MATHEMATICAL ENGINEERING
TECHNICAL REPORTS

A Faster Parametric Submodular Function Minimization Algorithm and Applications

Kiyohito NAGANO

(Communicated by Kazuo MUROTA)

METR 2007–43 July 2007

DEPARTMENT OF MATHEMATICAL INFORMATICS
GRADUATE SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY
THE UNIVERSITY OF TOKYO
BUNKYO-KU, TOKYO 113-8656, JAPAN

WWW page: http://www.i.u-tokyo.ac.jp/edu/course/mi/index_e.shtml

A Faster Parametric Submodular Function Minimization Algorithm and Applications

Kiyohito Nagano*

Department of Mathematical Informatics,
Graduate School of Information Science and Technology,
University of Tokyo, Bunkyo-ku, Tokyo 113-8656, Japan.
kiyohito_nagano@mist.i.u-tokyo.ac.jp

July 11, 2007

Abstract

This paper considers the computation of minimizers of every function in a strong map sequence of submodular functions. We show that the recent submodular function minimization algorithm of Orlin can be extended to solve the parametric submodular function minimization problem efficiently. For each function in the sequence, by careful discussions on subsystems, our method provides a compact representation of all the minimizers. So our algorithm is still an extension of Orlin's algorithm even if the number of functions is 1. Applications include faster algorithms for minimum ratio problems and convex optimization over submodular constraints. We also show 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 ones 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 [17] of it is convex. By extending the theory of submodular functions, Murota [19] 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 [22]. More recently, Orlin [21] developed a faster strongly polynomial algorithm for SFM. The running time of Orlin's algorithm is $\mathrm{O}(n^5\mathrm{EO} + n^6)$, where n is the cardinality of the ground set and EO is the time of one function evaluation.

Gallo, Grigoriadis and Tarjan [9] extended the push/relabel maximum flow algorithm devised by Goldberg and Tarjan [10] to solve problems in an important class of parametric maximum flows efficiently. Iwata, Murota and Shigeno [14] extended the result in [9] to solve polymatroid intersection. Let $f_1, \ldots, f_k$ be submodular functions that have the same ground set V with $|V| = n$ and form a strong map sequence [14, 24]. 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 $\mathrm{O}((n^7+kn^2)\mathrm{EO}+n^8)$ time by extending their push/relabel $\mathrm{O}(n^7\mathrm{EO} + n^8)$ SFM algorithm.

*Supported by Grant-in-Aid for JSPS Fellows.

In this paper, we give a faster algorithm for parametric SFM by embedding Orlin's SFM algorithm [21] within the parametric optimization framework successfully. Since it runs in $O((n^5+kn^3)EO+n^6)$ 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. So our algorithm is still an extension of Orlin's algorithm even if $k = 1$. 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 [23]. In addition, from the framework provided by Nagano [20], 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 $O(n^5EO + n^6)$ time.

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. Jüttner [16] showed a wide class of budgeted optimization problems can be solved in strongly polynomal time. But his approach does not necessarily gives fast algorithms.

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 $f : 2^V \to \mathbb{R}$ be a submodular set function with $f(\varnothing) = 0$, 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 $g : 2^V \to \mathbb{R}$ is monotone if $g(X) \leq g(Y)$ for each $X, Y \subseteq V$ with $X \subseteq Y$, and it 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 $H' \subseteq H$.

An optimization problem 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

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

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

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

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

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

So 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., [19, Note 10.11] for details.

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

In many SFM algorithms [18], 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 $\alpha \geq 0$ 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 $\mathbf{B}(f)$. Hence, staying 0 seems comfortable for each component $x(v)$. The following property is helpful to understand the algorithm in §3, which will be used implicitly. A direct proof of Proposition 2 will be given in the appendix A.

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

From Proposition 2, it is natural to say that an update direction $x' \in \mathbb{R}^V$ is good if

$$x'(V) = 0, \quad x'(v) = 0 \text{ for } v \in V^0 \quad \text{and} \quad x'(V^+) \leq x'(V^-), \tag{3}$$

where $V^0 = \{v : x(v) = 0\}$, $V^+ = \{v : x(v) > 0\}$, and $V^- = \{v : x(v) < 0\}$.

