Mathematical Programming 82 (1998) 3–12

Minimizing symmetric submodular functions [1]

Maurice Queyranne [2]

Faculty of Commerce and Business Administration, The University of British Columbia, Vancouver, BC, Canada V6T 1Z2

Received 8 September 1995; accepted 6 December 1996

Abstract

We describe a purely combinatorial algorithm which, given a submodular set function f on a finite set V, finds a nontrivial subset A of V minimizing $f[A]+f[V\setminus A]$. This algorithm, an extension of the Nagamochi–Ibaraki minimum cut algorithm as simplified by Stoer and Wagner [M. Stoer, F. Wagner, A simple min cut algorithm, Proceedings of the European Symposium on Algorithms ESA '94, LNCS 855, Springer, Berlin, 1994, pp. 141–147] and by Frank [A. Frank, On the edge-connectivity algorithm of Nagamochi and Ibaraki, Laboratoire Artémis, IMAG, Université J. Fourier, Grenbole, 1994], minimizes any symmetric submodular function using $O(|V|^3)$ calls to a function value oracle.

Keywords: Symmetric submodular function minimization; Submodular function minimization; Symmetric submodular functions; Submodular functions; Submodular systems

1. Introduction

Let V be a finite set, and $n \doteq |V|$. A real-valued function $f: 2^V \mapsto \mathbb{R}$ is *submodular* if and only if it satisfies the submodular inequality,

$$f[A\cup B]+f[A\cap B]\leqslant f[A]+f[B] \tag{1}$$

for all subsets $A, B \subseteq V$. The pair (V,f) is called a *submodular system*. Submodularity is one of the deepest and most useful properties in combinatorial optimization, see [1–4,31] for further discussion of submodularity and related properties. We assume throughout this paper that the function f is given by a *value oracle* which, for any input subset $A \subseteq V$, returns the numerical value of $f[A]$. The problem of finding a subset A which minimizes $f[A]$ is important in combinatorial optimization, see the above references. It is solved in polynomial time by Grötschel et al. [5] with the el-

[1] A preliminary version of this paper was presented at the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) in January 1995. This research was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

[2] E-mail: maurice.queyranne@commerce.ubc.ca.

0025-5610/98/$19.00
Published by Elsevier Science B.V.
PII S0025-5610(97)00094-4

lipsoid method, using $O(n^4)$ value oracle calls [6]. Cunningham [7] proposes a more combinatorial algorithm, with running time $O(Mn^3 \log [Mn])$, where $M \geqslant \max_S |f[S]|$; that is, the algorithm is pseudo-polynomial. Finding a purely combinatorial and polynomial time algorithm for this problem remains one of the important open questions in combinatorial optimization. In this paper we solve this problem for the class of symmetric submodular functions. We present a polynomial time algorithm which minimizes such functions using $O(n^3)$ oracle calls.

A set function $s: 2^V \mapsto \mathbb{R}$ is *symmetric* if

$$s[A] = s[V \setminus A] \quad \text{for all } A \in V. \tag{2}$$

Given two elements t and u of V, a subset A of V *separates t from u* if exactly one of t or u is in A. Cheng and Hu [8] show how to minimize a general symmetric function s using $n-1$ calls to an oracle which, given any two elements t and u of V, finds a subset A separating t and u and with minimum value $s[A]$. In this paper, we do not assume such a minimization oracle, but we require the function to be submodular. We also require the minimizer A^* of s to be a nontrivial subset of V, that is, $\emptyset \subset A^* \subset V$; otherwise, the problem would be trivial since the symmetry and submodularity of s imply $s[\emptyset] = s[V] \leqslant s[A]$ for all $A \subseteq V$.

Symmetric submodular functions were introduced in [9] by Fujishige, who provides a decomposition theory for such functions. A canonical example of a symmetric submodular function is the *cut function* of an undirected network. Given an undirected graph $G = (V, E)$ and an edge cost function $c: E \mapsto \mathbb{R}$, the cut function f is defined by

$$f[A] \doteq \sum \{c[e] \in E: \ e \text{ has one endpoint in } A \text{ and one in } V \setminus A\}.$$

