}(v^*) \in \mathbb{R}$ and $c = (c_v : v \in V^0) (\geq \mathbf{0})$. By solving a system of equations on A^* , set $\eta = 0$ if A^* is singular and $\eta = 1$ if A^* is nonsingular. With the aid of Theorem 5, the vector $\tilde{\gamma}$ satisfying (4) can be computed in $O(|V^0|^3) = O(n^3)$ time via Gaussian elimination. Then we set the update direction $x' := \sum_{v \in V^0 \cup \{v^*\}} \tilde{\gamma}(v) \cdot c_v \in \mathbb{R}^V$. Since $x'(V) = 0$, $x'(v) = 0$ if $v \in V^0$, $x'(v) \leq 0$ if $v = v^*$ and $x'(v) \geq 0$ if $v \in V \setminus (V^0 \cup \{v^*\})$, we can say that x' is a good update direction. The algorithm updates $x := x + \alpha x'$ where

$$(5) \quad \begin{aligned} \alpha &:= \min\{\alpha_1, \alpha_2\}, \\ \alpha_1 &:= \max\{\alpha : x(v) + \alpha x'(v) \geq 0 \forall v \in V^+\}; \\ &\quad x(v) + \alpha x'(v) \leq 0 \forall v \in V^-, \\ \alpha_2 &:= \max\{\alpha : \alpha \sum_v \{\tilde{\gamma}(v) : v \text{ with } p(v) = d\} \\ &\quad \leq \lambda(d) \quad \forall d \in D\} \end{aligned}$$

using Proposition 2, implicitly. We say the update is saturating if $\alpha = \alpha_2$, and nonsaturating otherwise. Accordingly, distance functions D and coefficients $\lambda(d)$ are updated as follows:

$$(6) \quad \begin{aligned} D &:= D \cup \{s(v) : v \in V^0 \cup \{v^*\}\}, \\ \lambda(d) &:= \lambda(d) + \alpha \sum_v \{\tilde{\gamma}(v) : v \text{ with } s(v) = d\} \\ &\quad - \alpha \sum_v \{\tilde{\gamma}(v) : v \text{ with } p(v) = d\}, \quad \forall d \in D. \end{aligned}$$

As x' is a good update direction, if $v \in V^0$ at some iteration of t -phase, $v \in V^0$ at all subsequent iterations of t -phase. Since each distance function d is added to D as a secondary function $s(v)$ for some $v \in V^0 \cup V^+$, the properties V1 and V2 remain satisfied. If the update is nonsaturating, the size of V^0 increases. So, the number of nonsaturating updates is at most n in each t -phase. When the update is saturating, at least one distance function $d = p(v)$ will be deleted from D . Hence, the auxiliary matrix changes by some columns. If A^* is changed by q columns at a given iteration, the system of equations can be solved in $O(qn^2)$ time at the next iteration, which is faster than $O(n^3)$.

3.2 Transitions and restrictions

Next, let us see transitions of functions and restrictions of the ground set V in our algorithm **P-SFM**. For each $t = 1, \dots, k$, our algorithm divides the submodular system (V^i, f_t) into small subsystems. For distance functions $D \subseteq \mathbb{Z}_{\geq 0}^V$ and $U \subseteq V$, we denote $\{d(v) : v \in U\} : d \in D$ by $D|U$.

Transitions of functions. In each t -phase, a base $x \in \mathbf{B}(f_t^V)$ will be updated until it becomes optimal for $\max\{x^-(V) : x \in \mathbf{B}(f_t^V)\}$ and after that, if $t + 1 \leq k$, the base $x \in \mathbf{B}(f_t^V)$ generated by (D, λ, f_t^V) will be transformed into the base $\tilde{x} \in \mathbf{B}(f_{t+1}^V)$ generated by (D, λ, f_{t+1}^V) . Then, the algorithm continues. This transformation maintains the validity.

Lemma 6 *If $1 \leq t \leq k - 1$ and (D, λ, f_t^V) is valid, (D, λ, f_{t+1}^V) is also valid.*

Proof: Property V2 does not depend on the function. By Lemma 4, $x \leq \tilde{x}$. Thus V1 is also satisfied. \square

Key lemmas for proper restrictions. We want to restrict V so that the property R2 is always satisfied. In addition, we want to solve the problem $\max_x \{x^-(V^i) : x \in \mathbf{B}(f_i)\}$ for each t to obtain a digraph representation of all the minimizers $\arg \min f_t$ with the aid of Lemma 1. The following two lemmas are useful to achieve that.

