COMBINATORICA 5 (3) (1985) 185—192

ON SUBMODULAR FUNCTION MINIMIZATION

WILLIAM H. CUNNINGHAM*

Received 4 December 1984

Earlier work of Bixby, Cunningham, and Topkis is extended to give a combinatorial algorithm for the problem of minimizing a submodular function, for which the amount of work is bounded by a polynomial in the size of the underlying set and the largest function value (not its length).

1. Introduction

Let E be a finite set. A real-valued function g defined on subsets of E is *submodular* if

$$g(A)+g(B) \geq g(A \cup B)+g(A \cap B)$$

for all $A, B \subseteq E$. Familiar examples of submodular functions include *cut functions;* $g(A)$ is the sum of (positive) capacities of edges leaving A, where E is the vertex-set of a flow network, and *matroid rank functions:* $g(A)$ is the rank of A, where E is the set of elements of a matroid. The problem of *minimizing* a submodular function includes as special cases, those of finding the minimum cut in a flow network, testing whether a given vector is a convex combination of incidence vectors of independent sets of a matroid, or testing whether a given vector is a feasible solution to any of a number of combinatorial optimization models, such as polymatroid intersection. Other applications and a description of the close connection between submodular functions and convex functions can be found in the survey paper of Lovász [9].

The general algorithmic problem of submodular function minimization is usually treated in an oracle context, that is, we assume that function g is available only through a "black box" subroutine which computes $g(A)$ for any $A \subseteq E$. The computational effort of a minimization algorithm is measured by the number of calls on the oracle, plus the amount of other computation performed; that is, an oracle-call is regarded as an "elementary step".

For the situation in which g is integer-valued and known to satisfy $|g(A)| \leq M$ for some positive integer M and all $A \subseteq E$, Grötschel, Lovász, and Schrijver [8]

Research partially supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

AMS subject classification (1980): 05 B 99, 68 C 25

have given a minimization algorithm for which the running time is bounded by a polynomial in $n=|E|$ and $\log M$. This is therefore a polynomial-time algorithm in the usual sense. However, it is based on the ellipsoid method, and seems to be of little practical value. We mention two special cases in which more satisfactory minimization algorithms are known. If the submodular function g is known to be a cut function, there is an easy algorithm [6] to construct a flow network realizing g, using an oracle for g. Combining this procedure with well-known methods for finding a minimum-capacity cut in a network, we have an efficient combinatorial algorithm for cut-function minimization. If the submodular function g is known to be of the form $g(A)=r(A)-x(A)$, where r is the rank function of a matroid and $x(A)$ denotes $\sum(x_j: j\in A)$ for a vector $(x_j: j\in E)$ (we use this notation throughout), then there is an efficient combinatorial minimization algorithm [5]. (Here we mention that r and x need not be known.) These two minimization algorithms share the desirable property that the number of steps counting oracle calls and arithmetic operations as single steps, *is bounded by a polynomial in n, independent of the size, or even the rationality, of g,* furthermore, the number of digits of any number occurring during this computation is bounded by a polynomial of the number of digits in the input number, that is, they are "strongly polynomial". In addition, the only arithmetic operations that they need are additions, subtractions, and comparisons. Of course, the ellipsoid method does not enjoy these properties.

However, the two special cases just mentioned seem to be the only ones so far solved in a satisfactory manner. It is an outstanding open problem to find a practical combinatorial algorithm to minimize a general submodular function, which also runs in polynomial time. In [1] a combinatorial approach to the problem, which generalizes the one used in the matroid case [5], was introduced, but the resulting algorithm was proved only to be finite. In this paper we describe an algorithm based on the same ideas, which runs in "pseudo-polynomial time". That is, where g is integer-valued and satisfies $|g(A)|\leqq M$ for all $A\subseteq E$, its running time is bounded by a polynomial in n and M. For several applications, for example, the matroid intersection problem, M is itself polynomially-bounded, and hence also is this algorithm.