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 $f^S : 2^S \to \mathbb{R}$ and $f_S : 2^{V \setminus S} \to \mathbb{R}$ are submodular and satisfy $f^S(\varnothing) = f_S(\varnothing) = 0$. 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 $f_2 \to f_1$ and we call this relation a strong map [14, 24]. 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 $b_t^{\prec}$ 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 $b_1^{\prec} \geq b_2^{\prec}$. Given a bunch of submodular functions that forms a strong map sequence, the parametric submodular function minimization (parametric SFM) 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 $f^{\#} : 2^V \to \mathbb{R}$, 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 submodular function minimization

We consider the efficient computation of a minimizer of every function in a strong map sequence of submodular functions $f_t : 2^{\{1,\ldots,n\}} \to \mathbb{R}$ $(t = 1, \ldots, k)$ such that $f_1 \leftarrow \cdots \leftarrow f_k$ and $f_t(\varnothing) = 0$ 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, $\min_X \{f_t(X) : X \subseteq \{1 \ldots, n\}\}$ for $t = 1, \ldots, k$, runs in $O(n^5\mathrm{EO} + n^6 + kn^3\mathrm{EO})$ time. If $k = 1$ it corresponds to Orlin's algorithm for SFM [21], which runs in $O(n^5\mathrm{EO} + n^6)$ time. Remark that our algorithm finds a compact representation of all the minimizers of each function f_t. Thus it is still an extension of the algorithm of Orlin even if $k = 1$.

Throughout this section, using some t and subset $V \subseteq \{1, \ldots, n\}$, 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 $t = 1, \ldots, k$. For a total order $\prec$ in V and $1 \le t \le k$, we denote by $b_t^{\prec}$ the extreme base of $\mathbf{B}(f_t^V)$ generated by $\prec$. We have the following relations.

Lemma 4 *$X_1^{\min} \supseteq \cdots \supseteq X_k^{\min}$, $X_1^{\max} \supseteq \cdots \supseteq X_k^{\max}$ and $b_1^{\prec} \le \cdots \le b_k^{\prec}$.*

Our algorithm **P-SFM** initializes $V := V^{\mathrm{i}} = \{1, \ldots, n\}$ and repeatedly restrict V. It minimizes $f_1, \ldots, f_k$ 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 $f = f_{t''}^{V''}$ at another iteration after that, we have $V' \supseteq V''$ and $t' \le t''$;

R2. the relation $X_t^{\min} \subseteq V$ holds while minimizing $f = f_t^V$.

Notice that the properties R1 and R2 are consistent with Lemma 4. We call a set of iterations of trying to minimize f_t a t-phase. To the algorithm in §3.3, roughly speaking, we explain inner operations of **P-SFM** in §3.1 and outer operations of **P-SFM** in §3.2.

3.1 Preparations from Orlin's result [21]

We make some preparations to give an iterative local search algorithm. They are from Orlin's paper [21]. 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 $1 \le t \le k$.

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

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

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

- a collection D of distance functions 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 $f(\varnothing) = 0$.

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 $D_{\min}(v) \le d(v) \le D_{\min}(v) + 1$ 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 $d' = \mathrm{INC}(d, v)$ be the distance function such that $d'(u) = d(v) + 1$ if $u = v$ and $d'(u) = d(u)$ otherwise. By submodularity,

$$b_{d'}(v) - b_d(v) \le 0 \quad \text{and} \quad b_{d'}(u) - b_d(u) \ge 0 \text{ if } u \in V \setminus \{v\}. \tag{4}$$

The algorithm maintains collection D of distance functions in nondecreasing 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 $s(v) \in \mathbb{Z}^V_{\geq 0}$ be $\mathrm{INC}(p(v), v)$, 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 (4), 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 (Orlin [21]) *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 $V^+ = \varnothing$, x is optimal for (2) and V is a minimizer of f bacause $x^-(V) = f(V)$. So we only consider the case where $V^+ \neq \varnothing$. To compute a good update direction $x' \in \mathbb{R}^V$, we choose some $v^* \in V^+$ and compute a vector $\widetilde{\gamma} \in \mathbb{R}^{V^0 \cup \{v^*\}}$ such that

$$\widetilde{\gamma} \neq \mathbf{0}, \quad \widetilde{\gamma} \geq \mathbf{0} \quad \text{and} \quad A^*\,\gamma + c\eta = \mathbf{0}, \tag{5}$$

where $\gamma = \widetilde{\gamma}^{V^0} \in \mathbb{R}^{V^0}$, $\eta = \widetilde{\gamma}(v^*) \in \mathbb{R}$ and $c = (c_{v^*}(v) : v \in V^0)(\geq \mathbf{0})$. 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 $\widetilde{\gamma}$ satisfying (5) can be computed in $\mathrm{O}(|V^0|^3) = \mathrm{O}(n^3)$ time via Gaussian elimination. Then we set the update direction

$$x' := \sum_{v \in V^0 \cup \{v^*\}} \widetilde{\gamma}(v) \cdot c_v = \sum_{v \in V^0 \cup \{v^*\}} \widetilde{\gamma}(v) \cdot (b_{s(v)} - b_{p(v)}) \in \mathbb{R}^V.$$