Lemma 7 (Orlin [19]) *Given $x \in \mathbf{B}(f)$. If subset $S \subseteq V$ is x -tight and $x(v) \geq 0$ for $v \in V \setminus S$, there is a minimizer $S^* \subseteq S$ of f .*

Proof: For $X \subseteq V$, $f(X) \geq f(X \cap S) + f(X \cup S) - f(S) \geq f(X \cap S) + x(X \cup S) - x(S) \geq f(X \cap S)$. \square

Lemma 8 *Given $S \subseteq V$ and $\tilde{x} \in \mathbf{B}(f_S) \subseteq \mathbb{R}^{V \setminus S}$ such that $\tilde{x} \geq \mathbf{0}$. If $\tilde{y} \in \mathbf{B}(f^S)$ is optimal for $\max_y \{y^-(S) : y \in \mathbf{B}(f^S)\}$, $x^* = \tilde{y} \oplus \tilde{x} \in \mathbb{R}^V$ is optimal for $\max_x \{x^-(V) : x \in \mathbf{B}(f)\}$.*

Proof: By Lemma 3, $x^* \in \mathbf{B}(f)$. By Lemma 7, there exists a minimizer S^* of f such that $S^* \subseteq S$. Clearly we have $\tilde{x}^-(V \setminus S) = 0$ and $\tilde{y}^-(S) = f(S^*)$. It follows that $(x^*)^-(V) = f(S^*)$. Therefore, x^* is optimal for $\max\{x^-(V) : x \in \mathbf{B}(f)\}$. \square

Restrictions and subsystems. Consider some iteration in which the algorithm **P-SFM** is minimizing $f = f_t^V$, the triple (D, λ, f) is valid and V has already been restricted $r - 1 \geq 0$ times. Let $G_D = (V, A_D)$ be a digraph with vertices V and arcs $A_D = \{(u, v) : u \prec_d v \text{ for some } d \in D\}$ and let

$$R_D = \{v \in V : v \text{ can reach } V^- \text{ in } G_D\}.$$

Clearly $x(v) \geq 0$ for $v \in V \setminus R_D$. Since $u \prec_d v$ for all $d \in D$, $u \in R_D$ and $v \in V \setminus R_D$, we obtain $x(R_D) = f(R_D)$. By Lemma 7, there exists a minimizer X of f with $X \subseteq R_D$. We can compute R_D in $O(n^2)$ time.

Suppose that $R_D \neq V$ (then we are going to restrict V) and define $t^{(r)} = t$. In view of Lemma 3 and 8, consider two subvectors $x^{(0)} = x^{R_D}$, $x^{(r)} = x^{V \setminus R_D}$ and two submodular systems $\mathcal{S}^{(0)} = (R_D, f^{R_D})$, $\mathcal{S}^{(r)} = (V \setminus R_D, f^{R_D})$. Remark that $x^{(r)} \geq \mathbf{0}$. Afterwards, the algorithm runs on the subsystem $\mathcal{S}^{(0)}$. However, we do not throw away information about $\mathcal{S}^{(r)}$. Define $U^{(r)} = V \setminus R_D$, $T^{(r)} = R_D$, $D^{(r)} = D|U^{(r)}$, $\lambda^{(r)} = \lambda$ and set $V := R_D$ (thereby $f := f^{R_D} = f_t^{R_D}$, $D := D|R_D$). The new triple (D, λ, f) remains valid. For convenience, define $T^{(0)} = V^i$. Inductively, we have $U^{(r)} = T^{(r-1)} \setminus T^{(r)}$ and V^i is equal to disjoint

union $V \cup U^{(r)} \cup \dots \cup U^{(1)}$. Moreover, define function $f_t^{(r)} : 2^{U^{(r)}} \rightarrow \mathbb{R}$ by $f_t^{(r)} = (f_t^{T^{(r-1)}})_{T^{(r)}}$ for each $t' = t^{(r)}, \dots, k$. Let $x_t^{(r)}$ be the base in $\mathbf{B}(f_t^{(r)})$ generated by the triple $(D^{(r)}, \lambda^{(r)}, f_t^{(r)})$ for each $t' = t^{(r)}, \dots, k$.

Lemma 9 $x_t^{(r)} \geq 0$ for each $t = t^{(r)}, \dots, k$.

Proof: We have $S^{(r)} = (U^{(r)}, f_t^{(r)})$ and $x_t^{(r)} \geq 0$. From the relation $f_t^{(r)} \leftarrow \dots \leftarrow f_k^{(r)}$ and Lemma 4, it follows that $0 \leq x_t^{(r)} \leq \dots \leq x_k^{(r)}$. \square

By combining Lemma 8 and Lemma 9, we will see that, at the end of t -phase, an optimal solution x_t to $\max_x \{x^-(V^i) : x \in \mathbf{B}(f_t)\}$ can be simply constructed.

