Bisubmodular Function Minimization

Satoru Fujishige [*] Satoru Iwata [†]

October 2000

Abstract

This paper presents the first combinatorial, polynomial-time algorithm for minimizing bisubmodular functions, extending the scaling algorithm for submodular function minimization due to Iwata, Fleischer, and Fujishige.

A bisubmodular function arises as a rank function of a delta-matroid. The scaling algorithm naturally leads to the first combinatorial polynomial-time algorithm for testing membership in delta-matroid polyhedra. Unlike the case of matroid polyhedra, it remains open to develop a combinatorial strongly polynomial algorithm for this problem.

[*]Division of Systems Science, Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka 560-8531, Japan (fujishig@sys.es.osaka-u.ac.jp). Research partly carried out while at Forschungsinstut für Diskrete Mathematik, Universität Bonn.

[†]Department of Mathematical Engineering and Information Physics, University of Tokyo, Tokyo 113-8656, Japan (iwata@sr3.t.u-tokyo.ac.jp).

1 Introduction

Let V be a finite nonempty set of cardinality n and 3^V denote the set of ordered pairs of disjoint subsets of V. Two binary operations $\sqcup$ and $\sqcap$ on 3^V are defined by

$$\begin{aligned}(X_1, Y_1) \sqcup (X_2, Y_2) &= ((X_1 \cup X_2)\backslash(Y_1 \cup Y_2), (Y_1 \cup Y_2)\backslash(X_1 \cup X_2)) \\ (X_1, Y_1) \sqcap (X_2, Y_2) &= (X_1 \cap X_2, Y_1 \cap Y_2).\end{aligned}$$

A function $f : 3^V \to \mathbf{R}$ is called *bisubmodular* if it satisfies

$$f(X_1, Y_1) + f(X_2, Y_2) \geq f((X_1, Y_1) \sqcup (X_2, Y_2)) + f((X_1, Y_1) \sqcap (X_2, Y_2))$$

for any (X_1, Y_1) and (X_2, Y_2) in 3^V. This paper presents the first combinatorial polynomial-time algorithm for minimizing bisubmodular functions.

A bisubmodular function generalizes a submodular function as follows. Let 2^V denote the family of all the subsets of V. A function $g : 2^V \to \mathbf{R}$ is called *submodular* if it satisfies

$$g(Z_1) + g(Z_2) \geq g(Z_1 \cup Z_2) + g(Z_1 \cap Z_2)$$

for any $Z_1, Z_2 \subseteq V$. For a submodular function g, we define a bisubmodular function $f : 3^V \to \mathbf{R}$ by

$$f(X, Y) = g(X) + g(V\backslash Y) - g(V).$$

If (X, Y) is a minimizer of f, then both X and $V\backslash Y$ are minimizers of g. Thus, bisubmodular function minimization is a generalization of submodular function minimization.

The first polynomial-time algorithm for submodular function minimization is due to Grötschel–Lovász–Schrijver [16]. They also give the first strongly polynomial algorithms in [17]. Their algorithms rely on the ellipsoid method, which is not efficient in practice. Recently, two combinatorial strongly polynomial algorithms are devised independently by Schrijver [22] and Iwata–Fleischer–Fujishige [18]. Both of these new algorithms are based on a combinatorial pseudopolynomial-time algorithm of Cunningham [8]. The algorithm of Schrijver [22] directly achieves the strongly polynomial bound, whereas Iwata–Fleischer–Fujishige [18] develop a scaling algorithm with weakly polynomial time complexity and then convert it to a strongly polynomial one.

In the present paper, we extend the scaling algorithm of Iwata–Fleischer–Fujishige [18] to solve the minimization problem for integer-valued bisubmodular functions. The resulting algorithm runs in $O(n^5 \log M)$ time, where M designates the maximum value of f. This bound is weakly polynomial, and it remains open to develop a combinatorial strongly polynomial algorithm.

