A Submodular Function Minimization Algorithm Based on the Minimum-Norm Base*

Satoru FUJISHIGE† and Shigueo ISOTANI‡

March 2009; revised June 2009

Abstract

We consider an application of the minimum-norm-point algorithm to submodular function minimization. Although combinatorial polynomial algorithms for submodular function minimization (SFM) have recently been obtained, there still remain (open) problems of reducing the complexity of the SFM algorithms and of constructing a practically fast SFM algorithms. We show some possible approach to the problems by means of the minimum-norm-point algorithm. Computational results on submodular function minimization reveal that our algorithm outperforms existing polynomial algorithms for SFM.

Keywords: Submodular function, Minimum norm point, Algorithms, Base polyhedron

AMS Subject Classifications: 65K05, 90C27, 52B40, 68Q25

Abbreviated title: A Submodular Function Minimization Algorithm

1. Introduction

Philip Wolfe [18] presented an algorithm for finding the minimum-norm point in the convex hull of a given finite set of points in the n-dimensional Euclidean space $\mathbf{R}^n$ (von Hohenbalken [7] also gave essentially the same algorithm for more general objective functions). In the present paper we consider an application of the minimum-norm-point

*Presented by the first author as a plenary talk at the fourth Sino-Japanese Optimization Meeting held on August 27–31, 2008, in Tainan, Taiwan.

†Research Institute for Mathematical Sciences, Kyoto University, Kyoto 606-8502, Japan. E-mail: fujishig@kurims.kyoto-u.ac.jp

‡Faculdade de Ciências Econômicas e Administrativas de Osasco – FAC-FITO, Osasco, SP, Brazil. E-mail: shigueo.isotani@gmail.com

algorithm to submodular function minimization. Combinatorial polynomial algorithms for submodular function minimization (SFM) have been obtained by [11, 9, 16, 15, 12]. However, there still remain (open) problems of reducing the complexity of the SFM algorithms and of constructing practically fast SFM algorithms. The present paper has been motivated by an attempt to solve these problems. We will show a possible approach to them by means of the minimum-norm-point algorithm to construct a practically fast algorithm for submodular function minimization. (Such an approach was first suggested by the first author in [4] and some preliminary computational experiments were made by the second author in [8].)

The minimum-norm-point algorithm keeps a simplex (a set of affinely independent set of points) chosen from the given point set P. Updating such a simplex requires a solution of a linear optimization problem over the convex hull $\hat{P}$ of P, and the algorithm works if the linear optimization can (efficiently) be done over the polytope $\hat{P}$. In the original problem setting by Wolfe [18] the polytope $\hat{P}$ is expressed as the convex hull of the set P of given points. Hence, the linear optimization over $\hat{P}$ can be done trivially by the evaluation of a linear function on given points in P. For a general polytope Q, however, the number of extreme points of Q can be exponentially large with respect to dimension n, so that the minimum-norm-point algorithm cannot be used in the original, trivial way. However, there are interesting classes of polytopes on which linear optimization can efficiently be done, even if the number of extreme points of Q is exponentially large with respect to dimension n.

The following is one of such classes of polytopes. It is well known that the greedy algorithm [1] works for base polyhedra associated with submodular functions (see [6] for details about submodular functions and related polyhedra). Hence we can easily make linear optimization over base polyhedra (although the number of extreme points can be equal to $n!$ with n being the dimension of the space that contains the base polyhedra). This fact leads us to an algorithm for submodular function minimization by means of the minimum-norm-point algorithm, which will be discussed in Section 3.

We examine the proposed algorithm for submodular function minimization in Section 4. Computational results for submodular function minimization will show that the minimum-norm-point algorithm outperforms the existing polynomial-time algorithms given in [9, 11, 16].

2. The Minimum-Norm-Point Algorithm

In this section we describe Wolfe's algorithm [18] for finding the minimum-norm point in a polytope for completeness.

2.1. Description of the minimum-norm-point algorithm

Consider the n-dimensional Euclidean space $\mathbf{R}^n$. Suppose that we are given a finite set P of points p_i $(i \in I)$ in $\mathbf{R}^n$. The problem is to find the minimum-norm point x^* in the convex hull $\hat{P}$ of points p_i $(i \in I)$.

Wolfe's algorithm [18] is given as follows.

The Minimum-Norm-Point Algorithm

Input: A finite set P of points p_i $(i \in I_0)$ in $\mathbf{R}^n$.
Output: The minimum-norm point x^* in the convex hull $\hat{P}$ of the points p_i $(i \in I_0)$.

Step 1: Choose any point p in P and put $S := \{p\}$ and $\hat{x} := p$.

Step 2: Find a point $\hat{p}$ in P that minimizes the linear function $\langle \hat{x}, p \rangle = \sum_{k=1}^{n} \hat{x}(k)p(k)$ in $p \in P$. Put $S := S \cup \{\hat{p}\}$.
If $\langle \hat{x}, \hat{p} \rangle = \langle \hat{x}, \hat{x} \rangle$, then return $x^* = \hat{x}$;
else go to Step 3.