The vector x' satisfies (3) because we have that $x'(V) = 0$, $x'(v) = 0$ if $v \in V^0$, $x'(v) \leq 0$ if $v = v^*$ and $x'(v) \geq 0$ if $v \in V \setminus (V^0 \cup \{v^*\})$. The algorithm updates $x := x + \alpha x'$ where

$$\begin{aligned} &\alpha := \min\{\alpha_1, \alpha_2\}, \\ &\alpha_1 := \max\{\alpha \in \mathbb{R} : \ x(v) + \alpha x'(v) \geq 0 \ \forall v \in V^+; \ \ x(v) + \alpha x'(v) \leq 0 \ \forall v \in V^-\}, \\ &\alpha_2 := \max\{\alpha \in \mathbb{R} : \ \alpha \textstyle\sum_v \{\widetilde{\gamma}(v) : v \text{ with } p(v) = d\} \leq \lambda(d) \ \ \forall d \in D\}, \end{aligned} \tag{6}$$

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:

$$\begin{aligned} D &:= D \cup \{s(v) : v \in V^0 \cup \{v^*\}\}, \\ \lambda(d) &:= \lambda(d) + \alpha \textstyle\sum_v \{\widetilde{\gamma}(v) : v \text{ with } s(v) = d\} \\ &\qquad - \alpha \textstyle\sum_v \{\widetilde{\gamma}(v) : v \text{ with } p(v) = d\}, \quad \forall d \in D. \end{aligned} \tag{7}$$

As x' is a good update direction, if v is in V^0 at some iteration of t-phase, v is 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 $\mathrm{O}(qn^2)$ time at the next iteration, which is faster than $\mathrm{O}(n^3)$.

3.2 Transitions, restrictions and subsystems

Next let us see transitions of functions and restrictions of the ground set V in our algorithm **P-SFM**. For each $t = 1, \ldots, k$, our algorithm divides the submodular system (V^{i}, f_t) into small subsystems. For distance functions $D \subseteq \mathbb{Z}^V_{\geq 0}$ 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 $\widetilde{x} \in \mathbf{B}(f_{t+1}^V)$ generated by (D, λ, f_{t+1}^V), and 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, we have $x \leq \widetilde{x}$. Thus V1 is also satisfied. □

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 $\max_x\{x^-(V^{\mathrm{i}}) : x \in \mathbf{B}(f_t)\}$ 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 [21]) *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 $X \subseteq V$, $f(X) \geq f(X \cap S) + f(X \cup S) - f(S) \geq f(X \cap S) + x(X \cup S) - x(S) \geq f(X \cap S)$. Thus we complete the proof. □

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

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 $x(R_D) = f(R_D)$. Therefore, by Lemma 7, there exists a minimizer X of f with $X \subseteq R_D$. Observe that R_D can be computed in $\mathrm{O}(n^2)$ 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 $T^{(0)} = V^{\mathrm{i}}$. Inductively, we have $U^{(r)} = T^{(r-1)} \setminus T^{(r)}$ and V^{i} is equal to dijoint union $V \cup U^{(r)} \cup \cdots \cup U^{(1)}$. Moreover, define function $f_{t'}^{(r)} : 2^{U^{(r)}} \to \mathbb{R}$ by $f_{t'}^{(r)} = (f_{t'}^{T^{(r-1)}})_{T^{(r)}}$ for each $t' = t^{(r)}, \ldots, k$. That is,

$$f_{t'}^{(r)}(X) = f_{t'}(X \cup T^{(r)}) - f_{t'}(T^{(r)}), \quad \forall X \subseteq U^{(r)} \big(= T^{(r-1)} \setminus T^{(r)}\big).$$

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 $t' = t^{(r)}, \ldots, k$.