2 Delta-Matroids

A bisubmodular function arises as a rank function of a delta-matroid introduced independently by Bouchet [3] and Chandrasekaran–Kabadi [6]. A delta-matroid is a set system $(V, \mathcal{F})$ with $\mathcal{F}$ being a nonempty family of subsets of V that satisfies the following exchange property:

$$\forall F_1, F_2 \in \mathcal{F}, \forall v \in F_1 \triangle F_2, \exists u \in F_1 \triangle F_2 : F_1 \triangle \{u, v\} \in \mathcal{F},$$

where $\triangle$ denotes the symmetric difference. A slightly restricted set system with an additional condition $\emptyset \in \mathcal{F}$ had been introduced by Dress–Havel [11]. A member of $\mathcal{F}$ is called a feasible set of the delta-matroid. Note that the base and the independent-set families of a matroid saitisfy this exchange property. Thus, a delta-matroid is a generalization of a matroid.

Chandrasekaran–Kabadi [6] showed that the rank function $\varrho : 3^V \to \mathbf{Z}$ defined by

$$\varrho(X, Y) = \max\{|X \cap F| - |Y \cap F| \mid F \in \mathcal{F}\}$$

is bisubmodular. The convex hull of the characteristic vectors of the feasible sets is described by

$$\mathrm{P}(\varrho) = \{x \mid x \in \mathbf{R}^V, \forall (X, Y) \in 3^V : x(X) - x(Y) \leq \varrho(X, Y)\},$$

which is called the delta-matroid polyhedron. This fact follows from the greedy algorithm [3, 6] for optimizing a linear function over the feasible sets.

Given a vector $x \in \mathbf{R}^V$, one can test if x belongs to $\mathrm{P}(\varrho)$ by minimizing a bisubmodular function $f(X, Y) = \varrho(X, Y) - x(X) + x(Y)$. Even for such a special case of bisubmodular function minimization, no combinatorial algorithm was known to run in polynomial time. This is in contrast with the matroid polyhedron, for which Cunningham [7] devised a combinatorial strongly polynomial algorithm for testing membership.

A simple example of a delta-matroid is a matching delta-matroid [4], whose feasible sets are the perfectly matchable vertex subsets of an undirected graph. The delta-matroid polyhedron is the matchable set polytope [2]. For this special case, Cunningham–Green-Krótki [10] developed an augmenting path algorithm for solving the separation problem in polynomial time with the aid of the scaling technique.

3 Bisubmodular Polyhedra

As a generalization of the delta-matroid polyhedron, a *bisubmodular polyhedron*

$$\mathrm{P}(f) = \{x \mid x \in \mathbf{R}^V, \forall (X, Y) \in 3^V : x(X) - x(Y) \leq f(X, Y)\}$$

is associated with a general bisubmodular function $f : 3^V \to \mathbf{R}$, where we assume $f(\emptyset, \emptyset) = 0$. For a vector $x \in \mathbf{R}^V$, we denote $\|x\| = \sum_{v \in V} |x(v)|$. The following min-max relation characterizes the minimum value of f.

Theorem 3.1 ([15]) *For any bisubmodular function f,*

$$\min\{f(X,Y) \mid (X,Y) \in 3^V\} = \max\{-\|x\| \mid x \in \mathrm{P}(f)\}.$$

■

The linear optimization problem over the bisubmodular polyhedron can be solved by the following greedy algorithm, which was first introduced by Dunstan–Welsh [12].

Let $\sigma : V \to \{+, -\}$ be a sign function. For any subset $U \subseteq V$, we denote by $U|\sigma$ the pair $(X,Y) \in 3^V$ with $X = \{u \mid u \in U, \sigma(u) = +\}$ and $Y = \{u \mid u \in U, \sigma(u) = -\}$. We also write $f(U|\sigma) = f(X,Y)$ for any function $f : 3^V \to \mathbf{R}$, and $x(U|\sigma) = x(X) - x(Y)$ for any vector $x \in \mathbf{R}^V$.