This function is symmetric and, when all $c[e]$ are nonnegative, submodular. Well-known algorithms for finding a minimum cut separating specified nodes s and t are based on network flow techniques; see e.g. [10]. Fixing s and repeating this for all $n-1$ choices of node t $(t \neq s)$ yields a (globally) minimum cut. Hao and Orlin, [11] present a fast implementation which also applies to directed networks. Gomory and Hu [12] show how to choose $n-1$ pairs of terminals to construct a tree (V, T), now known as a cut-equivalent tree (or Gomory–Hu tree), satisfying the following properties: (i) every tree edge has a value equal to the minimum capacity of a cut (in the original network) separating its endpoints; and (ii) the cut determined in the tree by a minimum value edge on the path connecting *any* two nodes s and t is a minimum cut separating s and t in the original network. Goemans and Ramakrishnan [30] point out that such a cut-equivalent tree exists for any symmetric submodular function, and show how to use it to minimize the function over certain families of subsets. The cut-equivalent tree is related to, but distinct from, the decomposition trees in [8,9].

Nagamochi and Ibaraki [13,14] provide a novel algorithm for finding a globally minimum cut in an undirected network. Their algorithm was primarily developed for unit capacities (all $c[e] = 1$) for which case it runs in $O(nm)$ time, where

$m \doteq |E|$. Its extension to arbitrary nonnegative edge costs was noted by Nagamochi and Ibaraki [14], and simplified independently by Stoer and Wagner [15] and by Frank [16], see also Subramanian's expository article [32]. Nagamochi et al. [17] report on computational experiments indicating that this algorithm is considerably faster than the previous fastest known algorithm due to Padberg and Rinaldi [18]. See [19] for additional computational results. The algorithm presented in this paper is an extension of the Nagamochi–Ibaraki algorithm to arbitrary symmetric submodular functions. It reduces precisely to the Stoer–Wagner and Frank versions of the Nagamochi–Ibaraki algorithm when applied to an undirected network cut function; see also Section 9.8 in Narayanan's monograph [31] for a presentation of the present algorithm closely following the notation of Stoer–Way

We now discuss some other examples of symmetric submodular functions. First, recall that the cut function of a *directed* network $N = (V, A, c)$ is also submodular when the arc costs are nonnegative. Gupta showed ([20], Lemma 2.1) that it is symmetric if and only if the network is *pseudosymmetric*, that is, if its arc cost function $c : A \mapsto \mathbb{R}$ satisfies

$$\sum\{c[ij]:\ i \text{ with } ij \in A\} = \sum\{c[jk]:\ k \text{ with } jk \in A\}$$

for all $j \in V$. The cut function has also been extended by Granot et al. [21] to mixed disconnecting sets, consisting of nodes and edges, in networks that may include both directed and undirected edges. The resulting function is submodular and, when the network has only undirected edges, it is also symmetric.

Wagner [22] pointed out the following hypergraph extension of the cut function. Let $\mathcal{H}$ be a family of subsets of V and a weight function w: $\mathcal{H} \mapsto \mathbb{R}$. Define the set function $f : 2^V \mapsto \mathbb{R}$ as follows:

$$f[A] = \sum\{w[H]:\quad H \in \mathcal{H},\quad H \cap A \neq \emptyset \text{ and } H \cap (V \setminus A) \neq \emptyset\}$$

for all $A \subset V$. This function is symmetric and, if $w \geqslant 0$, submodular (see also Example 9.8.1 (iii) in [31], p. 354). Subsequent to the conference version [23] of this paper, Wagner [22] has extended the Stoer–Wagner algorithm to this case, using the special hypergraph structure to speed up the algorithm.

A large class of symmetric submodular functions not related to networks may be constructed by the following standard process. Let f be an arbitrary submodular function and define its *symmetric part* s_f and *antisymmetric part* a_f as follows:

$$s_f[A] \doteq \tfrac{1}{2}(f[A] + f[V \setminus A]) \quad \text{and} \quad a_f[A] \doteq \tfrac{1}{2}(f[A] - f[V \setminus A]) \tag{3}$$

for all $A \subseteq V$. Then $f = s_f + a_f$ and s_f is symmetric. Further, s_f is submodular if f is. Note that, if f is symmetric, then $s_f = f$ and $a_f = 0$. In fact, the algorithm presented below minimizes the symmetric part s_f of any given submodular function f by directly using the f function values.