Step 3: Find the minimum-norm point y in the affine hull of points in S.
If y lies in the relative interior of the convex hull of S, then put $\hat{x} := y$ and go to Step 2.

Step 4: Let z be the point that is the nearest to y among the intersection of the convex hull of S and the line segment $[y, \hat{x}]$ between y and $\hat{x}$. Also let $S' \subset S$ be the unique proper subset of S such that z lies in the relative interior of the convex hull of S'. Put $S := S'$ and $\hat{x} := z$. Go to Step 3.
(End)

The cycle formed by Step 2 and Step 3 is called a *major cycle*, and the one by Step 3 and Step 4 a *minor cycle*. Every major cycle increases the size of the simplex S by one, while every minor cycle decreases the size of the simplex S by at least one. A simplex S is called a *corral* if the minimum-norm point in the affine hull of S lies in the relative interior of the convex hull of S. When we go from Step 3 to Step 2 in a major cycle, the current simplex S is a corral. Note that every corral S uniquely determines the minimum-norm point $\hat{x}$ and that every time we get a new corral, the norm of the new $\hat{x}$ strictly decreases. Also note that at most $n - 1$ repetitions of Step 3 and Step 4 in a minor cycle give a corral, so that the Wolfe algorithm described above terminates in a finite number of steps. (It is open to determine whether the Wolfe algorithm runs in polynomial time.)

In Step 3, for $S = \{p_i \mid i \in I\}$ with $I \subseteq I_0$ we have $y = \sum_{i \in I} \mu_i p_i$ with $\sum_{i \in I} \mu_i = 1$. Note that y lies in the relative interior of the convex hull of S if and only if $\mu_i > 0$ for all $i \in I$, where recall that S is affinely independent. In Step 4, both $\hat{x}$ and y are expressed as $\hat{x} = \sum_{i \in I} \lambda_i p_i$ and $y = \sum_{i \in I} \mu_i p_i$. Then, the point z is determined in such a way that $z = (1 - \beta)\hat{x} + \beta y$, $(1 - \beta)\lambda_i + \beta\mu_i \geq 0$ for all $i \in I$, and β is as large as possible.

Remark: When implementing the Wolfe algorithm, we should take care of numerical errors by introducing small tolerance intervals for decisions such as '$\alpha = \beta$?'. Besides these, the algorithm is self-correcting, so that it is stable against numerical errors. For the minimum norm base we can utilize further information about the bounded fractionality of the solution to make our algorithm stable against numerical errors, which will be discussed later (see Theorem 3.3). □

2.2. Applicability of the algorithm

The Wolfe algorithm requires linear optimization in Step 2, which can be done by computing $\langle \hat{x}, p \rangle$ for all points p in P. If the number of points in P is exponential in the dimension of the space $\mathbf{R}^n$, then it becomes hard to perform the linear optimization in Step 2 in such a primitive way.

Now, suppose that the set P is implicitly given as the set of extreme points of a polytope Q in $\mathbf{R}^n$. Then the Wolfe algorithm works if linear optimization over Q can efficiently be made. There are classes of polytopes on which linear optimization can efficiently be done while the number of the extreme points of such a polytope is exponentially large. For example, we have

(1) base polyhedra, associated with submodular functions, on which the so-called greedy algorithm of Edmonds [1] finds optimal (extreme) points, and

(2) zonotopes, on which every linear optimization can be done in a greedy way,

where a zonotope is the Minkowski sum of line segments (or an affine transformation of a unit hypercube).

Remark: A pointed polyhedron is called *edge-polynomial* [5] if the number of edge vectors of the polyhedron is polynomial in the dimension of the input data space, where edge vectors are identified up to nonzero multiples. Base polyhedra and zonotopes are typical edge-polynomial polyhedra. The number of edge vectors of base polyhedra is $\mathrm{O}(n^2)$ with n being the dimension of the space, and that of zonotopes is at most the number of the generators. It should be noted that linear optimization over any edge-polynomial polyhedron is easy (solvable in strongly polynomial time) under certain conditions, so that the minimum-norm-point algorithm works for edge-polynomial polyhedra. □

We shall show how the Wolfe algorithm works for base polyhedra in Section 3.

3. Base Polyhedra and Submodular Function Minimization

In this section we show how the Wolfe algorithm can be used to minimize submodular functions.

3.1. Submodular functions and base polyhedra

Let E be a finite nonempty set and f be a submodular function on 2^E, i.e., $f : 2^E \to \mathbf{R}$ satisfies

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y) \tag{3.1}$$

for any $X, Y \subseteq E$. We suppose that $f(\emptyset) = 0$ without loss of generality. We then define polyhedra

$$\mathrm{P}(f) = \{x \mid x \in \mathbf{R}^E,\ \forall X \in 2^E : x(X) \leq f(X)\}, \tag{3.2}$$

$$\mathrm{B}(f) = \{x \mid x \in \mathrm{P}(f),\ x(E) = f(E)\}. \tag{3.3}$$

Here, $\mathrm{P}(f)$ is called the *submodular polyhedron* and $\mathrm{B}(f)$ the *base polyhedron*, associated with submodular function f on 2^E.