Let $L = (v_1, \cdots, v_n)$ be a linear ordering of V. For each $j = 1, \cdots, n$, let $L(v_j) = \{v_1, \cdots, v_j\}$. The *greedy algorithm* with respect to L and a sign function σ assigns $y(v) := \sigma(v)\{f(L(v)|\sigma) - f(L(v)\backslash\{v\}|\sigma)\}$ for each $v \in V$. Then the resulting vector $y \in \mathbf{R}^V$ is an extreme point of the bisubmodular polyhedron $\mathrm{P}(f)$.

Given a linear ordering $L = (v_1, \cdots, v_n)$ and a sign function σ, for a weight function $w : V \to \mathbf{R}$ that satisfies $|w(v_1)| \geq \cdots \geq |w(v_n)|$ and $w(v) = \sigma(v)|w(v)|$ for each $v \in V$, the vector y generated by the greedy algorithm with respect to L and σ maximizes the linear function $\sum_{v \in V} w(v)y(v)$ over the bisubmodular polyhedron $\mathrm{P}(f)$. See [14, §3.5 (b)] for a survey on bisubmodular polyhedron including the validity of the greedy algorithm (also see [1]).

Based on the validity of this greedy algorithm, Qi [21] established a connection between bisubmodular functions and their convex extensions. This is a generalization of a result of Lovász [19] on submodular functions, and it leads to a polynomial-time algorithm for bisubmodular function minimization using the ellipsoid method.

The concept of bisubmodular polyhedron is extended to that of jump system by Bouchet–Cunningham [5]. A jump system is a set of lattice points satisfying a certain axiom. Examples include the set of degree sequences of a graph [9]. The lattice points contained in an integral bisubmodular polyhedron form a jump system, called a convex jump system, and conversely the convex hull of a jump system is an integral bisubmodular polyhedron. Recently, Lovász [20] investigated the membership problem in jump systems and proved a min-max theorem for a fairly wide class of jump systems. This result contains many interesting combinatorial theorems including Theorem 3.1. The present paper provides an algorithmic approach to this membership problem in convex jump systems.

4 Scaling Algorithm

This section presents a scaling algorithm for minimizaing an integer-valued bisubmodular function $f : 3^V \to \mathbf{Z}$, provided that an oracle for evaluating the function value is available.

The scaling algorithm works with a positive parameter δ. The algorithm keeps a vector $x \in \mathrm{P}(f)$ as a convex combination of extreme points of $\mathrm{P}(f)$. Namely, $x = \sum_{i \in I} \lambda_i y_i$ with $\lambda_i > 0$ for each $i \in I$ and $\sum_{i \in I} \lambda_i = 1$. Each extreme point y_i is generated by the greedy algorithm with respect to L_i and σ_i. It also keeps a pair of functions $\varphi : V \times V \to \mathbf{R}$ and $\psi : V \times V \to \mathbf{R}$. The function φ is skew-symmetric, i.e., $\varphi(u, v) + \varphi(v, u) = 0$ for any $u, v \in V$, while ψ is symmetric, i.e., $\psi(u, v) = \psi(v, u)$ for any $u, v \in V$. These functions are called δ-*feasible* if they satisfy $-\delta \leq \varphi(u, v) \leq \delta$ and $-\delta \leq \psi(u, v) \leq \delta$ for any $u, v \in V$. The boundaries $\partial\varphi$ and $\partial\psi$ are defined by $\partial\varphi(u) = \sum_{v \in V} \varphi(u, v)$ and $\partial\psi(u) = \sum_{v \in V} \psi(u, v)$.

The algorithm starts with an extreme point $x \in \mathrm{P}(f)$ generated by the greedy algorithm with respect to a linear ordering L and a sign function σ. The initial value of δ is given by $\delta := \|x\|/n^2$.

Each scaling phase starts by cutting the value of δ in half. Then it modifies φ and ψ to make them δ-feasible. This can be done by setting each $\varphi(u, v)$ and $\psi(u, v)$ to the closest values in the interval $[-\delta, \delta]$. The rest of the scaling phase aims at decreasing $\|z\|$ for $z = x + \partial\varphi + \partial\psi$.