2. Preliminaries

We review the approach described in [1, Section 4] and the underlying results of Edmonds [7]. That approach is based on the minimization of functions of the form $f(A)+x(E\setminus A)$, where $x\in\mathbf{R}^E$ and f is a *polymatroid function*, that is, f is submodular, increasing $(f(A)\leqq f(B)$, if $A\supseteq B)$, and normalized $(f(\emptyset)=0)$. Any submodular function minimization problem can be reduced to one of this form, based on the following easy result from [4].

Lemma 2.1. *Let* $x_j=g(E\setminus\{j\})-g(E)$, *for each* $j\in E$. *Define* f *by* $f(A)=g(A)+x(A)-g(\emptyset)$. *Then* f *is a polymatroid function.* ∎

For any $A\subseteq E$,

$$\begin{aligned} f(A)+x(E\setminus A) &= f(A)-x(A)+x(E) \\ &= g(A)-g(\emptyset)+x(E), \end{aligned}$$

so $f(A)+x(E\setminus A)$ is minimized precisely when g is minimized, as required. In addition, we can assume that $x\geqq 0$, by the following argument. By submodularity, $g(E\setminus\{j\})-g(E)\geqq g(A\setminus\{j\})-g(A)$ for any $A\subseteq E$ with $j\in A$. Therefore, if $x_j<0$, then j can be an element of no minimizer of g. Hence, where $B=\{j: x_j<0\}$ we can restrict attention to the function g' defined on subsets of $E\setminus B$ by $g'(A)=g(A)$.

The motivation for minimizing $f(A)+x(E\setminus A)$, where f is a polymatroid function and x is non-negative, comes from the work of Edmonds [7]. Consider the polyhedron $P(f)=\{y\in\mathbf{R}^E: y\geqq 0,\ y(A)\leqq f(A)$ for all $A\subseteq E\}$; $P(f)$ is a "polymatroid". The next result provides a characterization of minimizers of $f(A)+x(E\setminus A)$. Notice that, if g is integer-valued, then so too are x and f.

Theorem 2.2. (Edmonds) $\max(y(E): y\leqq x, y\in P(f))=\min(f(A)+x(E\setminus A): A\subseteq E)$. *Moreover, if f and x are integer-valued, then there exists a maximizing y which is integer-valued.* ∎

Notice that $\max\leqq\min$ in Theorem 2.2 is obvious. That Theorem 2.2 does provide a good characterization of the minimum is *not* completely obvious. It follows from the facts that $y\in P(f)$ can be expressed as a convex combination $\sum(\lambda_i v^i: i\in I)$ of at most $n+1$ extreme points v^i of $P(f)$, and that Edmonds' "greedy algorithm" [7] gives a good characterization of extreme points.

Given $y\in P(f)$, a set $A\subseteq E$ is *y-tight* if $y(A)=f(A)$. The union and intersection of tight sets are easily shown to be tight also, so there is a unique maximal tight set, which we denote by $\mathrm{cl}(y)$, the closure of y. The main result of [1] is an efficient algorithm which, given an extreme point v^i of $P(f)$, constructs a poset Q_i on $\mathrm{cl}(v^i)$ whose lower ideals (sets $B\subseteq\mathrm{cl}(v^i)$ such that $a\geqq b\in B$ implies $a\in B$) are precisely the v^i-tight sets. Moreover, for each pair a, b such that a covers b in Q_i, there is a number $\varepsilon=\varepsilon_i(a,b)>0$, also easily computed, such that $v^i+\varepsilon\{a\}-\varepsilon\{b\}$ is also an extreme point of $P(f)$. (For $A\subseteq E$, we also use A to denote the incidence vector of A, so $v^i+\varepsilon\{a\}-\varepsilon\{b\}$ is obtained from v^i by raising component a by ε and lowering component b by ε.) Finally, for each $a\notin\mathrm{cl}(v^i)$ there is a number $\varepsilon=\varepsilon_i(a,s)$ (this notation turns out to be convenient), also easily computed, such that $v^i+\varepsilon\{a\}$ is an extreme point of $P(f)$.