Cunningham [24] defines the *connectivity function* c_f of the function f by

$$c_f[A] \doteq f[A] + f[V \setminus A] - f[V]$$

for all $A \subseteq V$. It is clear that a nontrivial subset A of V minimizes c_f if and only if it minimizes s_f. Cunningham uses the connectivity function c_f to develop a decomposition theory for arbitrary submodular functions f, which extends Fujishige's decomposition theory [9] for symmetric submodular functions. Special cases include the decomposition of cut functions, graph functions and matroid functions, see [24] for details.

Baïou et al. [25] consider *partition inequalities*

$$x[\delta(S_1, \ldots, S_p)] \geqslant ap + b,$$

where $(S_1, \ldots, S_p)$ form a partition of the node set V of an undirected graph (V, E); $\delta(S_1, \ldots, S_p)$ is the set of all edges in E with endnodes in different sets of the partition; $x[e] \geqslant 0$ are given edge weights; and $a > 0$ and b are given constants. Partition inequalities with $a = 1$ and $b = -1$ define the dominant of the spanning tree polytope. They also arise as valid inequalities for facets of polyhedra related to k-connectivity problems, including various minimum cost k-edge or k-node connected subgraph or spanning subgraph problems, the Travelling Salesman Problem, and a network loading problem in communication network design; see [25] for details and references. Baïou et al. [25] show that, given $x \geqslant 0, a > 0$ and $b > -1$, a most violated partition inequality may be found by minimizing the symmetric part s_f of the submodular function f defined by

$$f[S] = \min \left\{ \sum_{i=1}^{k} (\delta[T_i] - 1): \quad k \geqslant 1 \text{ and } (T_1, \ldots, T_k) \text{ form a partition of } S \right\},$$

and $\delta[T]$ is the set of all edges in E with exactly one endpoint in T. They use minimum cut based algorithms of Cunningham [26] and Barahona [27] when $b/a \leqslant -1$, and the algorithm from the present paper when $b/a > -1$.

The contents of this paper are as follows. In Section 2 we define pendent pairs and present an algorithm for finding a pendent pair for a submodular system (V, f), using $O(n^2)$ f-value oracle calls. In Section 3 we use pendent pairs to find a nontrivial subset A or V which minimizes the symmetric part s_f of a submodular function f. Our algorithm recursively uses the pendent pair algorithm $n - 1$ times. It also uses the fact that it is easy to enforce, for symmetric submodular functions, the requirement that an optimum set must not separate any two given elements of V. Finally, in Section 4, we discuss some related problems.

2. Pendent pairs

In this section, we define pendent pairs for a submodular system (V, f), and we present an algorithm which finds a pendent pair using $O(n^2)$ calls to the f-value oracle. Throughout the paper, we use the simplified notations $A + u \doteq A \cup \{u\}$ when $u \notin A$; $A - u \doteq A \setminus \{u\}$; and $f[u] \doteq f[\{u\}]$.

An ordered pair (t, u) of elements of V is called a *pendent pair for* (V, f) if $\{u\}$ has minimum s_f value among all subsets of V which separate t from u, that is, if

$$s_f[u] = \min \{s_f[U]: \quad U \subset V, t \notin U \text{ and } u \in U\}. \tag{4}$$

The existence of pendent pairs follows from that of a cut-equivalent tree [30] for symmetric sub-modular functions; for example, take any leaf and its neighbour in a cut-equivalent tree. It will also follow from the constructive algorithm below.

We now describe an algorithm for finding a pendent pair in V. The algorithm constructs an ordering $(v_1, v_2, \ldots, v_n)$ of the elements in V, where the first element v_1 may be specified as an (arbitrary) input element $x \in V$. Let $W_0 \doteq \emptyset$ and, for $i = 1, \ldots, n$, let $W_i \doteq \{v_j: j \leqslant i\}$. At any step i $(i = 1, \ldots, n-1)$, the algorithm maintains a data structure Q that contains all the elements $u \in V \setminus W_i$ along with the key value

$$\text{key}[u] \doteq f[W_i + u] - f[u]. \tag{5}$$

The function UPDATE[key] updates (or recomputes) these key values after the counter i and consequently the set W_i have been incremented. The function EXTRACT-MIN[Q] returns an element from Q with least key value, and deletes it from the data structure.

