MATHEMATICS OF OPERATIONS RESEARCH
Vol. 10, No. 2, May 1985
Printed in U.S.A.

A PRIMAL-DUAL ALGORITHM FOR SUBMODULAR FLOWS*†

WILLIAM H. CUNNINGHAM‡ AND ANDRÁS FRANK§

Previously the only polynomial-time solution algorithm to solve the optimal submodular flow problem introduced by Edmonds and Giles was based on the ellipsoid method. Here, modulo an efficient oracle for minimizing certain submodular functions, a polynomial time procedure is presented which uses only combinatorial steps (like building auxiliary digraphs, finding augmenting paths). The minimizing oracle is currently available only via the ellipsoid method, in general; however in important special cases, such as network flows, matroid intersections, orientations, and directed cut coverings, the necessary oracle can be provided combinatorially.

1. Introduction. In [4] Edmonds and Giles proved a very general min-max relation concerning submodular functions on directed graphs. This "submodular flow" problem includes among others, the orientation [7], minimum cost flow [6], polymatroid intersection [3], and directed cut covering [17] problems. On the algorithmic side Grötschel, Lovász, and Schrijver [13] have discovered a good algorithm for the submodular flow problem based on the ellipsoid method. However the ellipsoid method seems to have only theoretical significance, so it is desirable to have an efficient solution algorithm which is purely combinatorial. Such combinatorial algorithms have proved fundamental from the practical point of view and theoretically as well in describing the structure of the problem in question.

In [7] a constructive method was presented for proving the Edmonds–Giles theorem. That procedure provides a polynomially-bounded algorithm when the variables are bounded by 0 and 1, regardless of the integrality of the cost function. Although the algorithm could be extended to the more general case when the constraints include general integer bounds on the variables, then its complexity would include a factor proportional to the maximum absolute value M of the bounds. Therefore, the resulting algorithm would not be a good algorithm, since the complexity of a proper polynomial algorithm is allowed only to involve a polynomial in $\log M$, the number of digits of M.

The main purpose of the present work is to develop a polynomial-time combinatorial procedure to solve the optimal submodular flow problem (that is, finding a minimum cost submodular flow) and its linear programming dual. In [8] an algorithm for finding a feasible solution to a submodular flow problem was described. A variant, called the maximal submodular flow algorithm, was also discussed there for maximizing $x(j)$ for a fixed arrow j over submodular flows x. This algorithm is polynomially bounded regardless of the variable-bounds and the values of the submodular function. The key to that algorithm is that the successive augmenting paths are chosen in a lexicographic order; this idea is due to Schönsleben [19] and Lawler and Martel [15]. Our present algorithm is obtained by combining the ideas of [7] and [8] with the

*Received September 24, 1982; revised November 18, 1983.
AMS 1980 subject classification. Primary: 90C35.
OR/MS Index 1978 subject classification. Primary: 484 Networks/graphs/flow algorithms.
Key words. Submodular flows, directed graphs, oracle methods, ellipsoid method.
†Supported by Sonderforschungsbereich 21 (DGF), Institut für Operations Research, Universität Bonn. Partially supported by a grant from NSERC of CANADA.
‡Carleton University.
§Research Institute for Telecommunication.

0364-765X/85/1002/0251$01.25

scaling technique. (The proof of the validity and efficiency of the algorithm relies heavily on [8].)

It should be emphasized that the algorithms in [7] and [8], as well as the present one and its specialization to polymatroid intersection, require an oracle which can, roughly, minimize a submodular function. Currently the only polynomial-time algorithm for this problem [13] is also based on the ellipsoid method. (However, in many of the important applications the relevant oracle is available via a combinatorial algorithm.) It is an extremely important open problem to find a good combinatorial algorithm for the general submodular function minimization problem.

Modulo this minimizing oracle our procedure provides an efficient solution algorithm for the weighted polymatroid intersection problem, that of finding a maximum-weight vector in the intersection of two polymatroids. Previously, efficient combinatorial algorithms were known only for the case of (0, 1) weights [15], [19]. Similarly, our method provides the first efficient combinatorial algorithm for the minimum cost versions of a number of other models, for example, independent flows [11], flows with set constraints [14], polymatroidal network flows [15], and generalized polymatroids [10], all of which have recently been shown to be equivalent to submodular flows. See Schrijver [20] for an excellent survey. Along the way we derive a purely combinatorial good characterization of dual feasibility for an optimal submodular flow problem.