Given δ-feasible φ and ψ, the algorithm constructs an auxiliary directed graph $G(\varphi, \psi)$ as follows. Let V^+ and V^- be the copies of V. For each $v \in V$, we denote its copies by $v^+ \in V^+$ and $v^- \in V^-$. The vertex set of $G(\varphi, \psi)$ is $V^+ \cup V^-$. The arc set $A(\varphi, \psi) = A(\varphi) \cup A(\psi)$ of $G(\varphi, \psi)$ is defined by

$$\begin{aligned} A(\varphi) &= \{(u^+, v^+) \mid u \neq v,\ \varphi(u, v) \leq 0\} \cup \{(u^-, v^-) \mid u \neq v,\ \varphi(u, v) \geq 0\}, \\ A(\psi) &= \{(u^+, v^-) \mid \psi(u, v) \leq 0\} \cup \{(u^-, v^+) \mid \psi(u, v) \geq 0\}. \end{aligned}$$

Let $S = \{v \mid v \in V, z(v) \leq -\delta\}$ and $T = \{v \mid v \in V, z(v) \geq \delta\}$. A simple directed path in $G(\varphi, \psi)$ from $S^+ \cup T^-$ to $S^- \cup T^+$ is called a δ-*augmenting path*. If there exists a δ-augmenting path P, the algorithm applies the following δ-augmentation to φ and ψ.

Augment$(\delta, P, \varphi, \psi)$:

- For each (u^+, v^+) in P, $\varphi(u, v) := \varphi(u, v) + \delta/2$ and $\varphi(v, u) := \varphi(v, u) - \delta/2$.
- For each (u^-, v^-) in P, $\varphi(u, v) := \varphi(u, v) - \delta/2$ and $\varphi(v, u) := \varphi(v, u) + \delta/2$.
- For each (u^+, v^-) in P, $\psi(u, v) := \psi(u, v) + \delta/2$ and $\psi(v, u) := \psi(v, u) + \delta/2$.
- For each (u^-, v^+) in P, $\psi(u, v) := \psi(u, v) - \delta/2$ and $\psi(v, u) := \psi(v, u) - \delta/2$.

As a result of a δ-augmentation, $\|z\|$ decreases by δ.

After each δ-augmentation, the algorithm computes an expression of x as a convex combination of affinely independent extreme points of $\mathrm{P}(f)$ chosen from among $\{y_i \mid i \in I\}$. This can be done by a standard linear programming technique using Gaussian elimination.

If there is no δ-augmenting path, let $X^+ \subseteq V^+$ and $Y^- \subseteq V^-$ be the sets of vertices reachable by directed paths from $S^+ \cup T^-$. Then we have $S \subseteq X$, $T \subseteq Y$, and $X \cap Y = \emptyset$. For each $i \in I$, consider a pair of disjoint subsets $W_i = \{u \mid u^{\sigma_i(u)} \in X^+ \cup Y^-\}$ and $R_i = \{u \mid u^{\sigma_i(u)} \in X^- \cup Y^+\}$. We now introduce two procedures Double-Exchange and Tail-Exchange.

Procedure Double-Exchange(i, u, v) is applicable if u immediately succeeds v in L_i and either $u \in W_i$ and $v \notin W_i$ or $u \notin R_i$ and $v \in R_i$ hold. Such a triple (i, u, v) is called *active*. The first step of the procedure is to compute

$$\beta := \sigma_i(u)\{f(L_i(u)\backslash\{v\}|\sigma_i) - f(L_i(u)|\sigma_i) + y_i(v)\}.$$

Then it interchanges u and v in L_i and updates y_i as $y_i := y_i + \beta(\sigma_i(u)\chi_u - \sigma_i(v)\chi_v)$. The resulting y_i is an extreme point generated by the new linear ordering L_i and sign function σ_i.