3.3 The algorithm P-SFM

We describe the algorithm for parametric SFM, $\min_X \{f_t(X) : X \subseteq V^i\}$ for each $t = 1, \dots, k$. When the size of D grows large, it performs a procedure **Reduce** (D, λ, f) that outputs distance functions $D' \subseteq D$ and a positive vector $\lambda' \in \mathbb{R}_{>0}^D$ such that extreme bases $b_{d'} \in \mathbf{B}(f)$ ($d' \in D'$) are affinely independent, $\sum_{d' \in D'} \lambda'(d') = 1$ and $\sum_{d \in D} \lambda(d) \cdot b_d = \sum_{d' \in D'} \lambda'(d') \cdot b_{d'}$. The running time of this procedure is $O(n^2|D|)$ using Gaussian elimination. We also run a procedure **Update-Dist** (D, λ) to eliminate any function d with $\lambda(d) = 0$ from D , maintain D in nondecreasing order of $d(V)$ and update distance functions $\{p(v) : v \in V\}$ and $\{s(v) : v \in V\}$.

The following algorithm **P-SFM** finds the subset $Z_t \subseteq V^i$ and the vector $x_t \in \mathbb{R}^{V^i}$ for each $t = 1, \dots, k$.

Algorithm P-SFM

Initialization :

$V := V^1$; $d_0 := \mathbf{0}$; $D := \{d_0\}$; $\lambda(d_0) := 1$;
 $t := 1$; $r := 0$; $T^{(0)} = V^1$;

while $t \leq k$ begin

$f := f_t^V$; $x := \sum_{d \in D} \lambda(d) \cdot b_d \in \mathbf{B}(f_t^V)$;
while $V^+ \neq \emptyset$ **begin**
 Choose $v^* \in V^+$ and let $c = (c_v : v \in V^0)$;
 Compute $\tilde{\gamma} \in \mathbb{R}^{V^0 \cup \{v^*\}}$; s. t. (4)
 Set $x' := \sum_{v \in V^0 \cup \{v^*\}} \tilde{\gamma}(v) \cdot c_v$;
 Set $x := x + \alpha x'$, where α is defined in (5);
 Update D, λ as in (6);
 If $|D| \geq 3n$ **then** **Reduce** (D, λ, f) ;
 Update-Dist (D, λ) ;
 If $R_D \neq V$ **then begin**
 $r := r + 1$;
 $U^{(r)} = V \setminus R_D$; $T^{(r)} = R_D$;
 $D^{(r)} = D|_{U^{(r)}}$; $\lambda^{(r)} = \lambda$;
 $V := R_D$; **Reduce** $(D^{(r)}, \lambda^{(r)}, f_t^{(r)})$;
 end;
end;
Return $Z_t = V$ and $x_t = x \oplus x_t^{(r)} \oplus \dots \oplus x_t^{(1)}$;
 $t := t + 1$;

end;

Let us show the correctness of the algorithm **P-SFM**.

Proposition 10 *During the execution of the algorithm P-SFM, the following conditions are satisfied:*

- (10.a) the triple (D, λ, f) is valid;
- (10.b) $D_{\min}(v)$ is nondecreasing and $D_{\min}(v) \leq n$;
- (10.c) while minimizing f_t^V , $X_t^{\min} \subseteq V$;
- (10.d) $V^i =$ disjoint union $V \cup U^{(1)} \cup \dots \cup U^{(r)}$.

Proof: (10.d) is obvious, so we consider the other conditions. Initially, conditions (10.a) to (10.c) are satisfied. First, fix t with $1 \leq t < k$ and assume that they are true at the end of t -phase. Then, at the beginning of $(t+1)$ -phase, by Lemma 6, (10.a) is satisfied and, by Lemma 4, $X_{t+1}^{\min} \subseteq X_t^{\min} \subseteq V$ holds and so (10.c) is satisfied. Since D is unchanged, (10.b) is also true.

Next, consider the iterations of each t -phase. In §3.1, it is shown that (10.a) remains satisfied. By Lemma 7, $X_t^{\min} \subseteq R_D$ holds if $X_t^{\min} \subseteq V$ and thus (10.c) remains satisfied. Consider (10.b). Each d is added to D as $s(u)$ for some $u \in V$, so $D_{\min}(v) \leq d(v) \leq D_{\min}(v) + 1$ for each $v \in V$. Hence $D_{\min}(v)$ is nondecreasing and it increases by at most 1. If $D_{\min}(v) = |V| (\leq n)$ for some $v \in V$, one can show that $v \notin R_D$. Thus $v \notin V$ at all subsequent iterations. \square

Theorem 11 *When the algorithm P-SFM terminates, Z_t is a minimizer of f_t and x_t is an optimal solution to $\max_x \{x^-(V^i) : x \in \mathbf{B}(f_t)\}$ for each $t = 1, \dots, k$.*

Proof: At the end of each t -phase, we have $V^+ = \emptyset$, and so x is optimal for (2) and V is a minimizer of f_t^V . As $X_t^{\min} \subseteq V$, $Z_t = V$ is a minimizer of f_t . Using Lemma 8 and 9 repeatedly, x_t is optimal for $\max_x \{x^-(V^i) : x \in \mathbf{B}(f_t)\}$. \square

Although we are dealing with multiple submodular functions, we can bound the number of columns added to A^* and the number of distance functions added to D in the same way as Orlin's paper [19]. The proof of Lemma 12 is deferred to the full version of the paper.