The strategy of our algorithm, roughly, is as follows. First we formulate the optimality criteria for the arrows with the help of the so-called potentials. The algorithm starts with a feasible solution and a potential which may violate the optimality criteria. It then improves the violated optimality criteria arrow by arrow. This is done by applying the maximal submodular flow algorithm [8] to a face of the original submodular flow polyhedron. We shall prove that it is also a submodular flow polyhedron, so the algorithm of [8] works.

This procedure provides a finite algorithm. However, by applying a kind of scaling technique, it is made to be an algorithm whose complexity involves a factor proportional to K, the logarithm of the maximum absolute value of the (integral) objective function, and does not otherwise depend on the numbers involved. In fact the algorithm requires the order of Kn^5 flow augmentations where n is the number of nodes.

Throughout the paper we work with a finite ground set V of n elements. If $A \subseteq V$, then $\bar{A}$ denotes $V \setminus A$. Sets A, B are *co-disjoint* if $\bar{A} \cap \bar{B} = \emptyset$. Sets A, B are *intersecting* if $A \cap B$ is nonempty. If, in addition, $A \cup B \neq V$, then A, B are *crossing*. A set A is a *$u\bar{v}$-set* if $u \in A$, $v \notin A$. A set function is *submodular* on A, B if $b(A) + b(B) \geqslant b(A \cap B) + b(A \cup B)$. If equality holds, b is *modular* on A, B. A set function $b: 2^V \to R \cup \{+\infty\}$ is *fully submodular* if b is submodular on every pair of subsets; b is an *intersecting* (*crossing*) *submodular* function if b is submodular on every intersecting (crossing) pair of subsets. Throughout we suppose that $b(\emptyset) = b(V) = 0$ for each set function b. Sometimes we construct new submodular functions b. Then $b(X)$ is meant to be $+\infty$ if no other value is assigned to X explicitly.

Let $G = (V, E)$ be a directed graph with node-set V and arrow-set E. Multiple arrows are allowed but loops are not. An arrow *uv leaves* (*enters*) $B \subseteq V$ if B is a $u\bar{v}$-set ($v\bar{u}$-set); the set of such arrows is denoted by $\delta(B)$ ($\rho(B)$). For a vector $x \in R^E$, and $B \subseteq V$, $\lambda_x(B)$ denotes $\sum(x(e): e \text{ enters } B) - \sum(x(e): e \text{ leaves } B)$. It is easy to check that λ_x is modular on pairs A, B of subsets of V. For a vector $z \in R^V$ and for $B \subseteq V$, $z(B)$ denotes $\sum(z(v): v \in B)$. Obviously z is modular and every modular set function with $z(\emptyset) = 0$ comes in this way. We do not distinguish between a modular function z with $z(\emptyset) = 0$ and a vector $z \in R^V$.

2. Submodular flows. Let b'' be a crossing submodular function. Let $d \in R^E$, that is, d is a real weighting of the arrows. Similarly, let $f \in (R \cup \{-\infty\})^E$, $g \in (R \cup$

$\{\infty\})^E$. Consider the following linear program:

$$\max dx \quad \text{subject to} \quad \lambda_x(F) \leqslant b''(F) \quad \text{for} \quad F \subseteq V; \quad f \leqslant x \leqslant g. \tag{1}$$

A feasible solution x of (1) is called a *submodular flow* and the set of feasible solutions is a *submodular flow polyhedron.*

THEOREM 1 (Edmonds–Giles). *If d is integer-valued and the linear program dual to* (1) *has an optimal solution, then it has an integer-valued optimal solution. If b'' and (the finite components of) f, g are integer-valued and* (1) *has an optimal solution, then it has an integer-valued optimal solution.*

(Edmonds and Giles formulated their theorem for submodular functions b'' defined on a crossing family of subsets. By extending b'' with value ∞ to every subset we get an equivalent formulation. But it is more convenient to work with functions defined on every subset.)