If $\lambda_i\beta \leq \delta$, Double-Exchange$(i, u, v)$ is called *saturating*. Otherwise, it is called *nonsaturating*. In the nonsaturating case, the procedure adds to I a new index k with y_k, σ_k and L_k being the previous y_i, σ_i and L_i, and assigns $\lambda_k := \lambda_i - \delta/\beta$ and $\lambda_i := \delta/\beta$. In both cases, x moves to $x := x + \alpha(\sigma_i(u)\chi_u - \sigma_i(v)\chi_v)$ with $\alpha = \min\{\delta, \lambda_i\beta\}$. In order to keep z invariant, the procedure finally modifies φ or ψ appropriately. If $\sigma_i(u) = \sigma_i(v)$, it updates $\varphi(u, v) := \varphi(u, v) - \sigma_i(u)\alpha$ and $\varphi(v, u) := \varphi(v, u) + \sigma_i(u)\alpha$. On the other hand, if $\sigma_i(u) \neq \sigma_i(v)$, then $\psi(u, v) := \psi(u, v) - \sigma_i(u)\alpha$ and $\psi(v, u) := \psi(v, u) - \sigma_i(u)\alpha$. A formal description of Double-Exchange is given in Figure 1.

Procedure Tail-Exchange(i, v) is applicable if v is the last element in L_i and $v \in R_i$. Such a pair (i, v) is also called *active*. The first step of the procedure is to reverse the sign $\sigma_i(v)$. It then computes

$$\beta := f(V|\sigma_i) - f(V\backslash\{v\}|\sigma_i) - \sigma_i(v)y_i(v)$$

and updates $y_i := y_i + \sigma_i(v)\beta\chi_v$. The resulting y_i is an extreme point generated by L_i and the new σ_i.

If $\lambda_i\beta \leq \delta$, Tail-Exchange$(i, v)$ is called *saturating*. Otherwise, it is called *nonsaturating*. In the nonsaturating case, the procedure adds to I a new index k with y_k, σ_k and L_k being the previous y_i, σ_i and L_i, and assigns $\lambda_k := \lambda_i - \delta/\beta$ and $\lambda_i := \delta/\beta$. In both cases, x moves to $x := x + \sigma_i(v)\alpha\chi_v$ with $\alpha = \min\{\delta, \lambda_i\beta\}$. In order to keep z invariant, the procedure finally modifies ψ as $\psi(v, v) := \psi(v, v) - \sigma_i(v)\alpha$. A formal description of Tail-Exchange is given in Figure 2.

Double-Exchange(i, u, v);
$\beta := \sigma_i(u)\{f(L_i(u)\backslash\{v\}|\sigma_i) - f(L_i(u)|\sigma_i) + y_i(v)\}$;
$\alpha := \min\{\delta,\ \lambda_i\beta\}$;
If $\alpha < \lambda_i\beta$ **then**
    $k \leftarrow$ a new index;
    $I := I \cup \{k\}$;
    $\lambda_k := \lambda_i - \alpha/\beta$;
    $\lambda_i := \alpha/\beta$;
    $y_k := y_i$;
    $L_k := L_i$;
Update L_i by interchanging u and v;
$y_i := y_i + \beta(\sigma_i(u)\chi_u - \sigma_i(v)\chi_v)$;
$x := x + \alpha(\sigma_i(u)\chi_u - \sigma_i(v)\chi_v)$;
If $\sigma_i(u) = \sigma_i(v)$ **then**
    $\varphi(u, v) := \varphi(u, v) - \sigma_i(u)\alpha$;
    $\varphi(v, u) := \varphi(v, u) + \sigma_i(u)\alpha$;
Else
    $\psi(u, v) := \psi(u, v) - \sigma_i(u)\alpha$;
    $\psi(v, u) := \psi(v, u) - \sigma_i(u)\alpha$.

Figure 1: Algorithmic description of Procedure Double-Exchange(i, u, v).

If there is no δ-augmenting path and neither Double-Exchange nor Tail-Exchange is applicable, the algorithm terminates the δ-scaling phase by cutting the value of δ in half.

A formal description of our scaling algorithm BFM is now given in Figure 3.