Lemma 12 (also see [19]) *Throughout the algorithm P-SFM, the number of distance functions added to D and the number of columns added to A^* are both $O(n^4)$.*

Since V^0 increases after a nonsaturating update, they are performed at most n times in a t -phase and at most kn times during the algorithm. Each saturating update deletes at least one distance function. Thus, by Lemma 12, the number of saturating updates is $O(n^4)$. So, the number of iterations of **P-SFM** is $O(n^4 + kn)$.

Theorem 13 *The algorithm P-SFM runs in $O(n^2 \text{EO} + n^6 + k(n^2 \text{EO} + n^3))$ time.*

Proof: Each computation of the initial base x of a t -phase takes $O(n^2 \text{EO})$ time. Consider the complexity to add columns to A^* and solve the systems of equations on A^* . At the first iteration of a t -phase, it takes $O(n^2 \text{EO} + n^3)$ time to construct A^* and solve the system of equations on A^* . Each column can be computed in $O(n \text{EO})$ time and it takes $O(n^2)$ additional time to add the column to A^* and transform A^*

into a canonical form for solving the system of equations. Thus by Lemma 12, the running time to add columns to A^* and solve the systems of equations on A^* is $(n^5\text{EO} + n^6 + k(n^2\text{EO} + n^3))$ time.

Next, consider the executions of **Reduce**. Each execution deletes at least $3n - |V| \geq 2n$ distance functions and takes $O(n^3)$ time. By Lemma 12, the running time for carrying out **Reduce** is $O(n^6)$. Moreover, it takes $O(n^2)$ to compute R_D and $O(n \log n)$ time to perform **Dist-Update** at each iteration. Since the number of iterations of the algorithm is $O(n^4 + kn)$, it takes $(n^6 + kn^3)$ time to carry out these operations.

As a result, the running time of the algorithm **P-SFM** is $O(n^5\text{EO} + n^6 + k(n^2\text{EO} + n^3))$. \square

At the end of each t -phase, for each $1 \leq j \leq r$, subvector $x_t^{(j)}$ is a convex combination of at most $|U^{(j)}|$ extreme bases of $\mathbf{B}(f_t^{(j)})$. As $|V| + \sum_{j=1}^r |U^{(j)}| = n$, we can easily represent x_t as a convex combination of at most n extreme bases of $\mathbf{B}(f_t)$. By Lemma 1 and Theorem 13, we have:

Corollary 14 *Subsets X_t^{\min}, X_t^{\max} ($1 \leq t \leq k$) and compact representations of $\arg \min f_t$ ($1 \leq t \leq k$) can be computed in $O((n^5 + kn^3)\text{EO} + n^6)$ time in total.*

3.4 Minimization in the reverse order

The algorithm **P-SFM** in §3.3 minimizes $f_1 \leftarrow \dots \leftarrow f_k$ in this order. It is easy to minimize them in the reverse order. Define the function $\bar{f}_t : 2^{V^i} \rightarrow \mathbb{R}$ by $\bar{f}_t(X) = f_t(V^i \setminus X)$ ($X \subseteq V^i$) for each $t = 1, \dots, k$. Clearly, each \bar{f}_t is submodular and we have $\bar{f}_k \leftarrow \dots \leftarrow \bar{f}_1$. By applying the algorithm **P-SFM** to $\bar{f}_k, \dots, \bar{f}_1$, functions $f_k \rightarrow \dots \rightarrow f_1$ can be minimized in this order and the running time is still $O((n^5 + kn^3)\text{EO} + n^6)$ in total. If we need to distinguish the algorithm **P-SFM** in §3.3 (the reverse order algorithm), we call the latter one **P_R-SFM**.

4 Applications

Let us see some applications of parametric submodular function minimization algorithms.

4.1 Some applications

Minimum ratio problems. Let $f : 2^V \rightarrow \mathbb{R}$ be a submodular function with $f(\emptyset) = 0$ and $w \in \mathbb{R}_{>0}^V$ be a nonnegative weight vector with $w \neq 0$. As stated in [4], one application is finding the minimizer of $f(X)/w(X)$ via Dinkelbach's discrete Newton method [2]. In this case, the number of the iterations of this method is at most $|V|$. In the execution of this method, for some $t_1 > \dots > t_k \geq 0$, we have to minimize k submodular functions $f - t_1 w, \dots, f - t_k w$ in this order. As we have the relation $f - t_1 w \leftarrow \dots \leftarrow f - t_k w$, the ratio $f(X)/w(X)$ can be minimized in $O(|V|^5\text{EO} + |V|^6)$ time using the algorithm **P-SFM**.