Our algorithm will find these optima efficiently in the case when d is integer-valued. A rational d can, of course, be replaced by Dd, where D is a common denominator for the components of d. It should be noted that the algorithm of [7] for the $(0, 1)$ case runs in polynomial time without any assumption on d. If there are no assumptions at all about the data, we will indicate how our techniques, without scaling, lead to a *finite* algorithm.

In [8] an algorithm was described for finding a feasible solution to (1). That paper also presents a variant, the maximal submodular flow algorithm, which solves (1) when $d(e) = 1$ for a specified arrow e and $d(e) = 0$ otherwise. Here we "apply" both algorithms but in different senses. The present algorithm starts with any feasible solution and this is obtained (for example) by applying the feasibility algorithm of [8]. The maximal submodular flow algorithm of [8] will play another kind of role in our discussion. Namely we shall use it for proving the validity of the method.

We also remark that the feasibility algorithm needs an oracle to minimize $b''(F) - z(F)$ over $u\bar{v}$-sets where z is an arbitrary modular function, while the submodular flow algorithm (once a starting feasible solution is available) needs this oracle only for modular functions of the form $z(F) = \lambda_x(F)$.

In applications (such as orientation or directed cut coverings) it is very important that b'' is required to be submodular on crossing sets only. On the other hand:

THEOREM 2. *A submodular flow polyhedron P can be defined by a fully submodular function.*

This is a result of Fujishige [12], and also follows from results in [7]. The proof depends on the following two lemmas whose proofs can be found in [7]. The first one originated in Dunstan [2] and Lovász [16].

LEMMA 3. *Where b' is an intersecting submodular function, b defined by $b(X) = \min(\sum b'(X_i) : \{X_i\}$ a partition of $X)$ is fully submodular. Moreover, $\{z \in R^V : z(F) \leqslant b'(F), F \subseteq V\} = \{z \in R^V : z(F) \leqslant b(F), F \subseteq V\}$.*

LEMMA 4. *Where b'' is a crossing submodular function, $b'(X) = \min(\sum b''(X_i) : \{\bar{X}_i\}$ a partition of $\bar{X})$ is an intersecting submodular function. Moreover, $\{z \in R^V : z(V) = 0, z(F) \leqslant b''(F), F \subseteq V\} = \{z \in R^V : z(V) = 0, z(F) \leqslant b'(F), F \subseteq V\}$.*

Applying first Lemma 4 to $z = \lambda_x$, one can see that a submodular flow polyhedron P can be determined by an intersecting submodular function. Now from Lemma 3 the theorem follows.

In [7] and [8] the first reduction (Lemma 4) was used to get a submodular flow problem in intersecting form. There we described the algorithm for intersecting

submodular functions and showed how to use it in the crossing case. The second reduction did not seem to have any advantage. In the present paper however we use both reductions because the discussion here becomes simpler. We describe the algorithm assuming fully submodular functions. It will turn out that this algorithm to get an optimal primal solution from any feasible solution can be used without any change for submodular flows given by a crossing submodular function. One difference between the case of fully submodular functions and crossing submodular functions occurs in finding a feasible solution. This is treated in [8]. The other difference is in getting the optimal dual solution. This is discussed in [7] and briefly outlined in §6.

Let us exhibit another important relation among the functions b'', b' and b in Lemmas 3 and 4. For $x \in P$ and $u, v \in V$, let $\epsilon_x(u,v) = \min(b(F) - \lambda_x(F) : F$ a $u\bar{v}$-set). $\epsilon'_x(u,v)$ and $\epsilon''_x(u,v)$ are defined analogously.

THEOREM 5. $\epsilon_x(u,v) = \epsilon'_x(u,v) = \epsilon''_x(u,v)$.

PROOF. The second equality was shown in [8]. The first one is shown similarly: obviously $\epsilon_x(u,v) \leqslant \epsilon'_x(u,v)$. Choose a $u\bar{v}$-set X such that $\epsilon_x(u,v) = b(X) - \lambda_x(X) = \sum(b'(X_i) - \sum\lambda_x(X_i))$ for some partition $\{X_i\}$ of X. One X_i is a $u\bar{v}$-set so $\sum(b'(X_i) - \lambda_x(X_i)) \geqslant \epsilon'_x(u,v)$. ∎

The next result will be important for proving the validity of the algorithm.