5 Validity and Complexity

This section is devoted to the analysis of our scaling algorithm. We first discuss the validity.

Lemma 5.1 *At the end of the δ-scaling phase, the current $(X, Y) \in 3^V$ and $z = x + \partial\varphi + \partial\psi$ satisfy $\|z\| \leq 2n\delta - f(X, Y)$.*

Proof. At the end of the δ-scaling phase, we have $y_i(X) - y_i(Y) = f(X, Y)$ for each $i \in I$. Hence, x satisfies $x(X) - x(Y) = f(X, Y)$. By the definition of (X, Y), we immediately have $\partial\varphi(X) > 0$, $\partial\varphi(Y) < 0$, $\partial\psi(X) > 0$, and $\partial\psi(Y) < 0$, where

Tail-Exchange(i, v);
$\sigma_i(v) := -\sigma_i(v)$;
$\beta := f(V|\sigma_i) - f(V\backslash\{v\}|\sigma_i) - \sigma_i(v)y_i(v)$;
$\alpha := \min\{\delta,\ \lambda_i\beta\}$;
If $\alpha < \lambda_i\beta$ **then**
    $k \leftarrow$ a new index;
    $I := I \cup \{k\}$;
    $\lambda_k := \lambda_i - \alpha/\beta$;
    $\lambda_i := \alpha/\beta$;
    $y_k := y_i$;
    $L_k := L_i$;
$y_i := y_i + \sigma_i(v)\beta\chi_v$;
$x := x + \sigma_i(v)\alpha\chi_v$;
$\psi(v, v) := \psi(v, v) - \sigma_i(v)\alpha$.

Figure 2: Algorithmic description of Procedure Tail-Exchange(i, v).

note that $\partial\varphi(X) = \sum\{\varphi(u, v) \mid u \in X, v \in V\backslash X\}$ with $\varphi(u, v) > 0$ $(u \in X, v \in V \setminus X)$ and similarly the other inequalities. Since $S \subseteq X$ and $T \subseteq Y$, we have $z(v) \geq -\delta$ for $v \in V\backslash X$ and $z(v) \leq \delta$ for $v \in V\backslash Y$ and . Therefore, we have $\|z\| \leq -z(X) + z(Y) + 2n\delta \leq -x(X) + x(Y) + 2n\delta = -f(X, Y) + 2n\delta$. ■

Theorem 5.2 *The algorithm obtains a minimizer of f at the end of the δ-scaling phase with $\delta < 1/3n^2$.*

Proof. Since $|\partial\varphi(v)| \leq (n-1)\delta$ and $|\partial\psi(v)| \leq n\delta$ for each $v \in V$, it follows from Lemma 5.1 that $\|x\| \leq (2n^2+n)\delta - f(X, Y) < 1 - f(X, Y)$. For any $(X', Y') \in 3^V$, we have $f(X', Y') \geq -\|x\| > f(X, Y) - 1$. Hence (X, Y) is a minimizer of the integer-valued function f. ■

We now give a running time bound of our algorithm.

Lemma 5.3 *Each scaling phase performs $O(n^2)$ augmentations.*

Proof. At the beginning of the δ-scaling phase, the algorithm modifies φ and ψ to make them δ-feasible. This changes $\|z\|$ by at most $2n^2\delta$. Therefore, by Lemma 5.1, the pair (X, Y) must satisfy $\|z\| \leq 2n^2\delta + 4n\delta - f(X, Y)$ after updating φ and ψ at the beginning of the δ-scaling phase. On the other hand, we have $\|z\| \geq -z(X) + z(Y) \geq -x(X) + x(Y) - 2n^2\delta = -f(X, Y) - 2n^2\delta$. Thus $\|z\|$

decreases by at most $4n\delta + 4n^2\delta$ until the end of the δ-scaling phase. Since each δ-augmentation decreases $\|z\|$ by δ, the number of δ-augmentations in the δ-scaling phase is at most $4n^2 + 4n$, which is $O(n^2)$. ■