Algorithm PENDENT-PAIR[V, f, x]
1. $v_1 \leftarrow x$
2. **for** $i = 1$ to $n - 1$
3. **do** UPDATE[key]
4. $v_{i+1} \leftarrow$ EXTRACT-MIN[Q]
5. **return** (v_{n-1}, v_n)

Lemma 1. *If (V, f) is a submodular system then for all $i = 1, \ldots, n-1$, all $y \in V \setminus W_i$ and all $X \subseteq W_{i-1}$,*

$$f[W_i] + f[y] \leqslant f[W_i \setminus X] + f[X + y]. \tag{6}$$

Proof. The lemma trivially holds for $i = 1$. By induction, assume that Eq. (6) holds for all $i = 1, \ldots, k-1$. Consider any $u \in V \setminus W_k$, and $S \subseteq W_{k-1}$. From Step 4 of the algorithm and the definition (5), we have

$$f[W_k] + f[u] \leqslant f[W_{k-1} + u] + f[v_k]. \tag{7}$$

Let j be the smallest integer such that $S \subseteq W_{j-1}$.

If $j = k$, then $v_{k-1} \in S$ and $W_{k-1} \setminus S \subseteq W_{k-2}$. Therefore,

$$\begin{aligned} f[W_k \setminus S] + f[S + u] &= f[(W_{k-1} \setminus S) + v_k] + f[S + u] \\ &\geqslant f[W_{k-1}] + f[v_k] - f[S] + f[S + u] \\ &\geqslant f[W_{k-1} + u] + f[v_k] \\ &\geqslant f[W_k] + f[u], \end{aligned}$$

where the inequalities follow respectively from the inductive assumption (6) with $i = k-1, y = v_k$ and $X \leftarrow W_{k-1} \setminus S$; the submodular inequality (1) with $A = W_{k-1}$ and $B = S + u$; and inequality (7).

If $j \leqslant k-1$ then $v_{j-1} \in S$ and none of $v_j, \ldots, v_k$ is in S. Since $\{v_j, \ldots, v_k\} = W_k \setminus W_{j-1}$, we have,

$$\begin{aligned} f[W_k \setminus S] + f[S+u] &= f[(W_{j-1} \setminus S) \cup (W_k \setminus W_{j-1})] + f[S+u] \\ &\geqslant f[(W_{j-1} \setminus S) \cup (W_k \setminus W_{j-1})] + f[W_j] - f[W_j \setminus S] + f[u] \\ &\geqslant f[W_k] + f[u], \end{aligned}$$

where the inequalities follow respectively from the inductive assumption (6) with $i = j, y = u$ and $X = S$; and the submodular inequality (1) with $A = (W_{j-1} \setminus S) \cup (W_k \setminus W_{j-1})$ and $B = W_j$. Thus Eq. (6) holds for $i = k$ and any $y = u \in V \setminus W_i$ and $X = S \subseteq W_{i-1}$. The proof is complete. □.

Theorem 2. *The algorithm PENDENT-PAIR returns a pendent pair* (v_{n-1}, v_n) *after using* $O(n^2)$ *calls to the f-value oracle and* $O(n^2)$ *other operations.*

Proof. Lemma 1 with $i = n-1$ (and therefore $y = v_n$) implies that the pair (v_{n-1}, v_n) is a pendent pair. There are $n-1$ calls to *UPDATE*, each requiring $O(n)$ oracle calls. In a simple implementation, the data structure Q is a linked list, and each of the $n-1$ calls to *EXTRACT-MIN* requires $O(n)$ operations. □

3. Using pendent pairs for finding an optimum set

We now present a recursive algorithm using pendent pairs to find a nontrivial subset A of V minimizing the symmetric part s_f of a submodular function f. Note that the problem would be trivial without the restriction that A be a nontrivial subset of V, since the submodularity of f implies that $s_f[\emptyset] = s_f[V] \leqslant s_f[A]$ for all $A \subseteq V$.