THEOREM 6. *Any face Q of a submodular flow polyhedron P is a submodular flow polyhedron.*

PROOF. Suppose that P is defined by a fully submodular function b and $Q \neq \emptyset$. It is clear that requiring any of the inequalities $f(e) \leqslant x(e)$ or $x(e) \leqslant g(e)$ to hold with equality causes no difficulty. So we suppose that Q is defined by fixing $\lambda_x(B)$ on the members of a family $\mathscr{B}$ of subsets, that is, $Q = \{x : x \in P, \lambda_x(B) = b(B)$ for $b \in \mathscr{B}\}$. Suppose that $\sum(|B|^2 : B \in \mathscr{B})$ is maximum.

Claim. For $B_1, B_2 \in \mathscr{B}$, either $B_1 \subset B_2$ or $B_2 \subset B_1$.

PROOF. Observe that $b(B_1) + b(B_2) = b(B_1 \cap B_2) \cup b(B_1 \cup B_2)$. Indeed, for $x \in Q$, $x(B_1) + x(B_2) = b(B_1) + b(B_2) \geqslant b(B_1 \cap B_2) + b(B_1 \cup B_2) \geqslant x(B_1 \cap B_2) + x(B_1 \cup B_2) = x(B_1) + x(B_2)$ whence equality follows everywhere. Since $\{x \in P : x(B_i) = b(B_i),\ i = 1,2\} = \{x \in P : x(B_1 \cap B_2) = b(B_1 \cap B_2),\ x(B_1 \cup B_2) = b(B_1 \cup B_2)\}$, $(\mathscr{B} \setminus \{B_1, B_2\}) \cup \{B_1 \cap B_2, B_1 \cup B_2\}$ defines Q as well. By the choice of $\mathscr{B}$ the claim follows.

Let $\mathscr{B} = \{B_1, B_2, \ldots, B_k\}$ where $\emptyset = B_1 \subset B_2 \subset \cdots \subset B_k = V$. Define $\hat{b}(X)$ as follows. $\hat{b}(X) = b(X \cup B_i) - b(B_i)$ if $X \subseteq B_{i+1} \setminus B_i$. Set $\hat{Q} = \{x \in R^E : f \leqslant x \leqslant g$, $\lambda_x(F) \leqslant \hat{b}(F)$ for $F \subseteq V\}$.

Claim. $\hat{b}$ is an intersecting submodular function and $Q = \hat{Q}$.

PROOF. Let $X, Y \subseteq B_{i+1} \setminus B_i$.

$$\begin{aligned}\hat{b}(X) + \hat{b}(Y) &= b(X \cup B_i) + b(Y \cup B_i) - 2b(B_i)\\ &\geqslant b((X \cap Y) \cup B_i) + b(X \cup Y \cup B_i) - 2b(B_i)\\ &= \hat{b}(X \cap Y) + \hat{b}(X \cup Y).\end{aligned}$$

Let $x \in \hat{Q}$ and $F \subseteq V$. Let B_i denote the maximal set in $\mathscr{B}$ for which $F \not\subseteq B_i$. By induction we suppose that $\lambda_x(Y) \leqslant b(Y)$ for proper subsets Y of F. We have

$$\begin{aligned}\lambda_x(F) &= \lambda_x(F \setminus B_i) + \lambda_x(F \cap B_i) \leqslant \hat{b}(F \setminus B_i) + b(F \cap B_i)\\ &= b(F \cup B_i) - b(B_i) + b(F \cap B_i) \leqslant b(F).\end{aligned}$$

Furthermore, for $B_h \in \mathscr{B}$, $b(B_h) = b(B_{h+1}) - \hat{b}(B_{h+1}\backslash B_h)$ and so

$$b(B_h) = -\sum(b(B_{i+1}\backslash B_i) : i = h, h+1, \ldots, k-1)$$
$$\leqslant -\sum \lambda_x(B_{i+1}\backslash B_i) = -\lambda_x(\bar{B}_h) = \lambda_x(B_h).$$