The above result, combined with the characterization of $\min(f(A)+x(E\setminus A))$ in Theorem 2.2, motivates the following augmenting path approach. We have extreme points v^i, $i\in I$, of $P(f)$ and positive numbers λ_i, $i\in I$, adding to one, such that $y=\sum(\lambda_i v^i: i\in I)\leqq x$. (We may begin for example, with $I=\{0\}$, $v^0=0$, and $\lambda_0=1$, so $y=0$.) We wish either to find A such that $y(E)=f(A)+x(E\setminus A)$, so A is a minimizer, or "augment" y to a new y (and I, λ) providing a better lower bound on the minimum. We construct a digraph $G=G(I,\lambda)$ as follows. The vertex-set is $E\cup\{r,s\}$, where $r, s\notin E$. There is an edge (r,a) for every $a\in E$ such that $y_a<x_a$. There is an edge (a,b) for every pair $a, b\in E$ such that, for some $i\in I$, a covers b in Q_i. There is an edge (a,s) for every $a\in E$ such that for some $i\in I$, $a\notin\mathrm{cl}(v^i)$.

Suppose that there exists no directed path from r to s in G. Then there is a set A such that no edge leaves $A\cup\{r\}$. It follows that $y(E\setminus A)=x(E\setminus A)$, since $y_j=x_j$ for each $j\in E\setminus A$. Moreover, A is v^i-tight for each $i\in I$, so $y(A)=\sum(\lambda_i v^i(A): i\in I)=\sum(\lambda_i f(A): i\in I)=f(A)$. Hence $y(E)=f(A)+x(E\setminus A)$, and we are finished.

Suppose, on the other hand, that there is a directed path in G having vertex-sequence $r=a_0, a_1, a_2, \ldots, a_{m}, a_{m+1}=s$. We can construct an expression for $y'=$

$=y+\delta\{a_1\}$ (for some $\delta>0$) as a convex combination of at most $n+m+1$ extreme points, as follows: For each j, $1\leqq j\leqq m$, choose $i(j)\in I$ such that a_j covers a_{j+1} in $Q_{i(j)}$, or $a_j\notin \mathrm{cl}(v^{i(j)})$ if $j=m$, and let ε_j denote the associated $\varepsilon_{i(j)}(a_j, a_{j+1})$. Replacing $\lambda_{i(j)}$ by $\lambda_{i(j)}-\alpha_j$ and increasing the coefficient of $v^{i(j)}+\varepsilon_j\{a_j\}-\varepsilon_j\{a_{j+1}\}$, or $v^{i(m)}+\varepsilon_m\{a_m\}$, by $\alpha_j>0$, has the effect of raising component a_j of y by $\alpha_j\varepsilon_j$ and (if $j<m$) of lowering component a_{j+1} by $\alpha_{j+1}\varepsilon_{j+1}$. If we choose each α_j so that $\alpha_j\varepsilon_j=\delta$ for all j, and make this change for each j, we obtain an expression for $y'=$ $=y+\delta\{a_1\}$ as a convex combination of at most $n+m+1$ extreme points of $P(f)$, provided that the α_j are sufficiently small. We must choose δ so that

$$\delta \leqq x(a_1)-y(a_1);$$

and so that

$$\alpha_j|\{j'\colon i(j')=i(j),\quad 1\leqq j'\leqq m\}|\leqq \lambda_{i(j)},\quad 1\leqq j\leqq m.$$

If we let $k(j)$ denote the cardinality of the above set, then the largest possible value for ε is

$$\min\big(x(a_1)-y(a_1),\ \min(\lambda_{i(j)}\varepsilon_j/k(j)\colon 1\leqq j\leqq m)\big).$$

After performing such an augmentation we can use a standard method from linear programming to obtain an expression for y' as a convex combination of at most $n+1$ extreme points. (This technique is not the simplex method; it is just the method for turning a feasible solution into a basic feasible solution.) It requires $O(n^3)$ time, as does the method for constructing G, so finding and performing a single augmentation can be done in time $O(n^3)$. In [1] arguments which involve refining the augmentation step so that the set I changes at each step, and can never be repeated, lead to a finite algorithm. Those arguments do not depend on any discreteness assumptions about f or x. In this paper we attempt to choose an augmentation whose *amount* δ is relatively large; for this approach to give useful bounds, we do need assumptions about f and x.