In the same way, we can deal with a more generalized minimum ratio problem $f(X)/g(X)$, where $g : 2^V \rightarrow \mathbb{R}$ is a set function such that g is monotone and supermodular. The number of the iterations is still at most $|V|$ (refer to [8, §7.2 (b.1)]) and, for $t_1 > \dots > t_k \geq 0$, we have the relation $f - t_1 g \leftarrow \dots \leftarrow f - t_k g$. Therefore, the problem can also be solved in $O(|V|^5\text{EO} + |V|^6)$ time.

Convex optimization over base polytopes. In [18], using parametric submodular function minimization algorithms, Nagano proposed a general framework for efficiently solving separable convex optimization problems over submodular constraints, including the lexicographically optimal base problems [7] and the submodular utility allocation market problems [15]. Using this framework, with the aid of the algorithms **P-SFM** and **P_R-SFM**, each of them can be solved in the same order as a single implementation of **P-SFM**.

4.2 Measuring robustness

As a new application of minimum ratio problems involving submodular functions, we consider the efficient computation of the robustness function of submodular optimization, which is a generalization of the results about minimum spanning trees [6] and matroid optimization [5].

We are given a monotone, submodular function $f : 2^V \rightarrow \mathbb{R}$ with $f(\emptyset) = 0$, an initial weight vector $w^\circ \in \mathbb{R}^V$, a positive cost vector $c \in \mathbb{R}^V$ and a budget $\beta^\circ \geq 0$. At first, a weight vector $w = w^\circ$ and it can be changed using the budget. We pay $c(v) \cdot \Delta$ for increasing the weight $w(v)$ of $v \in V$ by $\Delta \geq 0$. Define $\varphi(w) = \min_{x \in \mathbf{B}(f)} \langle w, x \rangle$ ($w \in \mathbb{R}^V$). For any nonnegative number $\beta \geq 0$, let

$$F(\beta) = \max_p \{ \varphi(w^\circ + p) - \varphi(w^\circ) : \langle c, p \rangle \leq \beta; p \geq 0 \}.$$

We call $F : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ the robustness function. In this section, we consider the computation of $F(\beta^\circ)$. This value measures the robustness of submodular optimization. We say the weights of $X \subseteq V$ are lifted by Δ if the weight vector w is changed to $w + \Delta \cdot \chi_X$. In the same way as [5, 6], starting with $w = w^\circ$, our algorithm **Lift** repeatedly update the weights like $w := w + \Delta \cdot \chi_X$ to compute $F(\beta^\circ)$. For any given weight vector w , let $w_1 < \dots < w_\ell$ be the distinct values of $w(v)$ ($v \in V$) and $w_{\ell+1} = +\infty$.

A set of w -minimum bases. Let us see the structure of a set of w -minimum bases, that is, a set of bases that minimize $\langle w, x \rangle$ over $\mathbf{B}(f)$. Put $L_0 = \emptyset$. For $i = 1, \dots, \ell$, put $L_i = \{v \in V : w(v) \leq w_i\}$ and put $K_i = L_i \setminus L_{i-1}$. We call each L_i a level set of w . Define $f^i : 2^{K_i} \rightarrow \mathbb{R}$ by $f^i = (f^{L_i})_{L_{i-1}}$ for each i . Function $f_w : 2^V \rightarrow \mathbb{R}$ defined by $f_w(X) = \sum_{i=1}^{\ell} f^i(X \cap K_i)$ for each $X \subseteq V$ is submodular. We have $\mathbf{B}(f_w) = \mathbf{B}(f^1) \oplus \dots \oplus \mathbf{B}(f^\ell)$. It is known that a base $x \in \mathbf{B}(f)$ is w -minimum if and only if $x \in \mathbf{B}(f_w)$ (see [8]). So, $\varphi(w) = \langle w, x_w \rangle$ for any $x_w \in \mathbf{B}(f_w)$.

The efficiency of each subset. Next, we try to

find a good subset for the efficient increase of φ by lifting. Fix a subset $X \subseteq V$ and define $\text{tol}(X, w) = \min_{v \in X} \text{tol}(v, w)$, where $\text{tol}(v, w) = w_{i+1} - w_i$ for $v \in V$ with $w(v) = w_i$. Let Δ be a number such that $0 \leq \Delta \leq \text{tol}(X, w)$ and let $w_\Delta = w + \Delta \cdot \chi_X$. By the definition of $\text{tol}(v, w)$, if $\Delta \neq \text{tol}(v, w)$, any w_Δ -minimum base x is also w -minimum. Thus we have $\varphi(w_\Delta) = \min_{x \in \mathbf{B}(f)} \langle w_\Delta, x \rangle = \min_{x \in \mathbf{B}(f_w)} \langle w_\Delta, x \rangle = \varphi(w) + \min_{x \in \mathbf{B}(f_w)} \langle \Delta \cdot \chi_X, x \rangle = \varphi(w) + \Delta \cdot f_w^\#(X)$. Define the efficiency of X at w as $\text{eff}(X, f, w) = \frac{f_w^\#(X)}{c(X)}$ ($= \frac{\varphi(w_\Delta) - \varphi(w)}{\Delta \cdot c(X)}$). A subset $X \subseteq V$ of largest efficiency is a good one for the efficient increase of φ . Remark that $f_w^\#$, the dual of f_w , can be represented as