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

PROOF: We have $\mathcal{S}^{(r)} = (U^{(r)}, f_{t^{(r)}}^{(r)})$ and $x_{t^{(r)}}^{(r)} \geq 0$. From the relation $f_{t^{(r)}}^{(r)} \leftarrow \cdots \leftarrow f_k^{(r)}$ and Lemma 4, it follows that $0 \leq x_{t^{(r)}}^{(r)} \leq \ldots \leq x_k^{(r)}$. □

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

3.3 The algorithm P-SFM

We describe the algorithm for parametric SFM, $\min_X\{f_t(X) : X \subseteq V^{\rm i}\}$ for each $t = 1, \ldots, k$. 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 $\lambda' \in \mathbb{R}^{D'}_{>0}$ 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 $Z_t \subseteq V^{\rm i}$ and the vector $x_t \in \mathbb{R}^{V^{\rm i}}$ for each $t = 1, \ldots, k$.

Algorithm P-SFM
Initialization :
$V := V^{\rm i}$; $d_\circ := \mathbf{0}$; $D := \{d_\circ\}$; $\lambda(d_\circ) := 1$; $t := 1$; $r := 0$; $T^{(0)} = V^{\rm i}$;
while $t \leq k$ **begin**
  $f := f^V_t$; $x := \sum_{d \in D} \lambda(d) \cdot b_d \in \mathbf{B}(f^V_t)$;
  while $V^+ \neq \varnothing$ **begin**
    Choose $v^* \in V^+$ and let $c = (c_{v^*}(v) : v \in V^0)$;
    Compute $\widetilde{\gamma} \in \mathbb{R}^{V^0 \cup \{v^*\}}$ s. t. (5) and set $x' := \sum_{v \in V^0 \cup \{v^*\}} \widetilde{\gamma}(v) \cdot c_v$;
    Set $x := x + \alpha x'$, where α is defined in (6), and update D, λ as in (7);
    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**$(D^{(r)}, \lambda^{(r)}, f^{(r)}_t)$;
    end ;
  end;
  Return $Z_t = V$ and $x_t = x \oplus x^{(r)}_t \oplus \cdots \oplus x^{(2)}_t \oplus x^{(1)}_t$;
  $t := t + 1$;
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^V_t, $X^{\min}_t \subseteq V$;

(10.d) $V^{\rm i}$ *is equal to disjoint union* $V \cup U^{(1)} \cup \cdots \cup U^{(r)}$.

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 begining of $t+1$-phase, by Lemma 6, (10.a) is satisfied and, by Lemma 4, $X^{\min}_{t+1} \subseteq X^{\min}_t \subseteq V$ 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, $X^{\min}_t \subseteq R_D$ holds if $X^{\min}_t \subseteq V$ 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. □

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

PROOF: At the end of each t-phase, we have $V^+ = \varnothing$, 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 $\max_x\{x^-(V^{\mathrm{i}}) : x \in \mathbf{B}(f_t)\}$. □

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 [21]. To make the paper self-contained, the proof will be given in Appendix B.

Lemma 12 (also see [21]) *Throughout the algorithm* **P-SFM**, *the number of distance functions added to D and the number of columns added to A^* are both* $\mathrm{O}(n^4)$.

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 $\mathrm{O}(n^4)$. So, the number of iterations of **P-SFM** is $\mathrm{O}(n^4 + kn)$.

Theorem 13 *The algorithm* **P-SFM** *runs in* $\mathrm{O}(n^5\mathrm{EO} + n^6 + k(n^2\mathrm{EO} + n^3))$ *time.*

PROOF: Each computation of the initial base x of a t-phase takes $\mathrm{O}(n^2\mathrm{EO})$ 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 $\mathrm{O}(n^2\mathrm{EO} + n^3)$ time to construct A^* and solve the system of equations on A^*. Each column can be computed in $\mathrm{O}(n\mathrm{EO})$ time and it takes $\mathrm{O}(n^2)$ 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 $(n^5\mathrm{EO} + n^6 + k(n^2\mathrm{EO} + n^3))$ time.