The algorithm is based on the following simple observation. If (t, u) is a pendent pair for (V, f), then an optimum subset A either separates t from u, in which case $s_f[A] = s_f[u]$, or it does not separate t from u. For the latter case, we may "merge" elements t and u together as a single element denoted, say, by t. Thus we are lead to consider the system (V', f') where $V' \doteq V - u$ and $f' : 2^{V'} \mapsto \mathbb{R}$ is defined by

$$f'[A'] \doteq \begin{cases} f[A'+u] & \text{if } t \in A', \\ f[A'] & \text{otherwise.} \end{cases}$$

It is easy to verify that the function f' is submodular when f is. Thus (V', f') is a submodular system, $s_{f'}[A-u] = s_f[A]$ whenever both $t, u \in A$, and the minimization of $s_{f'}$ can be performed recursively in exactly the same fashion as for (V, f). After any number of successive mergings, every element of the current set V' represents a subset of the original set V. In effect, V' defines a partition of V. After $n-1$ steps,

the current set V' has a single element, so there is no nontrivial subset and the recursive process stops. At this point, we have identified $n-1$ candidate subsets A^h $(h=1,\ldots,n-1)$ along with their values $s_f[A^h]$. A globally optimum set is then an A^h for which $s_f[A^h]$ is minimum. We outline below an algorithm implementing this recursive scheme.

In this algorithm, the current submodular system (V',f') defines a partition $\{S_j: j \in V'\}$ of V. Initially $V'=V$ and each $S_j=\{j\}$. The function $f': 2^{V'} \mapsto \mathbb{R}$ is defined by

$$f'[A'] \doteq f[\cup_{j\in A'} S_j] \tag{8}$$

for all $A' \subseteq V'$. It is well known, e.g. [4], that (V',f') indeed forms a submodular system when (V,f) does.

Algorithm OPTIMAL-SET$[V,f,x]$
1. **for** $j \in V$ **do** $S_j \leftarrow \{j\}$; let $V' \leftarrow \{S_j: j \in V\}$; throughout, f' is as defined in (8) above,
2. **for** $h=1$ to $|V|-1$,
3. **do** $(S_t,S_u) \leftarrow$ PENDENT-PAIR$[V',f',S_x]$,
4. store $A^h \leftarrow S_u$ and $s^h \leftarrow s_{f'}[S_u]$,
5. update $S_t \leftarrow S_t \cup S_u$ and $V' \leftarrow V' \setminus \{S_u\}$,
6. Choose $i \in \arg\min\{s^h: h=1,\ldots,n-1\}$,
7. **return** A^i.

Theorem 3 follows immediately from the preceding observation, and from Theorem 2. Note that, in counting operations, we aggregate as a single oracle call to f all the operations involved in Eq. (8), including the expansion of subset A' into $\cup_{j\in A'} S_j$.

Theorem 3. *When (V,f) is a submodular system, the algorithm OPTIMAL-SET returns a nontrivial subset of V which minimizes s_f, using $O(n^3)$ calls to the f-value oracle, and $O(n^3)$ other operations.*

If f is a *symmetric* submodular function, then $s_f = f$ and the algorithm described in the previous sections directly minimizes f over all nontrivial subsets of V. Thus we have described a purely combinatorial algorithm which minimizes a symmetric submodular function using $O(n^3)$ calls to a function value oracle, improving over the $O(n^4)$ oracle calls used by the ellipsoid-based, general submodular minimization algorithm of [5]. In some applications, one may reduce the running time of the algorithm by efficiently updating previously computed function values, and by using a more sophisticated data structure for Q. This is the case when f is the cut function of an undirected network, see [14,15] for details.

4. Related problems

The symmetric submodular s, t-cut problem is as follows: given a submodular function f and two distinct elements $s,\ t \in V$, find a subset separating s from t and with

minimum s_f-value. The existence of a combinatorial algorithm for this problem would immediately extend to several related problems. Indeed, the cut-equivalent tree for the symmetric function s_f can be constructed by solving $n-1$ symmetric submodular s,t-cut problems. Extending results of Padberg and Rao [28], Grötschel et al. [5], and Gabow et al. [29] on network cuts, Goemans and Ramakrishnan [30] show that the cut-equivalent tree may then be used to find an s_f-minimum subset in certain subset families, in particular so-called T-even and T-odd subsets. In addition, Cunningham's decomposition [24] for a submodular function f can be constructed by solving $O(n^3)$ symmetric submodular s,t-cut problems involving the function s_f.