$$(7) \quad f_w^\#(X) = \sum_{i=1}^{\ell} f^{i\#}(X \cap K_i)$$

for each $X \subseteq V$. The following lemma says that there exists a subset of largest efficiency such that all elements in it have the same weight.

Lemma 15 *Let $X \subseteq V$ be a subset of largest efficiency at w . Suppose $X \cap K_i \neq \emptyset$. Then we have $\text{eff}(X \cap K_i, f, w) = \text{eff}(X, f, w)$. Furthermore, $\text{eff}(Y, f, w) = \text{eff}(X, f, w)$ for subset $Y \subseteq K_i$ that maximizes $f^{i\#}(Y)/c(Y)$ over 2^{K_i} .*

The algorithm Lift. Our algorithm **Lift**, which is similar to those in [5, 6], updates the weight vector in a greedy way. By Lemma 15, each iteration of **Lift** finds a subset of largest efficiency at w . The parameter β represents the budget already spent. The algorithm **Lift** halts when β becomes β° .

Algorithm Lift($f, \beta^\circ, w^\circ, c$)

```

 $\beta := 0$ ;  $w := w^\circ$ ;
while  $\beta < \beta^\circ$  begin
  Find  $X^i \subseteq K_i$  with  $f^{i\#}(X^i)/c(X^i)$  maximum
  for each  $i = 1, \dots, \ell$ ;
  Set  $X := X^{i^*}$ , where  $X^{i^*}$  is a subset with
   $\text{eff}(X^{i^*}, f, w) = \max_{1 \leq i \leq \ell} \text{eff}(X^i, f, w)$ ;
  Set  $\Delta = \min\{\text{tol}(X, w), (\beta^\circ - \beta)/c(X)\}$ ;
  Set  $w := w + \Delta \cdot \chi_X$ ;
  Set  $\beta := \beta + \Delta \cdot c(X)$ ;
end;
Return  $w_L := w$ ;

```

Since $\sum_i |K_i| = |V|$, each iteration takes $O(|V|^5 \text{EO} + |V|^6)$ time using the discrete Newton method and the algorithm **P-SFM** (see §4.1). Let $w_1^\circ < \dots < w_{\ell^\circ}^\circ$ be distinct values of $w^\circ(v)$ ($v \in V$).

Theorem 16 *The number of while-loops of Lift is at most $n\ell^\circ$. Thus, it runs in $O(|V|^6 \ell^\circ \text{EO} + |V|^7 \ell^\circ)$ time.*

The correctness of Lift. Let $p^* \in \mathbb{R}^V$ be an optimal solution to $\max_p \{\varphi(w^\circ + p) : \langle c, p \rangle \leq \beta; p \geq 0\}$ and let $w^* = w^\circ + p^*$. We will show that the algorithm **Lift** computes $F(\beta^\circ)$ correctly, i.e., the equality $\varphi(w_L) = \varphi(w^*)$ is satisfied.

Given two weight vectors $w, w' \in \mathbb{R}^V$ and subset $X \subseteq V$, we consider the relation between $f_w^\#(X)$ and $f_{w'}^\#(X)$. For each $v \in V$, let $L_v = \{u \in V : w(u) \leq w(v)\}$ and $L'_v = \{u \in V : w'(u) \leq w'(v)\}$.

Lemma 17 *If $L'_v \subseteq L_v$ for each $v \in X$, we have $f_w^\#(X) \leq f_{w'}^\#(X)$.*

Proof: Consider a total order \prec in X such that $u \prec v$ for each $u, v \in X$ with $w'(u) < w'(v)$. Define $X_v = L_v \setminus \{u \in X : v \prec u\}$, $X'_v = L'_v \setminus \{u \in X : v \prec u\}$ for each $v \in X$. By (7), one can observe that

$$f_w^\#(X) = \sum_{v \in X} \{f(X_v) - f(X_v \setminus \{v\})\},$$

$$f_{w'}^\#(X) = \sum_{v \in X} \{f(X'_v) - f(X'_v \setminus \{v\})\}.$$

By assumption, we have $X'_v \subseteq X_v$ for each $v \in X$. Therefore, by submodularity, $f(X_v) - f(X_v \setminus \{v\}) \leq f(X'_v) - f(X'_v \setminus \{v\})$. Thus $f_w^\#(X) \leq f_{w'}^\#(X)$. \square

With the aid of Lemma 17, we can show the correctness of the algorithm **Lift**.