Lemma 5.4 *The algorithm performs Procedure* Double-Exchange $O(n^3)$ *times and* Tail-Exchange $O(n^2)$ *times between δ-augmentations.*

Proof. Procedure Double-Exchange moves a vertex of W_i towards the head of L_i and/or a vertex in R_i towards the tail of L_i. Procedure Tail-Exchange changes a vertex of R_i to W_i. No vertex goes out of W_i. A vertex of R_i can be switched to W_i by Tail-Exchange. However, it does not go out of $R_i \cup W_i$. Thus, for each $i \in I$, after at most $O(n^2)$ applications of Double-Exchange and $O(n)$ applications of Tail-Exchange to $i \in I$, the subset R_i is empty and $W = L(w)$ holds for some $w \in V$. At this point, neither Double-Exchange nor Tail-Exchange is applicable to $i \in I$.

After each δ-augmentation, the algorithm updates the convex combination $x = \sum_{i \in I} \lambda_i y_i$ so that $|I| \leq n + 1$. A new index is added to I as a result of nonsaturating Double-Exchange(i, u, v) and Tail-Exchange(i, v). In both cases, v joins W_i. This can happen at most $n - 1$ times before the algorithm finds a δ-augmenting path or finishes the δ-scaling phase. Hence, $|I|$ is always $O(n)$, and the algorithm performs Double-Exchange $O(n^3)$ times and Tail-Exchange $O(n^2)$ times between δ-augmentations. ■

Let M be the maximum value of f. Since $f(\emptyset, \emptyset) = 0$, the maximum value M is nonnegative.

Theorem 5.5 *The scaling algorithm finds a minimizer of f in $O(n^5 \log M)$ time.*

Proof. For the initial $x \in P(f)$, let $B = \{v \mid x(v) > 0\}$ and $C = \{v \mid x(v) < 0\}$. Then we have $\|x\| = x(B) - x(C) \leq f(B, C) \leq M$. Hence the algorithm performs $O(\log M)$ scaling phases. It follows from Lemmas 5.3 and 5.4 that each scaling phase performs $O(n^5)$ function evaluations and arithmetic operations. Therefore the total running time is $O(n^5 \log M)$. ■

6 Conclusion

We have described a polynomial-time algorithm for minimizing integer-valued bisubmodular functions. If we are given a positive lower bound ϵ for the difference between the minimum and the second minimum value of f, a variant of the present algorithm works for any real-valued bisubmodular function f. The only required modification is to change the stopping rule $\delta < 1/3n^2$ to $\delta < \epsilon/3n^2$. The running time is $O(n^5 \log(M/\epsilon))$. Thus we obtain a polynomial-time algorithm for testing membership in delta-matroid polyhedra. One can make this algorithm strongly

polynomial with the aid of a generic preprocessing technique of Frank–Tardos [13] using simultaneous Diophantine approximation. However, a more natural strongly polynomial algorithm is desirable.

References

[1] K. Ando and S. Fujishige: On structures of bisubmodular polyhedra, *Math. Programming*, 74 (1996), 293–317.

[2] E. Balas and W. R. Pulleyblank: The perfectly matchable subgraph polytope of an arbitrary graph, *Combinatorica*, 9 (1989), 321–337.

[3] A. Bouchet: Greedy algorithm and symmetric matroids, *Math. Programming*, 38 (1987), 147–159.

[4] A. Bouchet: Matchings and $\triangle$-matroids, *Discrete Appl. Math.*, 24 (1989), 55–62.

[5] A. Bouchet and W. H. Cunningham: Delta-matroids, jump systems and bisubmodular polyhedra, *SIAM J. Discrete Math.*, 8 (1995), 17–32.

[6] R. Chandrasekaran and S. N. Kabadi: Pseudomatroids, *Discrete Math.*, 71 (1988), 205–217.

[7] W. H. Cunningham: Testing membership in matroid polyhedra, *J. Combin. Theory*, B36 (1984), 161–188.