Therefore $\lambda_x(B_h) = b(B_h)$ and $\hat{Q} \subseteq Q$. Now let $x \in Q$ and $X \subseteq B_{i+1}\backslash B_i$. $\lambda_x(X) = \lambda_x(X \cup B_i) - \lambda_x(B_i) \leqslant b(X \cup B_i) - b(B_i) = \hat{b}(X)$, that is, $Q \subseteq \hat{Q}$. ∎

Let $\gamma_x(F) = b(F) - \lambda_x(F)$, $\hat{\gamma}_x(F) = \hat{b}(F) - \lambda_x(F)$, and $\hat{\epsilon}_x(u,v) = \min(\hat{\gamma}_x(F) : F$ a $u\bar{v}$-set). Let Q and $B_1 \subset B_2 \subset \cdots \subset B_k$ be as in Theorem 6.

THEOREM 7. *For $x \in Q$, $\hat{\epsilon}_x(u,v) = \epsilon_x(u,v)$ if $u, v \in B_{i+1}\backslash B_i$ for some i, and $\hat{\epsilon}_x(u,v) = 0$ otherwise.*

PROOF. For $u \in B_{i+1}\backslash B_i$, $v \notin B_{i+1}\backslash B_i$, we have

$$0 \leqslant \hat{\epsilon}_x(u,v) \leqslant \hat{b}(B_{i+1}\backslash B_i) - \lambda_x(B_{i+1}\backslash B_i)$$
$$= b(B_{i+1}) - b(B_i) - \lambda_x(B_{i+1}) + \lambda_x(B_i) = 0,$$

so $\hat{\epsilon}_x(u,v) = 0$.

Let $u, v \in B_{i+1}\backslash B_i$. For some $u\bar{v}$-set $X \subseteq B_{i+1}\backslash B_i$,

$$\hat{\epsilon}_x(u,v) = \hat{b}(X) - \lambda_x(X) = b(X \cup B_i) - b(B_i) - \lambda_x(X \cup B_i) - \lambda_x(B_i) \geqslant \epsilon_x(u,v).$$

On the other hand for some $u\bar{v}$-set X

$$\epsilon_x(u,v) = \gamma_x(X) \geqslant \gamma_x(X \cup B_{i+1}) + \gamma_x(X \cap B_{i+1}) - \gamma_x(B_{i+1}) \geqslant 0 + \epsilon_x(u,v) + 0,$$

that is, $\epsilon_x(u,v) = \gamma_x(X \cap B_{i+1})$. It is shown similarly that $\epsilon_X(u,v) = \gamma_X(Z)$ where $Z = (X \cap B_{i+1}) \cup B_i$. Now

$$\epsilon_x(u,v) = b(Z) - \lambda_x(Z) = \hat{b}(Z\backslash B_i) - b(B_i) - \lambda_x(Z\backslash B_i) + \lambda_x(B_i) \geqslant \hat{\epsilon}_x(u,v). \quad ∎$$

By Theorems 6 and 7, our algorithm can be used to optimize over any face of a submodular flow polyhedron, provided that the number of sets X for which we require $\lambda_x(X) = b(X)$ is not too large.

3. Feasibility and optimality criteria. A combinatorial primal feasibility theorem was given in [8]. Here we formulate combinatorial conditions for dual feasibility; they extend similar conditions for network flows.

Let B be a $(0, \pm 1)$ matrix with rows corresponding to sets X with $b''(X) < \infty$ and columns corresponding to arrows $e \in E$, defined by $b_{X,e} = +1$ if e enters X, -1 if e leaves X, and 0 otherwise.

Then (1) can be written:

$$\max dx \quad \text{subject to} \quad Bx \leqslant b''; \quad f \leqslant x \leqslant g. \tag{1}$$

The linear programming dual is

$$\min b''y + gz - fw \quad \text{subject to}$$
$$(y,z,w)\begin{bmatrix} B \\ I \\ -I \end{bmatrix} = d; \quad y, z, w \geqslant 0. \tag{2}$$

Here the components of g correspond to sets X with $b''(X) < \infty$, the components of z correspond to elements of E with $g(e) < \infty$, and the components of w correspond to elements of E with $f(e) > -\infty$. Furthermore I denotes the identity matrix of appropri-

ate size. Define a digraph $H = (V, E')$ and a cost function d' as follows:
$e = uv \in E'$ if $uv \in E$ and $g(uv) = \infty$; $d'(e) = d(e)$.
$e = vu \in E'$ if $uv \in E$ and $f(uv) = -\infty$; $d'(e) = -d(e)$.
$e = uv \in E'$ if $\epsilon_0''(uv) = +\infty$; $d'(e) = 0$.