Theorem 18 *The algorithm Lift correctly computes the weight vector w^* and the value $F(\beta^\circ)$.*

Proof: Let q^* be the number of iterations of **Lift**. Assume that in iteration q ($1 \leq q \leq q^*$) the parameter β changes from β_{q-1} to β_q and the weights in the subset S_q are lifted by the amount Δ_q . Clearly, $0 = \beta_0 < \dots < \beta_{q^*} = \beta^\circ$ and $\beta_q - \beta_{q-1} = c(S_q) \cdot \Delta_q$ for each q . One can imagine that value $\beta \in \mathbb{R}$ gradually increases from 0 to β° , that is, we gradually spend the budget, and, accordingly, the algorithm **Lift** gradually updates the weight vector w . In the following, we regard β as a time parameter. For any real number β with $0 \leq \beta \leq \beta^\circ$, let $S_\beta \subseteq V$ be the subset which is being lifted by the continuous version of **Lift** and $w_\beta \in \mathbb{R}^V$ be the weight vector at that moment. If $\beta_{q-1} \leq \beta < \beta_q$, we have $S_\beta = S_q$. Let $\Omega_L = \{w_\beta : 0 \leq \beta \leq \beta^\circ\}$.

We imaginarily consider another set of weight vectors $\Omega_{\text{OPT}} = \{w_\beta^* : 0 \leq \beta \leq \beta^\circ\}$ that illustrates an update consisted of iterative lifting procedures and basically mimics Ω_L . The vector w_β^* changes from w° into w^* using the budget gradually. For each integer q with $1 \leq q \leq q^*$, while β increases from β_{q-1} to β_q , the vector w_β^* changes as follows, using the budget $\beta_q - \beta_{q-1} = c(S_q) \cdot \Delta_q$.

Phase q-1:

Weights of elements in $Y := S_\beta \cap \{v \in V : w_\beta^*(v) < w^*(v)\}$ are gradually increased by the amount Δ_q . If $w_\beta^*(v)$ reaches $w^*(v)$ before it is increased by Δ_q for some $v \in Y$, set $Y := Y \setminus \{v\}$ at that moment and this phase continues.

Phase q-2:

Weights of elements in $Z = \{v \in V : w_\beta^*(v) < w^*(v)\}$ are increased using the budget $\beta_q - \beta_{q-\frac{1}{2}}$, where $\beta_{q-\frac{1}{2}} = \beta_{q-1} + (\text{the budget spent in Phase q-1})$. Again, Z might be decreased in this phase.

In **Phase q-1**, the vector w_β^* is updated in a similar way

to the update by **Lift** as long as $w_\beta^*(v)$ is not reached $w^*(v)$. In **Phase q-2**, the elements whose weights $w_\beta^*(v)$ are not reached $w^*(v)$ are equally raised. Starting with $w_0^* = w^\circ$, it is easy to see that we really have $w_{\beta^\circ}^* = w^*$. For any β with $0 \leq \beta \leq \beta^\circ$, let $S_\beta^* \subseteq V$ be the subset which is being lifted at the moment β .

Functions F_L, F^* defined by $F_L(\beta) = \varphi(w_\beta)$ ($0 \leq \beta \leq \beta^\circ$), $F^*(\beta) = \varphi(w_\beta^*)$ ($0 \leq \beta \leq \beta^\circ$) are both continuous. For almost all $\beta \in (0, \beta^\circ)$, we have $\frac{d}{d\beta} F_L(\beta) = \text{eff}(S_\beta, f, w_\beta)$ and $\frac{d}{d\beta} F^*(\beta) = \text{eff}(S_\beta^*, f, w_\beta^*)$. Moreover, $F_L(0) = F^*(0) = 0$ and $F^*(\beta^\circ) = F(\beta^\circ)$. So, if we show that the inequality $\text{eff}(S_\beta, f, w_\beta) \geq \text{eff}(S_\beta^*, f, w_\beta^*)$ for $\beta \in (0, \beta^\circ)$, the proof will be completed. Fix $\beta \in (0, \beta^\circ)$ and assume $\beta_{q-1} \leq \beta < \beta_q$. Let $v \in S_\beta^*$ and consider the value of $w_\beta^*(v)$. By definition, $w_\beta^*(v) < w^*(v)$. Thus, during **Phase q'-1** ($1 \leq q' \leq q-1$), the update of the weight of v is similar to that of **Lift**. During **Phase q'-2** ($1 \leq q' \leq q-1$), the weight of v has always been increased. From these facts, whether $\beta_{q-1} \leq \beta < \beta_{q-\frac{1}{2}}$ or $\beta_{q-\frac{1}{2}} \leq \beta < \beta_q$, one can see that the relation $w_\beta(v') \leq w_\beta(v)$ implies $w_\beta^*(v') \leq w_\beta^*(v)$ for each $v' \in V$. In Lemma 17, setting $w' = w_\beta$, $w = w_\beta^*$ and $X = S_\beta^*$, we obtain $f_{w_\beta}^\#(S_\beta^*) \geq f_{w_\beta^*}^\#(S_\beta^*)$ and so $\text{eff}(S_\beta^*, f, w_\beta) = f_{w_\beta}^\#(S_\beta^*)/c(S_\beta^*) \geq f_{w_\beta^*}^\#(S_\beta^*)/c(S_\beta^*) = \text{eff}(S_\beta^*, f, w_\beta^*)$. As the algorithm **Lift** always chooses the subset of largest efficiency, we have $\text{eff}(S_\beta, f, w_\beta) \geq \text{eff}(S_\beta^*, f, w_\beta)$. Therefore, we obtain $\text{eff}(S_\beta, f, w_\beta) \geq \text{eff}(S_\beta^*, f, w_\beta^*)$. \square