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

[9] W. H. Cunningham and J. Green-Krótki: b-matching degree sequence polyhedra, *Combinatorica*, 11 (1991), 219–230.

[10] W. H. Cunningham and J. Green-Krótki: A separation algorithm for the matchable set polytope, *Math. Programming*, 65 (1994), 139–150.

[11] A. Dress and T. F. Havel: Some combinatorial properties of discriminants in metric vector spaces, *Adv. Math.*, 62 (1986), 285–312.

[12] F. D. J. Dunstan and D. J. A. Welsh: A greedy algorithm solving a certain class of linear programmes, *Math. Programming*, 5 (1973), 338–353.

[13] A. Frank and É. Tardos: An application of simultaneous Diophantine approximation in combinatorial optimization, *Combinatorica*, 7 (1987), 49–65.

[14] S. Fujishige: *Submodular Functions and Optimization*, North-Holland, 1991.

[15] S. Fujishige: A min-max theorem for bisubmodular polyhedra, *SIAM J. Discrete Math.*, 10 (1997), 294–308.

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

[17] M. Grötschel, L. Lovász, and A. Schrijver: *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, 1988.

[18] S. Iwata, L. Fleischer, and S. Fujishige: A combinatorial strongly polynomial algorithm for minimizing submodular functions, *J. ACM*, submitted.

[19] L. Lovász: Submodular functions and convexity. *Mathematical Programming — The State of the Art*, A. Bachem, M. Grötschel and B. Korte, eds., Springer-Verlag, 1983, 235–257.

[20] L. Lovász: The membership problem in jump systems, *J. Combin. Theory*, Ser. B, 70 (1997), 45–66.

[21] L. Qi: Directed submodularity, ditroids and directed submodular flows, *Math. Programming*, 42 (1988), 579–599.

[22] A. Schrijver: A combinatorial algorithm minimizing submodular functions in strongly polynomial time, *J. Combin. Theory*, Ser. B, to appear.

BFM(f):
Initialization:
$L \leftarrow$ a linear ordering on V;
$\sigma \leftarrow$ a sign function on V;
$x \leftarrow$ an extreme vector in $\mathrm{P}(f)$ generated by L and σ;
$I := \{\ell\}$, $y_\ell := x$, $\lambda_\ell := 1$, $L_\ell := L$;
$\varphi := \mathbf{0}$, $\psi := \mathbf{0}$;
$\delta \leftarrow \|x\|/n^2$;
While $\delta \geq 1/3n^2$ **do**
$\delta := \delta/2$;
For $(u, v) \in V \times V$ **do**
Change $\varphi(u, v)$ and $\psi(u, v)$ to the closest values in the interval $[-\delta, \delta]$;
$S := \{v \mid x(v) + \partial\varphi(v) + \partial\psi(v) \leq -\delta\}$;
$T := \{v \mid x(v) + \partial\varphi(v) + \partial\psi(v) \geq \delta\}$;
$X^+ \leftarrow$ the set of vertices in V^+ reachable from $S^+ \cup T^-$ in $G(\varphi, \psi)$;
$Y^- \leftarrow$ the set of vertices in V^- reachable from $S^+ \cup T^-$ in $G(\varphi, \psi)$;
$Q \leftarrow$ the set of active triples and active pairs;
While $\exists\delta$-augmenting path or $Q \neq \emptyset$ **do**
If $\exists P$: δ-augmenting path **then**
Augment$(\delta, P, \varphi, \psi)$;
Update S, T, X^+, Y^-, Q;
Express x as $x = \sum_{i \in I} \lambda_i y_i$ by possibly smaller affinely independent subset I and positive coefficients $\lambda_i > 0$ for $i \in I$;
Else
While $\nexists\delta$-augmenting path and $Q \neq \emptyset$ **do**
Find an active $(i, u, v) \in Q$ or active $(i, v) \in Q$;
Apply Double-Exchange(i, u, v) or Tail-Exchange(i, v);
Update X^+, Y^-, Q;
Return (X, Y);
End.

Figure 3: A scaling algorithm for bisubmodular function minimization.