3. A refined algorithm

As pointed out before, if g is integer-valued, then so too are f and x. By a well-known result [7], each extreme point v^i of $P(f)$ is integer-valued in this case, and hence also are the numbers $\varepsilon_i(a, b)$. Now suppose that g satisfies $|g(A)|\leqq M$ for $A\subseteq E$. Then f satisfies

$$f(A)=g(A)-g(\emptyset)+\sum(g(E\setminus\{j\})-g(E)\colon j\in A)$$

$$\leqq 2M(|A|+1),\quad \text{for all}\quad A\subseteq E.$$

Moreover, x satisfies $x_j=g(E\setminus\{j\})-g(E)\leqq 2M$, for all $j\in E$. In the sequel we assume, for convenience, that M has been redefined so that the factor of 2 disappears. That is, we assume that $f(A)\leqq M(|A|+1)$ for all $A\subseteq E$ and that $x_j\leqq M$ for all $j\in E$. The method for obtaining an improved bound is based on the following result.

Theorem 3.1. *Suppose that f is integer-valued and satisfies $f(A)\leqq M(|A|+1)$ for all $A\subseteq E$. Let N be a positive number. Given y, I, λ at any stage of the algorithm, there is an efficient algorithm which finds either*

(a) *An* (r, s) *dipath in* $G(I, \lambda)$ *giving an augmentation of amount at least* $N/(Mn(n+1)^2)$;

or

(b) *A set* $A \subsetneq E$ *with* $y(E)+N > f(A)+x(E \setminus A)$.

Proof. We claim that if $A \subsetneq E$ satisfies $y(E)+N \leqq f(A)+x(E \setminus A)$, then there exists either an edge (r, a) of G with $a \notin A$ and $x_a - y_a \geqq N/(Mn(n+1)^2)$, or an edge (a, b) with $a \in A$ and $b \in (E \setminus A) \cup \{s\}$ and $i \in I$ with a covering b in Q_i and $\lambda_i \varepsilon_i(a, b)/n \geqq N/(Mn(n+1)^2)$. Then the algorithm which searches for an (r, s) dipath in the subgraph of G having only edges of these two types must terminate with either (a) or (b). (We are using the fact that any augmenting path has $m \leqq n$ and hence $\min(\lambda_{i(j)}\varepsilon_j/k(j)\colon 1 \leqq j \leqq m)$ is at least $\min(\lambda_{i(j)}\varepsilon_j/n\colon 1 \leqq j \leqq m)$.) Now we prove the claim. Let $p = f(A) - y(A)$.

First, consider the case when $p < N(|A|+1)/(n+1)$. Then $A = E$ would imply $p = f(E) - y(E) \geqq N$, a contradiction, so $n - |A| \geqq 1$. Now $x(E \setminus A) - y(E \setminus A) \geqq N - p$, so there exists $j \in E \setminus A$ with

$$x_j - y_j \geqq (N-p)/(n-|A|) > \big(N - N(|A|+1)/(n+1)\big)/(n-|A|)$$
$$= N/(n+1) \geqq N/\big(n(n+1)^2 M\big),$$

as required.