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

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

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* $\mathrm{O}((n^5 + kn^3)\mathrm{EO} + n^6)$ *time in total.*

3.4 Parametric minimization in the reverse order

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

4 Applications

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

4.1 Minimum ratio problems and convex optimization

Minimum ratio problems. Let $f : 2^V \to \mathbb{R}$ be a submodular function with $f = (\varnothing)$ and $w \in \mathbb{R}^V_{\geq 0}$ be a nonnegative weight vector with $w \neq \mathbf{0}$. 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 $t_1 > \cdots > t_k \geq 0$, we have to minimize k submodular functions $f - t_1 w, \ldots, f - t_k w$ in this order. As we have the relation $f - t_1 w \leftarrow \cdots \leftarrow f - t_k w$, the ratio $f(X)/w(X)$ can be minimized in $\mathrm{O}(|V|^5 \mathrm{EO} + |V|^6)$ 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 $g : 2^V \to \mathbb{R}$ 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 $t_1 > \cdots > t_k \geq 0$, we have the relation $f - t_1 g \leftarrow \cdots \leftarrow f - t_k g$. Therefore, the problem can also be solved in $\mathrm{O}(|V|^5 \mathrm{EO} + |V|^6)$ time.

Convex optimization over base polytopes. In [20], 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 Nagano's framework, with the aid of the algorithms **P-SFM** and $\mathbf{P_R}$**-SFM**, each of them can be solved in the same order as a single implementation of **P-SFM**.

4.2 Measuring robustness in submodular optimization

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 $f : 2^V \to \mathbb{R}$ with $f(\varnothing) = 0$, 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

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

We call $F : \mathbb{R}_{\geq 0} \to \mathbb{R}$ 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 $w_1 < \cdots < w_\ell$ 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 $L_0 = \varnothing$. For $i = 1, \ldots, \ell$, 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 $f^i : 2^{K_i} \to \mathbb{R}$ by $f^i = (f^{L_i})_{L_{i-1}}$ for each i. Function $f_w : 2^V \to \mathbb{R}$ defined by

$$f_w(X) = \sum_{i=1}^{\ell} f^i(X \cap K_i) \left(= \sum_{i=1}^{\ell} \{f((X \cap L_i) \cup L_{i-1}) - f(L_{i-1})\} \right)$$

for each $X \subseteq V$ is submodular. We have $\mathbf{B}(f_w) = \mathbf{B}(f^1) \oplus \cdots \oplus \mathbf{B}(f^\ell)$. 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

$$\begin{aligned} &\mathrm{tol}(X, w) = \min_{v \in X} \mathrm{tol}(v, w), \\ &\qquad \text{where } \mathrm{tol}(v, w) = w_{i+1} - w_i \text{ for } v \in V \text{ with } w(v) = w_i. \end{aligned}$$

Let Δ be a number such that $0 \leq \Delta \leq \mathrm{tol}(X, w)$ and let $w_\Delta = w + \Delta \cdot \chi_X$. By the definition of $\mathrm{tol}(v, w)$, if $\Delta \neq \mathrm{tol}(v, w)$, 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 $\mathrm{eff}(X, f, w) = \frac{f_w^\#(X)}{c(X)} \left(= \frac{\varphi(w_\Delta) - \varphi(w)}{\Delta \cdot c(X)} \right)$. 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

$$f_w^\#(X) = \sum_{i=1}^{\ell} f^{i\#}(X \cap K_i) \left(= \sum_{i=1}^{\ell} \{ f(L_i) - f((L_i \setminus X) \cup L_{i-1}) \} \right) \tag{8}$$

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 $X \cap K_i \neq \varnothing$. Then we have $\mathrm{eff}(X \cap K_i, f, w) = \mathrm{eff}(X, f, w)$. Furthermore, $\mathrm{eff}(Y, f, w) = \mathrm{eff}(X, f, w)$ for subset $Y \subseteq K_i$ that maximizes $f^{i\#}(Y)/c(Y)$ over 2^{K_i}.*

Proof: Let $X_i = X \cap K_i$ for $i = 1, \ldots, \ell$ and let $I = \{i : X_i \neq \varnothing\}$. We have $\mathrm{eff}(X, f, w) = f_w^\#(X)/c(X) = \sum_{i \in I} f^{i\#}(X_i) / \sum_{i \in I} c(X_i) \leq \max_{i \in I} f^{i\#}(X_i)/c(X_i)$. In addition, for $Y \subseteq K_i$, $f^{i\#}(Y)/c(Y) = f_w^\#(Y)/c(Y) = \mathrm{eff}(Y, f, w)$. Thus, for each $i \in I$, $\mathrm{eff}(X_i, f, w) = \mathrm{eff}(X, f, w)$ and X_i maximizes $f^{i\#}(Y)/c(Y)$ over 2^{K_i}. Moreover, if $i \in I$ and subset $Y \subseteq K_i$ maximizes $f^{i\#}(Y)/c(Y)$ over 2^{K_i}, we have $\mathrm{eff}(Y) = \mathrm{eff}(X)$. □