Acknowledgments

I thank Satoru Iwata for useful comments.

References

- [1] R. E. Bixby, W. H. Cunningham and D. M. Topkis: The partial order of a polymatroid extreme point. *Mathematics of Operations Research*, **10** (1985), pp. 367–378.
- [2] W. Dinkelbach: On nonlinear fractional programming. *Management Science*, **13** (1967), pp. 492–498.
- [3] J. Edmonds: Submodular functions, matroids, and certain polyhedra. In R. Guy, H. Hanai, N. Sauer, and J. Schönheim, editors, *Combinatorial Structures and Their Applications*, Gordon and Breach, New York, 1970, pp. 69–87.
- [4] L. Fleischer and S. Iwata: A push-relabel framework for submodular function minimization and applications to parametric optimization. *Discrete Applied Mathematics*, **131** (2003), pp. 311–322.
- [5] G. Frederickson and R. Solis-Oba: Algorithms for measuring perturbability in matroid optimization. *Combinatorica*, **18** (1998), pp. 503–518.
- [6] G. Frederickson and R. Solis-Oba: Increasing the weight of minimum spanning trees. *Journal of Algorithms*, **33** (1999), pp. 244–266.
- [7] S. Fujishige: Lexicographically optimal base of a polymatroid with respect to a weight vector. *Mathematics of Operations Research*, **5** (1980), pp. 186–196.
- [8] S. Fujishige: *Submodular Functions and Optimization (Second Edition)*. Elsevier, Amsterdam, 2005.
- [9] G. Gallo, M. D. Grigoriadis and R. E. Tarjan: A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, **18** (1989), pp. 30–55.
- [10] A. Goldberg and R. E. Tarjan: A new approach to the maximum-flow problem. *Journal of the ACM*, **35** (1988), pp. 721–740.
- [11] M. Grötschel, L. Lovász and A. Schrijver: *Geometric Algorithms and Combinatorial Optimization*. Springer, Berlin, 1988.
- [12] A. Hayrapetyan, C. Swamy and É. Tardos: Network design for information networks. *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms* (2005), pp. 933–942.
- [13] S. Iwata, L. Fleischer, and S. Fujishige: A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM*, **48** (2001), pp. 761–777.
- [14] S. Iwata, K. Murota and M. Shigeno: A fast parametric submodular intersection algorithm for strong map sequences. *Mathematics of Operations Research*, **22** (1997), pp. 803–813.
- [15] K. Jain and V. V. Vazirani: Eisenberg-Gale markets: algorithms and structural properties. *Proceedings of the 39th ACM Symposium on Theory of Computing* (2007), pp. 364–373.
- [16] L. Lovász: Submodular functions and convexity. *Mathematical Programming — The State of the Art* (A. Bachem, M. Grötschel, and B. Korte, eds.), Springer-Verlag, 1983, pp. 235–257.
- [17] K. Murota: *Discrete Convex Analysis*. SIAM, Philadelphia, 2003.
- [18] K. Nagano: On convex minimization over base polytopes. *Proceedings of the 12th IPCO Conference* (2007), pp. 252–266.
- [19] J. B. Orlin: A faster strongly polynomial time algorithm for submodular function minimization. *Proceedings of the 12th IPCO Conference* (2007), pp. 240–251.
- [20] A. Schrijver: A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory (B)*, **80** (2000), pp. 346–355.
- [21] Y. Sharma, C. Swamy and D. P. Williamson: Approximation algorithms for prize-collecting forest problems with submodular penalty functions. *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms* (2007), pp. 1275–1284.
- [22] D. M. Topkis: Minimizing a submodular function on a lattice. *Operations Research*, **26** (1978), pp. 305–321.