Next consider the case when $p \geqq N(|A|+1)/(n+1)$. In this case, it is clear that $p > 0$. Then $p = f(A) - y(A) = \sum(\lambda_i(f(A) - v^i(A))\colon i \in I)$. Hence there exists $i \in I$ with $\lambda_i(f(A) - v^i(A)) \geqq p/(n+1)$. For this i, $f(A) > v^i(A)$. Since $f(A) - v^i(A) \leqq f(A) \leqq M(|A|+1)$,

$$\lambda_i \geqq p/\big((n+1)M(|A|+1)\big),$$
$$\geqq N/(n+1)^2 M.$$

Now since $f(A) > v^i(A)$, there exist $a \in A$, $b \in (E \setminus A) \cup \{s\}$ such that a covers b in Q_i. Then $\varepsilon_i(a, b) \geqq 1$, so $\lambda_i \varepsilon_i(a, b)/n \geqq N/\big(n(n+1)^2 M\big)$, as required. ∎

Let us put $N = 1$ in Theorem 3.1. The first time that the algorithm fails to find an augmentation of amount at least $1/(Mn(n+1)^2)$, it finds a set $A \subsetneq E$ such that $f(A)+x(E \setminus A) < y(E)+1$. But then it follows from Theorem 2.2 that this A minimizes $f(A)+x(E \setminus A)$, and we are finished. On the other hand, since $\max\big(y(E)\colon y \leqq x,\ y \in P(f)\big)$ cannot exceed $x(E) \leqq Mn$, there can be at most $M^2 n^2 (n+1)^2$ augmentations of amount at least $1/(Mn(n+1)^2)$. Hence, the algorithm terminates after at most $M^2 n^2 (n+1)^2$ augmentations.

We can improve this bound on the number of augmentations by using Theorem 3.1 in a slightly more sophisticated way. Let q denote $n(n+1)^2 M$ and let K denote $\lfloor \log nM \rfloor$. (Unless otherwise specified, the logarithm base is 2.) We refine the algorithm further, as follows. For $k = K, K-1, \ldots, 0$, look for and perform augmentations of amount at least $2^k/q$, until the above method first fails to find one. At the beginning of stage k, the current y satisfies $y(E)+2^{k+1} > \min\big(f(A)+x(E \setminus A)\colon A \subsetneq E\big)$. This is true for $k = K$ by the definition of K, and for other values of k follows from putting $N = 2^{k+1}/q$ in Theorem 3.1. Therefore, the number of augmentations during stage k is at most $2^{k+1}/(2^k/q) = 2q$. It follows that the total number of augmentations is at most $2q(K+1)$. (Actually, there is a lot of slack in these

bounds. It is easy to see that, if the number of augmentations in stage k *is* $2q$, then there can be no more augmentations at all in later stages. In fact, one can show that there can be at most $q \log Mn$ augmentations in total.) Thus we have a bound of $O(Mn^3 \log Mn)$ for the number of augmentations, and a time bound of $O(Mn^6 \log Mn)$ for the algorithm.

There is a simpler version of the algorithm for which the same time bound holds. Suppose that, at each step, we find an augmenting path which maximizes $\min(x(a_1)-y(a_1), \min(\lambda_{i(j)}\varepsilon_j/n: 1 \leqq j \leqq m))$, and stop when this value becomes less than $1/(Mn(n+1)^2)$. Then every augmentation in this "greedy" method will be valid in the above "staged" method, so the greedy method has the same bound on the number of augmentations. It is slightly more work to find an augmenting path in the greedy method than in the other method, but this work is dominated by the other work (auxiliary digraph construction, linear algebra) involved in an augmentation, so the overall bound is valid.

The greedy method has the advantages of being simpler to state and of making sense (except for the termination criterion) even in case g is not assumed to be integer-valued. (Lest the reader be misled by the name, the greedy method does *not* attempt to find an augmentation of largest possible amount; that would seem to be very difficult to do.)