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 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 $\frac{f^{i\#}(X^i)}{c(X^i)}$ maximum for each $i = 1, \ldots, \ell$;
    Set $X := X^{i^*}$, where X^{i^*} is a subset with $\mathrm{eff}(X^{i^*}, f, w) = \max_{1 \leq i \leq \ell} \mathrm{eff}(X^i, f, w)$;
    Set $w := w + \Delta \cdot \chi_X$, where $\Delta = \min\{\mathrm{tol}(X, w), \frac{\beta^\circ - \beta}{c(X)}\}$;
    Set $\beta := \beta + \Delta \cdot c(X)$;
  end;
  Return $w_{\mathbf{L}} := w$;

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

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

Proof: Consider any iteration of **Lift** other than the last one. Since elements in X have the same weight, one can inductively show that $w(v) \in \{w_1^\circ, \ldots, w_{\ell^\circ}^\circ\}$ for each $v \in V$. Define $\Gamma = \sum_{i=1}^{\ell^\circ} |K_i^\circ|(\ell^\circ - i)$, where $K_i^\circ = \{v \in V : w(v) = w_i^\circ\}$. The update $w := w + \Delta \cdot \chi_X$ changes the weight of $v \in X$ from w_i° to $w_{i'}^\circ$ for some i, i' with $i < i'$. So, Γ is decreased by at least 1 in each iteration. Initially, $\Gamma \leq |V|(\ell^\circ - 1)$. Thus, the number of iterations is at most $|V|(\ell^\circ - 1) + 1 \leq |V| \cdot \ell^\circ$. □

The correctness of Lift. Let $p^* \in \mathbb{R}^V$ be an optimal solution to $\max_p \{\varphi(w^\circ + p) : \langle c, p \rangle \leq \beta;\ p \geq \mathbf{0}\}$ and let $w^* = w^\circ + p^*$. We will show that the algorithm **Lift** computes $F(\beta^\circ)$ correctly, i.e., the equality $\varphi(w_{\mathbf{L}}) = \varphi(w^*)$ 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 (8), one can observe that

$$f^{\#}_w(X) = \sum_{v \in X}\{f(X_v) - f(X_v \setminus \{v\})\}, \quad f^{\#}_{w'}(X) = \sum_{v \in X}\{f(X'_v) - f(X'_v \setminus \{v\})\}.$$

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 we have $f^{\#}_w(X) \leq f^{\#}_{w'}(X)$. □

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, $0 = \beta_0 < \cdots < \beta_{q^*} = \beta^\circ$ 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 $\Omega_{\mathbf{L}} = \{w_\beta : 0 \leq \beta \leq \beta^\circ\}$.

We imaginarily consider another set of weight vectors $\Omega_{\mathrm{OPT}} = \{w^*_\beta : 0 \leq \beta \leq \beta^\circ\}$ that illustrates an update consisted of iterative lifting procedures and basically mimics $\Omega_{\mathbf{L}}$. 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: First, the 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: Next, the 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 $\beta_{q-\frac{1}{2}} = \beta_{q-1} + (\text{the budget spent in Phase } q\text{-1})$. Again, Z might be decreased in the execution of this phase.