Remark: Since $\mathrm{B}(f)$ defined as above is bounded, it is also called a base polytope. Note that $\mathrm{P}(f)$ is always unbounded. In the general theory of submodular functions (see [6]) we consider a distributive lattice $\mathcal{D} \subseteq 2^E$ (a set of subsets of E that is closed with respect to set union $\cup$ and intersection $\cap$) and a submodular function f on $\mathcal{D}$. We assume that $\emptyset, E \in \mathcal{D}$ and $f(\emptyset) = 0$. Then $\mathrm{B}(f)$ is defined similarly as in (3.2), and is bounded only if $\mathcal{D} = 2^V$. □

The linear optimization over base polyhedron $\mathrm{B}(f)$ can easily be made by the greedy algorithm of Edmonds [1]. Here we assume that we are given an oracle for evaluation of the function value $f(X)$ for any $X \subseteq E$.

The Greedy Algorithm

Input A weight vector $w \in \mathbf{R}^E$.
Output: An optimal $x^* \in \mathrm{B}(f)$ that minimizes the linear objective function $\sum_{e \in E} w(e)x(e)$ in $x \in \mathrm{B}(f)$.

Step 1: Find a linear ordering $e_1, e_2, \cdots, e_n$ of elements of E such that

$$w(e_1) \leq w(e_2) \leq \cdots \leq w(e_n). \tag{3.4}$$

Step 2: Compute

$$x^*(e_i) = f(\{e_1, e_2, \cdots, e_i\}) - f(\{e_1, e_2, \cdots, e_{i-1}\}) \quad (i = 1, 2, \cdots, n). \tag{3.5}$$

Return x^*.
(End)

We also have the following theorem that characterizes the minimizers of a submodular function $f : 2^E \to \mathbf{R}$ with $f(\emptyset) = 0$.

Theorem 3.1 ([1]): *We have*

$$\min\{f(X) \mid X \subseteq E\} = \max\{x^-(E) \mid x \in \mathrm{B}(f)\}, \tag{3.6}$$

where $x^-(e) = \min\{x(e), 0\}$ *for* $e \in E$.

Moreover, if f *is integer-valued, then the maximum in the right-hand side is attained by an integral base* $x \in \mathrm{B}(f)$. □

Note that for any $X \subseteq E$ and $x \in \mathrm{B}(f)$ we have $f(X) \geq x^-(E)$. The gap $f(X) - x^-(E)$ evaluates an upper bound for to what extent $f(X)$ is close to the minimum of f. In particular, if f is integer-valued, the gap $f(X) - x^-(E)$ being less than one implies that X is a minimizer of f.

3.2. The minimum-norm point in a base polyhedron and submodular function minimization

Concerning a relationship between the minimum-norm base and the submodular function minimization, we have the following theorem.

Theorem 3.2 ([4], [6, Sec. 7.1.(a)]): *Let* x^* *be the minimum-norm point in the base polyhedron* $\mathrm{B}(f)$ *given by* (3.3)*. Define*

$$A_+ = \{e \mid e \in E,\ x^*(e) \leq 0\}, \tag{3.7}$$
$$A_- = \{e \mid e \in E,\ x^*(e) < 0\}. \tag{3.8}$$

Then, A_+ *is the unique maximal minimizer of* f*, and* A_- *the unique minimal minimizer of* f. □

Because of this theorem we can solve the submodular function minimization problem by finding the minimum-norm point in the base polyhedron $\mathrm{B}(f)$. The minimum-norm-point algorithm described in Section 2 can directly be employed to solve the submodular function minimization problem by means of the greedy algorithm of Edmonds. Computational results will be given in Section 4.

The following characterization of the minimum-norm base is known, which is useful for avoiding numerical errors to perform submodular function minimization.

Theorem 3.3 ([3], [6, Sec. 9.2]): *A base $b \in \mathrm{B}(f)$ is the minimum-norm base if and only if for distinct values $\alpha_1 < \cdots < \alpha_k$ of $b(e)$ $(e \in E)$ and for $F_i = \{e \in E \mid b(e) \leq \alpha_i\}$ $(i = 1, \cdots, k)$ we have $b(F_i) = f(F_i)$ $(i = 1, \cdots, k)$.*