We remark that there is a slightly more direct proof of the bound for the greedy method. By Theorem 3.1 the gap between the current $y(E)$ and $\min(f(A)+x(E \setminus A): A \subseteq E)$ changes by a factor of at most $1-1/q$, where $q=Mn(n+1)^2$. Since the natural logarithm function satisfies $\ln(1+z)<z$ for $-1<z<0$, we have $q \ln(1-1/q)<-1=\ln(1/e)$. Therefore, $(1-1/q)^{q \ln(Mn)}<(1/e)^{\ln(Mn)}=1/Mn$. Since the initial gap is at most Mn, it follows that the gap is less than 1 after at most $q \ln(Mn)$ augmentations. After at most q more augmentations the algorithm can be terminated, giving the same $O(Mn(n+1)^2 \log(Mn))$ bound for the number of augmentations.

4. Remarks

4.1. Improving the bound. The bound we have obtained is based on the fact that each augmentation changes the gap between the current and final values of $y(E)$ by a factor of at most $(q-1)/q$. The fact that q depends linearly on M leads to the pseudo-polynomial bound. If, for example, we could always find an augmentation for which the corresponding ratio were determined by a polynomial in n only, we would obtain a polynomial bound. Such a result would seem to depend on the existence of $i \in I$ and an edge (a, b) of G leaving $A \subseteq E$ with $\varepsilon_i(a, b)$ large, whenever $f(A)-y(A)$ is large. (In the proof of Theorem 3.1 we used only the fact that $\varepsilon_i(a, b) \geqq 1$, by integrality.) In particular, one might expect such an edge to exist whenever $f(A)-v^i(A)$ is large. This need not be true, as the following example shows. Let $E=\{1, 2, 3\}$, and define f by $f(\emptyset)=0$, $f(\{1\})=N+1$, $f(\{2\})=3N$, $f(\{1, 2\})=$ $=f(\{3\})=4N$, $f(\{1, 3\})=5N$, $f(\{2, 3\})=f(\{1, 2, 3\})=6N$. Here N is a large integer. It is easy to check that f is a polymatroid function, and that $v=(N+1, 3N-1, 2N)$ is a vertex of the associated polymatroid. The poset determined by v is linearly ordered, with 3 covering 2, and 2 covering 1. Hence the only edge of G leaving $A=\{2, 3\}$ is (2,1), and $\varepsilon(2, 1)=f(\{2\})-v_2=1$. But $f(A)-v(A)=6N-(5N-1)=$

$=N+1$. This example suggests that the arguments used here to obtain a pseudo-polynomial algorithm may not be enough to obtain a polynomial algorithm.

There may be more hope to refine the algorithm of [1] in a different direction, as in the matroid case [5]. In that paper a method was presented which combined at one step many augmentations of the type used here, and then performed these "grand augmentations" in a special order. For the resulting scheme, the auxiliary digraph can change in a rather limited way, and this leads to a purely combinatorial convergence argument (that is, one which does not depend on the size of augmentations).

4.2. Computing a maximizing y. The algorithm we have described is able to determine a minimizing set $A \subseteq E$ in the min—max formula of Theorem 2.2 by recognizing that the current y is close enough to being maximum. The question arises whether the algorithm can be used to *find* a maximizing y, under the same integrality assumptions on f and x. In polymatroid terminology, this is the problem of computing a basis in an integral polymatroid. There is a simple property of integral polymatroids which allows the present algorithm to be used. (In fact, this property is part of the definition [7].) Namely, every maximal integral vector y satisfying $y \leqq x$ and $y \in P(f)$, has maximum component-sum. We can obtain an integral $y \in P(f)$, such that $y(E) \geqq \min\{f(A)+x(E \setminus A): A \subseteq E\}-n$, by simply rounding down to the next integer each component of the y with which the algorithm terminates. For $e \in E$ such that $y_e < x_e$, we use the algorithm again to test whether increasing y_e by 1 yields a vector in $P(f)$. In this way, after at most $2n$ applications of the algorithm, we obtain a maximal integral vector y satisfying $y \leqq x$ and $y \in P(f)$. This is the desired maximizing y in Theorem 2.2, and it has been computed in pseudo-polynomial time.