In **Phase** q-1, the vector w^*_β is updated in a similar way to the update by **Lift** as long as $v \in V$ is not reached $w^*(v)$. In **Phase** q-2, the elements that are not reached $w^*(v)$ are equally raised. Starting with $w^*_0 = w^\circ$, it is easy to see that $w^*_{\beta^\circ} = w^*$. 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 $F_{\mathbf{L}}$, F^* defined by $F_{\mathbf{L}}(\beta) = \varphi(w_\beta)$ ($0 \leq \beta \leq \beta^\circ$), $F^*(\beta) = \varphi(w^*_\beta)$ ($0 \leq \beta \leq \beta^\circ$) are both continuous. For almost all $\beta \in (0, \beta^\circ)$, we have $\frac{\mathrm{d}}{\mathrm{d}\beta}F_{\mathbf{L}}(\beta) = \mathrm{eff}(S_\beta, f, w_\beta)$ and $\frac{\mathrm{d}}{\mathrm{d}\beta}F^*(\beta) = \mathrm{eff}(S^*_\beta, f, w^*_\beta)$. Moreover, $F_{\mathbf{L}}(0) = F^*(0) = 0$ and $F^*(\beta^\circ) = F(\beta^\circ)$. So, if we show the inequality $\mathrm{eff}(S_\beta, f, w_\beta) \geq \mathrm{eff}(S^*_\beta, f, w^*_\beta)$ 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 $v' \in V$. By 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 $\mathrm{eff}(S^*_\beta, f, w_\beta) = f^{\#}_{w_\beta}(S^*_\beta)/c(S^*_\beta) \geq f^{\#}_{w^*_\beta}(S^*_\beta)/c(S^*_\beta) = \mathrm{eff}(S^*_\beta, f, w^*_\beta)$. As the algorithm **Lift** always chooses the subset of largest efficiency, we have $\mathrm{eff}(S_\beta, f, w_\beta) \geq \mathrm{eff}(S^*_\beta, f, w_\beta)$. Therefore, we obtain $\mathrm{eff}(S_\beta, f, w_\beta) \geq \mathrm{eff}(S^*_\beta, f, w^*_\beta)$. □

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] A. Jüttner: On budgeted optimization problems. *SIAM Journal on Discrete Mathematics*, **20** (2006), pp. 880–892.

[17] 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.

[18] S. T. McCormick: Submodular function minimization. In K. Aardal, G. L. Nemhauser, and R. Weismantel, editors, *Discrete Optimization* (Handbooks in Operations Research and Management Science 12), Elsevier, Amsterdam, 2005, Chapter 7, pp. 321–391.

[19] K. Murota: *Discrete Convex Analysis*. SIAM, Philadelphia, 2003.

[20] K. Nagano: On convex minimization over base polytopes. *Proceedings of the 12th IPCO Conference* (2007), pp. 252–266.

[21] J. B. Orlin: A faster strongly polynomial time algorithm for submodular function minimization. *Proceedings of the 12th IPCO Conference* (2007), pp. 240–251.

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

[23] 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.

[24] D. M. Topkis: Minimizing a submodular function on a lattice. *Operations Research*, **26** (1978), pp. 305–321.

Appendix A. A direct proof of Proposition 2

Orlin's algorithm [21] gives a proof of Proposition 2. Here we give its non-algorithmic proof.

PROOF OF PROPOSITION 2. Let $V^0 = \{v \in V : x(v) = 0\}$. Take an optimal solution y to (2) that minimizes $\Gamma(y) = \sum_{v \in V^0} (y(v) - x(v))^2$. If $\Gamma(y) = 0$, y is the base we needed. To the contrary, assume $\Gamma(y) > 0$. Then, for some $v \in V^0$, $x(v) \neq y(v)$. First, we assume $x(v) = 0 > y(v)$. Now there is an element u such that $x(u) < y(u)$ and $y' = y + \varepsilon(e_v - e_u) \in \mathbf{B}(f)$ for some $\varepsilon > 0$, where e_v is the unit vector that has value 1 on v and 0 elsewhere. Then, there is a base y'' in line segment $[y, y']$ such that y'' is optimal for (2) and $\Gamma(y) > \Gamma(y'')$, a contradiction to the definition of y. We also have a contradiction if $x(v) = 0 < y(v)$. □

Appendix B. The number of columns added to A^*

To estimate the number of columns added to A^* throughout the algorithm **P-SFM**, the analysis that is basically the same as that of Orlin [21] for SFM works although we are dealing with a bunch of submodular functions. In order to make this paper self-contained, we will see the proof of Lemma 12.

For $v \in V$, let $h(v) = \sum_{u \in V} d(u)$ and $\widehat{h}(v) = \sum_{u \in V} (d(u) - D_{\min}(u))$, where $d = p(v)$. By the validity of D, we have $0 \leq \widehat{h}(v) \leq |V| \leq n$. We denote the pair $(D_{\min}(v), h(v))$ by D-Pair(v). Let $H(v)$ be the distance functions $\subseteq D$ that could have been chosen as $p(v)$, i.e., $H(v) = \{d \in D : d(V) = h(v),\ D_{\min}(v) = d(v)\}$. Easily we have

$$s(v) \notin H(u), \quad \forall u, v \in V.$$

Therefore, if a saturating update does not change D-Pair(v), any distance function will not be added to $H(v)$. Define the potential function $\Phi = \sum_{v \in V} |H(v)|$. We utilize this function Φ to estimate the number of the columns added to A^*.