Hence we have

$$b(e) = \frac{f(F_i) - f(F_{i-1})}{|F_i \setminus F_{i-1}|} \qquad (e \in F_i \setminus F_{i-1}) \tag{3.9}$$

for $i = 1, \cdots, k$, where $F_0 \equiv \emptyset$. □

Remark: Because of this theorem, if we know the value

$$\min\{|f(Y) - f(X)| \mid f(X) \neq f(Y), X \subset Y \subseteq E\} \tag{3.10}$$

or its positive lower bound (say, ϵ), then we have

$$A_+ = \{e \mid e \in E,\ x^*(e) < \epsilon/2|E|\}, \tag{3.11}$$

$$A_- = \{e \mid e \in E,\ x^*(e) < -\epsilon/2|E|\}. \tag{3.12}$$

instead of (3.7) and (3.8). Note that when f is integer-valued, we can take $\epsilon = 1.0$ and that we can avoid the possible numerical errors up to $1/2|E|$ for computing A_+ and A_- through (3.11) and (3.12). The present idea will be incorporated into the code of our proposed algorithm. □

3.3. Possible problem reduction

In this subsection we consider a possible way of accelerating our algorithm FW by extracting information about minimizers of a given submodular function f.

Suppose that a current base x is given as a convex combination

$$x = \sum_{i \in I} \lambda_i b_i \tag{3.13}$$

of extreme bases b_i $(i \in I)$, where $\lambda_i > 0$ for all $i \in I$. Each $i \in I$ is associated with a linear ordering L_i. A set $A \subseteq E$ is called *x-tight* if $x(A) = f(A)$ and that $B \subseteq E$ is called *x-cotight* if $x(B) = f^{\#}(B) (\equiv f(E) - f(E \setminus B))$. It can easily be seen that the following three are equivalent:

(a) A is x-tight.
(b) A is b_i-tight for all $i \in I$.
(c) A is an initial segment of L_i for all $i \in I$.

Similarly in a dual form, the following three are equivalent:

(a′) B is x-cotight.
(b′) B is b_i-cotight for all $i \in I$.
(c′) B is a terminal segment of L_i for all $i \in I$.

Moreover, we can show the following.

Theorem 3.4: *For any set $A \subseteq E$ such that*
(1) A is x-tight, i.e., $x(A) = f(A)$ and
(2) $x(e) < 0$ for all $e \in A$,
then A is contained in every minimizer of f.

Dually, for any set $B \subseteq E$ such that
(1′) B is x-cotight, i.e., $x(B) = f(E) - f(E \setminus B)$ and
(2′) $x(e) > 0$ for all $e \in B$,
then every minimizer of f is contained in $E \setminus B$. □

Because of Theorems 3.3 and 3.4 we can consider the following procedure of the problem reduction.

During the execution of our FW algorithm, when we get a corral S and a point x (the minimum-norm point within the relative interior of $\mathrm{Conv}(S)$) and try to augment it by finding a new extreme base y, we may carry out the following.

Problem Reduction

1. Sort $x(e)$ $(e \in E)$ so as to get $x(e_1) \leq \cdots \leq x(e_n)$.
2. Find the maximum $i \in \{1, \cdots, n\}$ such that $x(\{e_1, \cdots, e_i\}) = f(\{e_1, \cdots, e_i\})$ and for all $k = 1, \cdots, i$ we have $x(e_k) < 0$. If such an i exists, put $A \leftarrow \{e_1, \cdots, e_i\}$; otherwise put $A \leftarrow \emptyset$.
3. Find the minimum $j \in \{1, \cdots, n\}$ such that $x(\{e_j, \cdots, e_n\}) = f^{\#}(\{e_j, \cdots, e_n\})$ and for all $k = j, \cdots, n$ we have $x(e_k) > 0$. If such a j exists, put $B \leftarrow \{e_j, \cdots, e_n\}$; otherwise put $B \leftarrow \emptyset$. □

If we find nonempty A or B by the above procedure we can consider a new function $f_A^{E \setminus B} : 2^{E \setminus (A \cup B)} \rightarrow \mathbf{R}$ defined by

$$f_A^{E \setminus B}(X) = f(A \cup X) - f(A), \qquad X \subseteq E \setminus (A \cup B). \tag{3.14}$$

Every minimizer X^* of f is given by a minimizer Y^* of $f_A^{E \setminus B}$ as

$$X^* = Y^* \cup A \tag{3.15}$$

and *vice versa.*

In particular, we have a duality gap $f(A) - x^*(A)$, so that if f is integer-valued and $f(A) - x^*(A) < 1$, then we can terminate our FW algorithm with a minimizer A of f. This stopping rule accelerates FW, which we incorporate into FW in our computational experiments given in the next section.

4. Computational results

First combinatorial polynomial algorithms for submodular function minimization (SFM) were devised independently by Iwata, Fleischer, and Fujishige [11], and Schrijver [16]. Also Fleischer and Iwata [2] proposed a polynomial preflow-push algorithm, which has the same complexity as Schrijver's ([17]). See nice surveys [13] and [10] for more details about recent developments in SFM algorithms (also see [6, Chapter VI]). Currently the theoretically fastest SFM algorithm has been obtained by Orlin [15](also see [12]).