The next result shows that the symmetric submodular s,t-cut problem is just as hard as that of minimizing a general (i.e., nonsymmetric) submodular function. Let $f: 2^{V'} \mapsto \mathbb{R}$ be a submodular function, and M a positive number such that $-M < \min_S f[S]$ and $-M < \min_S f[S] - \max_S f[S]$. (We may take $M = 1 + 2^{\beta+1}$ where β is a given upper bound on the size of $f[S]$ for any S, as in [5], or obtain M from $O(|V'|)$ calls to the f-oracle.) Define $V \doteq V' \cup \{s,t\}$ for $s,t \notin V'$ and the function $g: 2^V \mapsto \mathbb{R}$ as follows:

$$g[A] = \begin{cases} f[A-s] & \text{if } s \in A \text{ and } t \notin A, \\ -M \mid A \mid & \text{if } s,t \in A, \text{ and} \\ g[V \setminus A] & \text{if } s \notin A. \end{cases}$$

Lemma 4. *Let $f: 2^{V'} \mapsto \mathbb{R}$ be a submodular function, and V and g as just defined. Then*
1. *g is symmetric and submodular; and*
2. *if A is a g-minimum subset of V with $s \in A$ and $t \notin A$, then $A - s$ is an f-minimum subset of V'.*

Proof. The symmetry of g immediately follows from the third part of its definition, and statement (2) from the first part. Hence we only need to prove that g satisfies the submodular inequality (1). This is immediate if $A \cap \{s,t\} = B \cap \{s,t\}$, or if one of A or B contains the other. Otherwise, first assume that $s \in A$ and $t \notin A$. If $s \notin B$ and $t \in B$, then

$$g[A \cup B] + g[A \cap B] \leqslant -2M < f[A-s] + f[(V \setminus B) - s] = g[A] + g[B],$$

whereas if $s,t \in B$ (and $A \not\subseteq B$), then

$$g[A \cup B] - g[B] \leqslant -M < f[A-s] - f[(A \cap B) - s] = g[A] - g[A \cap B].$$

If, on the other hand, $s,t \in A$ and $s,t \notin B$ then $g[A \cup B] \leqslant g[A]$ and $g[A \cap B] \leqslant g[B]$. Finally, the other cases follow by symmetry. □

We now turn to a more general problem for which no efficient combinatorial algorithm is known at the present: given a submodular function f and a *modular* function m also defined on V, find a subset of V with minimum $(s_f + m)$-value. (Recall that a modular function f satisfies condition (1) as an equality for all subsets A,B

of the ground set V.) The symmetric submodular s,t-cut problem is a special case where $m = M\chi_t - M\chi_s$ with M a large positive constant (as in the proof of Lemma 4) and $\chi_a[A] = 1$ if $a \in A$ and 0 otherwise. Minimizing $s_f + m$ for a general modular function m is equivalent to the *membership problem* for the submodular polyhedron P_{s_f} (and for $P_{s_f+\mu}$ for any modular function μ) where

$$P_f \doteq \{x \in V:\ x[S] \leqslant f[S] \text{ for all } S \subseteq V\}$$

for any set function f defined on V (see, e.g. [4]). Note, however, that pendent pairs need not exist for a submodular function $g = s_f + m$ with s_f symmetric submodular and m modular. Indeed, consider the case where $V = \{1,\ldots,n\}$ with n even, say, $n = 2p$; $s_f = 0$; and $m = \sum_{v\in S} m_v\chi_v$ with $m_v < 0$ for $1 \leqslant v \leqslant p$, and $m_v > 0$ for $p+1 \leqslant v \leqslant n$. For any distinct $s,t, \in V$, the unique subsets separating s from t and with minimum g-value are $S_{st} = (\{1,\ldots,p\} - t) + s$ and its complement. Thus $|S_{st}| \geqslant p-1$ and $|V \setminus S_{st}| \geqslant p-1$ and therefore there are no pendent pairs for g whenever $n \geqslant 6$. Note also that there are $(p+1)^2 = O(n^2)$ distinct separating subsets S_{st} for g, and thus their structure must be more complex than that of a cut-equivalent (Gomory–Hu) tree.

By Lemma 4 above, the problems discussed in this section are equivalent to that finding an efficient combinatorial algorithm for minimizing a general submodular function, which remains unsolved at present.

Acknowledgements

The author thanks Professors Satoru Fujishige (Osaka University) and Ashok Subramanian (Indian Institute of Science, Bangalore) for their comments on earlier version of this papar. In particular, questions raised by Subramanian led to the results in Section 4."