4.3. Computing (I, λ). It is also of interest to compute an expression $\sum(\lambda_i v^i: i \in I)$ for the maximizing y as a convex combination of vertices of $P(f)$, or to compute such an expression for any element of $P(f)$. The ellipsoid method [8] can also do this in polynomial time. The finite algorithm of [1] computes (I, λ), but the present version of it clearly does not. The problem of computing (I, λ) by an efficient combinatorial algorithm is interesting, and is not easy even for some cases where there do exist easy methods for computing y and A in Theorem 2.2. For example, let $\{E, F\}$ be a bipartition of a graph G, let $x_j = 1$ for $j \in E$, and let $f(A)$ denote $|\{f \in F: f \text{ is adjacent to an element of } A\}|$. Then f is a polymatroid function, and y and A of Theorem 2.2 are easily computed by a bipartite matching algorithm. But I see no easy way to obtain an expression for y as a convex combination of vertices of $P(f)$.

4.4. Some applications. The algorithm described here provides a polynomial-time combinatorial algorithm to minimize an integer-valued submodular function g satisfying, for example, $g(A) \geqq kn$ for all $A \subseteq E$ and some constant k. We mention several examples. For each of them there already exists a more efficient combinatorial algorithm, but that algorithm uses more information than just the values of g. For example, where r_1 and r_2 are rank functions of matroids on E, let $g(A) = r_1(A) + r_2(E \setminus A)$. The minimum of this function is well known to be the maximum cardinality of a common independent set of the two matroids. Similarly, one can use the algorithm to compute the minimum of $|E \setminus A| + \sum r_i(A)$, which is the maximum

cardinality of a set partitionable into sets independent in each of k matroids, having rank functions $r_1, \ldots, r_k$.

Finally, we describe an application due to Bouchet [2]. Let G be a strongly connected digraph having vertex-set E, and let M be its adjacency matrix. Given $A, B \subseteq E$, let $M(A, B)$ be the submatrix of M whose rows are indexed by A and whose columns are indexed by B. Define $g(A)$ to be the rank of $M(A, E \setminus A)$ plus the rank of $M(E \setminus A, A)$. For all A, $\emptyset \subset A \subset E$, $g(A) \geq 2$. It is not difficult to see that there exists A with $g(A)=2$ and $|A| \geq 2 \leq |E \setminus A|$ if and only if G is decomposable, as in [3]. It is possible [4] to find such an A, or determine that none exists, by minimizing at most n^2 submodular functions easily obtained from g, and these functions have values bounded by n.

References

[1] R. E. BIXBY, W. H. CUNNINGHAM and D. M. TOPKIS, The poset of a polymatroid extreme point, *Math. of Operations Res., to appear.*
[2] A. BOUCHET, *private communication,* 1984.
[3] W. H. CUNNINGHAM, Decomposition of directed graphs, *SIAM J. Alg. and Disc. Methods* **3** (1982), 214—228.
[4] W. H. CUNNINGHAM, Decomposition of submodular functions, *Combinatorica* **3** (1983), 53—68.
[5] W. H. CUNNINGHAM, Testing membership in matroid polyhedra, *Journal of Combinatorial Theory (B)* **36** (1984), 161—188.
[6] W. H. CUNNINGHAM, Minimum cuts, modular functions, and matroid polyhedra, *Networks, to appear.*
[7] J. EDMONDS, Submodular functions, matroids, and certain polyhedra, in: *Combinatorial Structures and their Applications* (ed.: R. K. Guy et al.), Gordon and Breach, New York, 1970.
[8] M. GRÖTSCHEL, L. LOVÁSZ and A. SCHRIJVER, The ellipsoid method and its consequences in combinatorial optimization, *Combinatorica* **1** (1981), 169—197.
[9] L. LOVÁSZ, Submodular functions and convexity, in: *Mathematical Programming: The State of the Art* (ed: A. Bachem et al.), Springer-Verlag (1983), 235—257.

William H. Cunningham

Department of Mathematics and Statistics
Carleton University
Ottawa, K1S 5B6 Canada