Part of the following computational results on SFM algorithms are based on a report of [8].

4.1. Computational Setup

We used a Dynabook G6/X18PDE with an Intel Pentium 4, CPU 1.80GHz, 768MB of memory and running Linux RedHat version 2.4.18. All programs are written in C language and compiled with gcc using the -O4 optimization option.

We denote by FW the proposed SFM algorithm by means of the minimum-norm-point algorithm [4]. The Iwata-Fleischer-Fujishige algorithm [11] is denoted by SFM3 and SFM8 (an updated version of SFM3) and Schrijver's algorithm [16] by LEX2. We also have Fleischer and Iwata's algorithm [2], denoted by PR. Moreover, HYBRID is an algorithm, proposed by Iwata [9], that combines techniques involved in SFM3, SFM8, and PR (also see [10]). We have employed the codes of SFM3, SFM8, PR, and HYBRID offered by Satoru Iwata.

The original version of FW program was first written in FORTRAN language by Masahiro Nakayama and later in PASCAL by Shingo Shikita in their graduation theses at the University of Tsukuba in February, 1985 and in February, 1987, respectively, under the guidance by the first author. We employed Quick Sort for the sorting algorithm required in the greedy algorithm. We rewrote the program in C language and improved some part of it, incorporating into it the idea shown in the previous section.

We tested the algorithms using two kinds of submodular functions. One is proposed by Satoru Iwata and the other is a class of cut functions.

4.2. Iwata's Test Function

The submodular function suggested by Satoru Iwata is

$$f(X) = |X||V \setminus X| - \sum_{j \in X}(5j - 2n) \qquad (X \subseteq V)$$

where $V = \{1, 2, \cdots, n\}$.

The results on this function are shown in Table 1 and Table 2.

Table 1: Results on Iwata's function

Running time (sec)						
n	FW	HYBRID	SFM3	SFM8	LEX2	PR
100	0.00	0.41	1.00	0.04	2644.52	277.36
200	0.00	4.92	18.69	0.44		
300	0.00	21.77	115.44	2.04		
400	0.00	67.12	369.13	6.32		
500	0.00	166.73	894.33	15.35		
600	0.01	325.26	2820.83	34.88		
700	0.01	568.54		62.38		

Table 2: Numbers of generated bases on Iwata's function

Number of generated bases						
n	FW	HYBRID	SFM3	SFM8	LEX2	PR
100	2	1163	766	1	337348	373324
200	2	3732	4618	1		
300	2	6710	7309	1		
400	2	10803	9914	1		
500	2	16701	18835	1		
600	2	22011	33849	1		
700	2	28699		1		

This class of test problems is very special for FW and SFM8, where SFM8 is a new version of SFM3 adapted to the test problems by a preprocessing to choose an appropriate initial extreme base. Except for FW and SFM8, HYBRID outperformed the others, LEX2 and PR.

It should be noted that FW starts with an initial extreme base b_0 corresponding to linear ordering $L_0 = (1, 2, \cdots, n)$ and we have $b_0(i) = -7i + 3n + 1$ $(i = 1, \cdots, n)$, so that $b_0(n) < b_0(n-1) < \cdots < b_0(1)$. Hence FW generates the next extreme base b_1 corresponding to linear ordering $L_1 = (n, n-1, \cdots, 1)$, where b_1 is given by $b_1(i) = -3i+n-1$ $(i = 1, \cdots, n)$. We then update the current simplex $S = \{b_0, b_1\}$ to $\{b_1\}$. Here we have $b_1(n) < b_1(n-1) < \cdots < b_1(1)$ again and each initial segment $\{n, n-1, \cdots, i\}$ of L_1 is a b_1-tight set for $i = 1, \cdots, n$. Hence b_1 is the minimum-norm base and FW terminates by generating two extreme bases.

4.3. Cut Functions

In the case of cut functions, we need to generate networks. We used the generator GENRMF available from DIMACS Challenge [19]. Each generated network has b grid-like frames of size $(a \times a)$. The number of vertices is a^2b and that of arcs $5a^2b - 4ab - a^2$. All vertices in each frame are connected to its grid neighbors and each vertex is connected by an arc to a vertex randomly chosen from the next frame.

All the running times reported here are in seconds, and we only report the user CPU time. We generated five instances for each problem family of specified size, using different random seeds. Each number shown in the tables is the averaged time over five runs.