References

[1] D.M. Topkis, Minimizing a submodular function on a lattice, Operations Research 26 (1978) 305–321.
[2] L. Lovász, Submodular functions and convexity, in: A. Bachem et al. (Eds.), Mathematical Programming – The State of The Art, Springer, Berlin, 1983 pp. 235–257.
[3] S. Fujishige, Submodular systems and related topics, Mathematical Programming Study 22 (1984) 113–131.
[4] S.Fujishige, Submodular Functions and Optimization, North-Holland, Amsterdam, The Netherlands, 1991.
[5] M. Grötschel, L. Lovász, A. Schrijver, Geometric Algorithms and Combinatorial Optimization, Springer, Berlin, 1988.
[6] L. Lovász, Personal communication, Electronic mail, 1996.
[7] W.H. Cunningham, On submodular function minimization, Combinatorica 5 (1985) 185–192.
[8] C.K. Cheng, T.C. Hu, Maximum concurrent flows and minimum cuts, Algorithmica 8 (1992) 233–249.

[9] S. Fujishige, Canonical decomposition of symmetric submodular systems, Discrete Applied Mathematics 5 (1983) 175–190.
[10] L.E. Ford Jr., D.R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, NJ, 1962.
[11] J. Hao, J.B. Orlin, A faster algorithm for finding a minimum cut in a graph, Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms, 1992, pp. 165–174.
[12] R.E. Gomory, T.C. Hu, Multiterminal network flows, SIAM Journal on Applied Mathematics 9 (1961) 551–570.
[13] H. Nagamochi, T. Ibaraki, Linear time algorithms for finding a sparse k-connected spanning subgraph of a k-connected graph, Algorithmica 7 (1992) 583–596.
[14] H. Nagamochi, T. Ibaraki, Computing edge connectivity in multigraphs and capacitated graphs, SIAM Journal on Discrete Mathematics 5 (1992) 54–66.
[15] M. Stoer, F. Wagner, A simple min cut algorithm, Proceedings of the European Symposium on Algorithms ESA, '94, LNCS 855, Springer, Berlin, 1994, pp. 141–147.
[16] A. Frank, On the edge-connectivity algorithm of Nagamochi and Ibaraki, Laboratoire Artémis, IMAG, Université J. Fourier, Grenoble, 1994.
[17] H. Nagamochi, T. Ono, T. Ibaraki, Implementing an efficient minimum capacity cut algorithm, Mathematical Programming 67 (1994) 325–341.
[18] M.W. Padberg, G. Rinaldi, An Efficient algorithm for the minimum capacity cut problem, Mathematical Programming 47 (1990) 19–36.
[19] K. Mehlhorn, C. Uhrig, The minimum cut algorithm of Stoer and Wagner, Max Planck, Institute for Computer Science, Saarbrücken, Germany, 1994.
[20] R.P. Gupta, On flows in pseudosymmetric networks, SIAM Journal on Applied Mathematics (1966) 215–225.
[21] F. Granot, M. Penn, M. Queyranne, Disconnecting sets in single and two-terminal-pair networks, Networks 27 (1996) 117–123.
[22] F. Wagner, Personal communication, Berlin, 1995.
[23] M. Queyranne, A combinatorial algorithm for minimizing symmetric submodular functions, Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 1995, pp. 98–101.
[24] W.H. Cunningham, Decomposition of submodular functions, Combinatorica 3 (1983) 53–68.
[25] M. Baïou, F. Barahona, A.R. Mahjoub, Separation of partition inequalities, Laboratoire SPO, Département d'Informatique, Université de Bretagne Occidentale, Brest, France, 1996.
[26] W.H. Cunningham, Optimal attack and reinforcement of a network, Journal of the ACM 32 (1985) 549–561.
[27] F. Barahona, Separating from the dominant of the spanning tree polytope, Operations Research Letters 12 (1992) 201–203.
[28] M.W. Padberg, M.R. Rao, Odd-minimum cut-sets and b-matchings, Mathematics of Operations Research 7 (1982) 67–80.
[29] H.N. Gabow, M.X. Goemans, D.P. Williamson, An efficient approximation algorithm for the survivable network design problem, Mathematical Programming, this issue.
[30] M.X. Goemans, V.S. Ramakrishnan, Minimizing submodular functions over families of sets, Combinatorica 15 (1995) 499–513.
[31] H. Narayanan, Submodular Functions and Electrical Networks, North-Holland, Amsterdam, 1997.
[32] A. Subramanian, Two recent algorithms for the global minimum cut problem, , SIGACT News 26 (2) (1995) 78–87.