The total decrease (resp. increase) in Φ is the sum of the decreases (resp. increases) in Φ over all iteration of the algorithm. Let us see that the numbers of distance functions added to D is at most the total decrease in Φ.

For example, consider some iteration in which we first have $|H(v_1)| = 1$, $|H(v_2)| = 2$, $|H(v_3)| = 3$ and, due to a saturating update, a single distance function d such that $d = p(v)$ only for $v = v_1, v_2, v_3$ is removed from D. As a result of this deletion, D-Pair(v_1) changes, D-Pair(v_2), D-Pair(v_3) do not change and we assume that $|H(v_1)|$, $|H(v_2)|$, $|H(v_3)|$ become 7, 1, 2 respectively. Then, we say the decreases in $|H(v_1)|$, $|H(v_2)|$, $|H(v_3)|$ are 1, 1, 1 and the increases in $|H(v_1)|$, $|H(v_2)|$, $|H(v_3)|$ are 7, 0, 0. Thus we say the decrease and the increase in Φ by this saturating update are 3 and 7. Furthermore, each secondary distance function $s(v_j)$ $(j = 1, 2, 3)$ would be added to D if $s(v_j) \notin D$. Hence distance

functions added to D due to the saturating update is at most the decrease in Φ in this case. One can see that this is true in general.

Next, consider the execution of Procedure Reduce. For some v, $H(v)$ might be decreased, or, $H(v)$ might become empty and accordingly D-Pair(v) would change. In either case, distance functions added to D is at most the decrease in Φ. Restrictions of V also cause the decrease in Φ, but they do not add distance functions to D.

Therefore, the numbers of distance functions added to D and the number of columns added to A^* throughout the algorithm **P-SFM** are both at most the total decrease in Φ.

Lemma 19 *For each $v \in V^{\mathrm{i}} = \{1, \ldots, n\}$,* D-Pair$(v) = (D_{\min}(v), h(v))$ *changes* $\mathrm{O}(n^2)$ *times during the execution of the algorithm* P-SFM.

Proof: The number of the restrictions of V is at most n and, for each $u \in V^{\mathrm{i}}$, $D_{\min}(u)$ is nondecreasing and at most n. Moreover $\widehat{h}(v) - h(v) = \sum_{u \in V} D_{\min}(u)$. Thus, if we show the number of changes in $\widehat{h}(v)$ is $\mathrm{O}(n^2)$, it follows that the number of changes of D-Pair(v) is also $\mathrm{O}(n^2)$.

Consider the set of iterations in which $D_{\min}(v) = \ell$ is fixed and we call it an ℓ-stage. Clearly $h(v)$ is nondecreasing in each ℓ-stage. So, $\widehat{h}(v)$ decreases by at most 1 if $D_{\min}(u)$ increases for some u, and all other changes in $\widehat{h}(v)$ are increases. Let $C(v, \ell)$ be the number of increases in $D_{\min}(u)$ for some $u \in V$. Since $0 \leq \widehat{h}(v) \leq n$, the value $\widehat{h}(v)$ changes at most $n + 2 \cdot C(v, \ell)$ times in each ℓ-stage. As $\sum_{\ell} C(v, \ell) = \mathrm{O}(n^2)$ and the number of stages is at most n, $\widehat{h}(v)$ changes $\mathrm{O}(n^2)$ times throughout P-SFM. □

Lemma 20 *The total increase and the total decrease in Φ throughout* P-SFM *are both* $\mathrm{O}(n^4)$.

Proof: Since $\Phi = \mathrm{O}(n^2)$, it suffices to show the total increase in Φ is $\mathrm{O}(n^4)$. The potential function Φ can be increased only when the pair D-Pair(v) changes for some $v \in V$. Each change of D-Pair(v) can increase $|H(v)|$ by at most $|D| = \mathrm{O}(n)$. By Lemma 19, the total increase in Φ is $\mathrm{O}(n^4)$. □

We are ready to show Lemma 12.

Proof of Lemma 12: The number of distance functions added to D and the number of columns added to A^* during the algorithm P-SFM are both at most the total decrease in Φ. By Lemma 20, it is $\mathrm{O}(n^4)$. □