Table 3: Results on Genrmf-Long

Running time (sec)								
n	m	FW	FW$^-$	HYBRID	SFM3	SFM8	LEX2	PR
63	222	0.03	0.04	4.02	10.95	19.11	1.42	1.24
126	453	0.28	0.36	70.82	280.52		53.36	23.28
256	1008	3.77	3.79	7376.47			3209.70	3507.49
525	2180	35.02	46.05					
1008	4332	275.46	366.21					

Table 4: Results on Genrmf-Wide

Running time (sec)								
n	m	FW	FW$^-$	HYBRID	SFM3	SFM8	LEX2	PR
75	290	0.04	0.05	3.34	20.58	64.75	4.19	3.48
147	602	0.41	0.41	55.99	749.13		141.49	89.87
324	1395	4.07	4.59	4265.14			9607.36	2433.57
576	2544	19.97	27.17					
1024	4608	171.13	172.52					

We used GENRMF to produce two kinds of networks as follows:

- *Genrmf-Long*. The number of vertices of a generated graph is $n = 2^x$. The parameters are $a = 2^{x/4}$ and $b = 2^{x/2}$.
- *Genrmf-Wide*. The number of vertices of a generated network is $n = 2^x$. The parameters are $a = 2^{2x/5}$ and $b = 2^{x/5}$.

We used the submodular function minimization algorithms to compute minimum cuts. The running times for the computation are shown in Table 3 and Table 4, and numbers

of generated extreme bases in Table 5 and Table 6. Here FW$^-$ stands for FW without the acceleration indicated in the last paragraph of Section 3.3. Figure 1 and Figure 2, respectively, represent Table 3 and Table 4 except for FW$^-$ and SFM8.

Table 5: Results on Genrmf-Long

Number of generated extreme bases							
n	m	FW	HYBRID	SFM3	SFM8	LEX2	PR
63	222	**86**	**23029**	**28288**	**207527**	**526**	**1918**
126	453	**193**	**112328**	**140678**		**2280**	**5732**
256	1008	**479**	**690950**			**8757**	**14605**
525	2180	**1096**					
1008	4332	**2310**					

Table 6: Results on Genrmf-Wide

Number of generated extreme bases							
n	m	FW	HYBRID	SFM3	SFM8	LEX2	PR
75	290	**91**	**13564**	**18507**	**507757**	**756**	**3519**
147	602	**193**	**80240**	**66346**		**3878**	**7694**
324	1395	**413**	**661802**			**14066**	**20553**
576	2544	**646**					
1024	4608	**1235**					

For the Genrmf-Long networks LEX2 and PR were faster than HYBRID. However, for the Genrmf-Wide networks LEX2 was slower than HYBRID. In both cases FW outperformed the others (except for FW$^-$). We have achieved about 10 to 25% run-time reduction from FW$^-$ to FW for the Genrmf-Long networks while no significant reduction for the Genrmf-Wide networks.

Figures 3, 4, 5, and 6 show sample behaviors of iteration vs. duality gap for Genrmf-Long with $n = 63$ and $m = 222$. Here, one iteration means a generation of a new extreme base.

5. Concluding Remarks

The computational results on submodular function minimization have shown that the minimum-norm-base algorithm FW runs very fast, which allures us to conjecture that FW is (strongly) polynomial. It is, however, open to determine the complexity of FW for submodular function minimization.

Acknowledgments

We are very grateful to Satoru Iwata for providing us with his programs of the SFM algorithms. Thanks are also due to Nobuyuki Tsuchimura and Satoko Moriguchi for their useful comments and computational experiments on earlier versions of our FW algorithm, which improved our code FW. The present research was supported partly by a Grant-in-Aid from the Ministry of Education, Culture, Sports, Science and Technology of Japan and by Japan International Cooperation Agency.

References

[1] J. Edmonds: Submodular functions, matroids, and certain polyhedra. *Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications* (R. Guy, H. Hanani, N. Sauer and J. Schönheim, eds., Gordon and Breach, New York, 1970), pp. 69–87; also in: *Combinatorial Optimization—Eureka, You Shrink!* (M. Jünger, G. Reinelt, and G. Rinaldi, eds., Lecture Notes in Computer Science **2570**, Springer, Berlin, 2003), pp. 11–26.

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

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

[4] S. Fujishige: Submodular systems and related topics. *Mathematical Programming Study* **22** (1984) 113–131.

[5] S. Fujishige: Submodularity and polyhedra. 4th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications (Budapest, June 3–6, 2005).

[6] S. Fujishige: *Submodular Functions and Optimization*, (Second Edition) (Annals of Discrete Mathematics **58**) (Elsevier, Amsterdam, 2005).

[7] B. von Hohenbalken: A finite algorithm to maximize certain pseudoconcave functions on polytopes. *Mathematical Programming* **8** (1975) 189–206.

[8] S. Isotani and S. Fujishige: Submodular function minimization: Computational experiments. Unpublished manuscript, 2003.

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

[10] S. Iwata: Submodular function minimization. *Mathematical Programming* **112** (2008) 45–64.

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

[12] S. Iwata and J. B. Orlin: A simple combinatorial algorithm for submodular function minimization. ACM-SIAM Symposium on Discrete Algorithms (SODA09) (January 4-6, 2009, New York, NY).

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

[14] S. Moriguchi and N. Tsuchimura: Discrete L-/M-convex function minimization based on continuous relaxation. Mathematical Engineering Technical Reports METR 2007-59, Department of Mathematical Informatics, Graduate School of Information Science and Technology, The University of Tokyo, November 2007.

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

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

[17] J. Vygen: A note on Schrijver's submodular function minimization algorithm. *Journal of Combinatorial Theory* **B88** (2003) 399–402.

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

[19] The First DIMACS international algorithm implementation challenge: The core experiments, 1990. Available at ftp://dimacs.rutgers.edu/pub/netflow/general-info/core.tex .

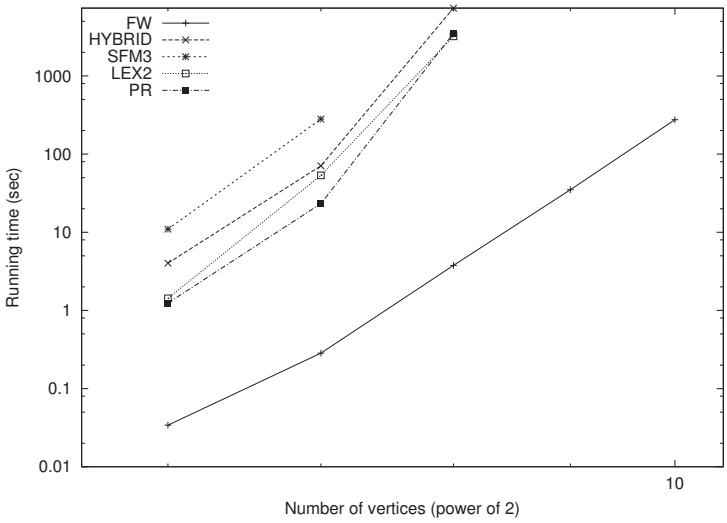

Figure 1: The number of vertices vs. running time on Genrmf-Long

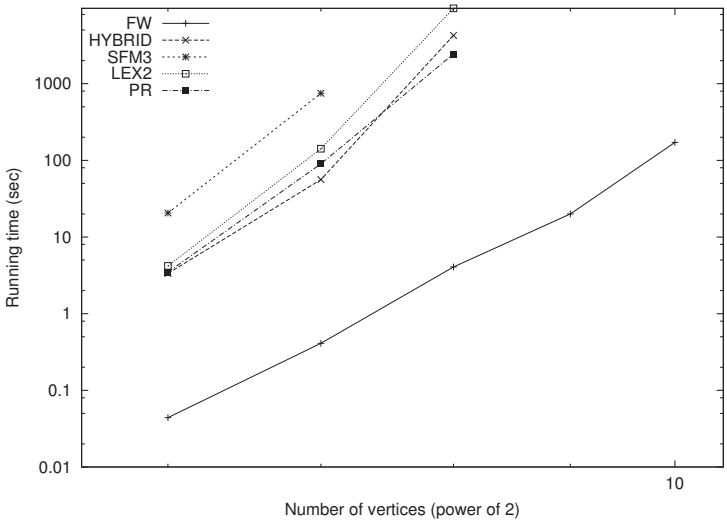

Figure 2: The number of vertices vs. running time on Genrmf-Wide

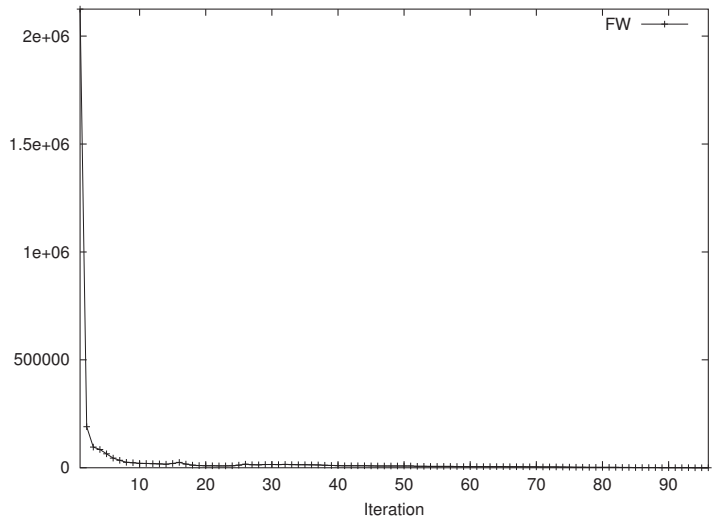

Figure 3: The number of iterations vs. duality gap of FW

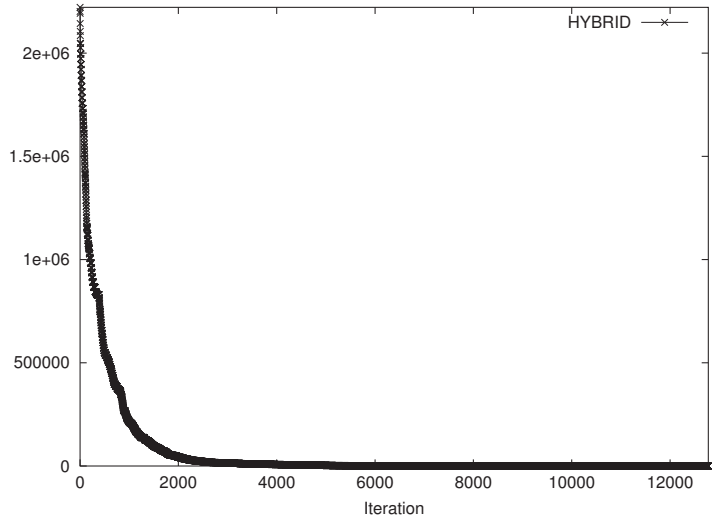

Figure 4: The number of iterations vs. duality gap of HYBRID

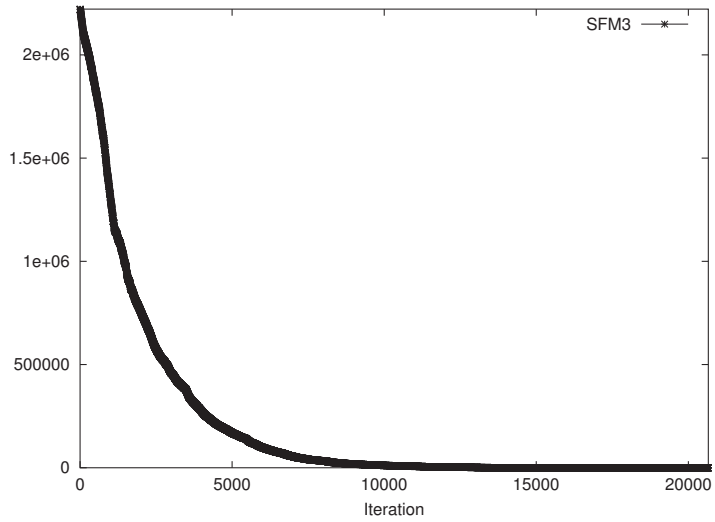

Figure 5: The number of iterations vs. duality gap of SFM3

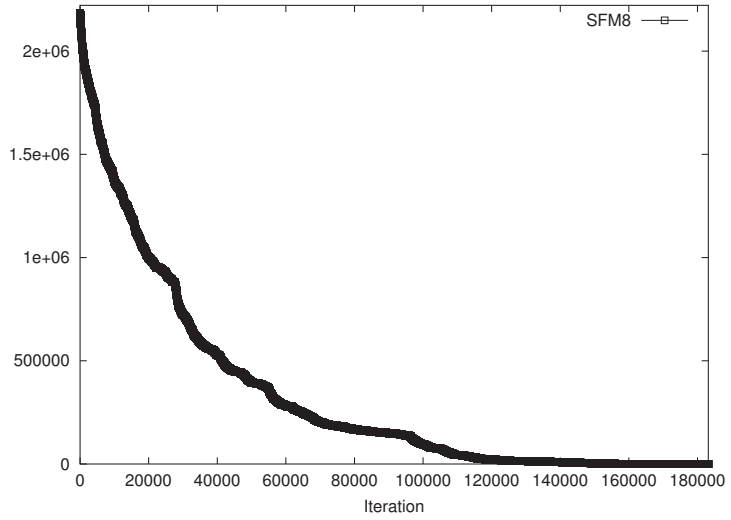

Figure 6: The number of iterations vs. duality gap of SFM8

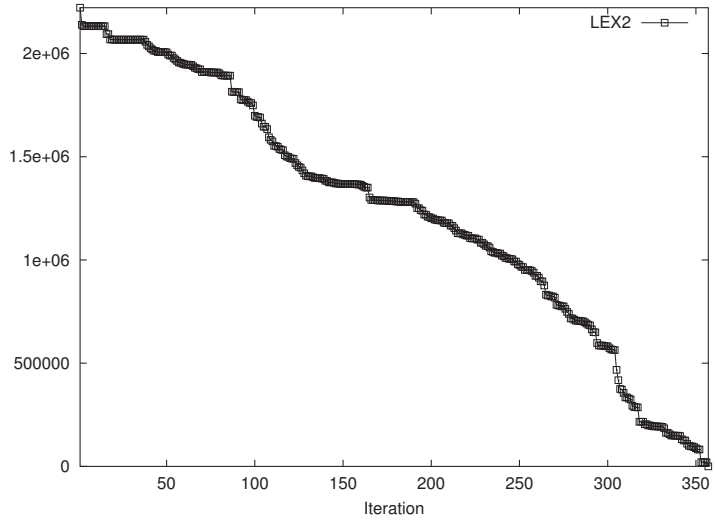

Figure 7: The number of iterations vs. duality gap of LEX2