THEOREM 8. *The linear program dual to* (1) *has a feasible solution if and only if* H *has no directed circuit of positive cost.*

By Theorem 5 it is enough to prove Theorem 8 when P is defined by a fully submodular function b.

PROOF OF NECESSITY. Suppose that H has no directed circuit of positive cost. By an elementary linear programming result, it will be enough to prove that the l.p. problem $\max(dx : Bx \leqslant 0, f' \leqslant x \leqslant g')$ is unbounded, where B denotes the constraint matrix of (1) and where f' (g') is 0 wherever f (g) is finite, and agrees with f (g) otherwise. Denote by x^0 the incidence vector of the subset of those arrows of a positive-cost directed circuit which correspond to original arrows of G. (Here $x^0(uv) = 1$ or -1 according to whether the arrow uv arises from arrow uv or vu of G.) Then $dx^0 > 0$ and $Bx^0 \leqslant 0$. Hence $\max(dx : Bx \leqslant 0, f' \leqslant x \leqslant g')$ is unbounded, as required. ∎

The proof of sufficiency in Theorem 8 will be a consequence of the algorithm and appears in §4.

By Theorem 8, only a simple shortest path calculation is required to decide whether a submodular flow problem is dual feasible. Moreover, to do this we need, instead of the submodular oracle, only an oracle to decide whether there exists a $u\bar{v}$-set X with $b''(X) < \infty$ for each $uv \in V$. (Our algorithm would detect dual infeasibility, but it is technically more convenient to check it ahead of time.)

Henceforth suppose that the submodular flow polyhedron P is determined by a fully submodular function b. As usual, appropriate values for the dual variables z, w can be determined from these for y. Namely, where $a(e)$ denotes the column of B indexed by $e \in E$, $z(e) = \max(d(e) - ya(e), 0)$ and $w(e) = \max(ya(e) - d(e), 0)$. So the complementary slackness conditions for (1) and (2) can be written:

$$d(e) - ya(e) > 0 \Rightarrow x(e) = g(e) \quad (< \infty) \quad \text{for} \quad e \in E; \tag{3a}$$

$$d(e) - ya(e) < 0 \Rightarrow x(e) = f(e) \quad (> -\infty) \quad \text{for} \quad e \in E; \tag{3b}$$

$$y(F) > 0 \Rightarrow \lambda_x(F) = b(F), \qquad F \subseteq V. \tag{3c}$$

A set F for which $\gamma_x(F)$ $(= b(F) - \lambda_x(F)) = 0$ is called *b-tight* or briefly *tight* with respect to $x \in P$.

Claim. The union and intersection of tight sets are tight.

PROOF. Let X, Y be tight. Then $0 + 0 = \gamma_x(X) + \gamma_x(Y) \geqslant \gamma_x(X \cap Y) + \gamma_x(X \cup Y) \geqslant 0 + 0$ whence equality follows everywhere. ∎

It will turn out that y can be chosen in such a way that the sets for which $y(X) > 0$ form a chain $B_1 \subset B_2 \subset \cdots \subset B_k$. One idea, introduced in [6], is that a chain of sets can be represented by a vector $\pi \in R^V$, called a *potential*. To be more precise let x be a submodular flow and $\pi \in R^V$ a vector so that, where $\bar{d}(uv)$ denotes $d(uv) + \pi(u) - \pi(v)$:

$$\text{For} \quad e \in E \qquad \bar{d}(e) < 0 \Rightarrow x(e) = f(e) \quad (> -\infty); \tag{4a}$$

$$\text{For} \quad e \in E \qquad \bar{d}(e) > 0 \Rightarrow x(e) = g(e) \quad (< \infty); \tag{4b}$$

$$\text{For} \quad u, v \in V \qquad \pi(u) > \pi(v) \Rightarrow \epsilon_x(u, v) = 0. \tag{4c}$$

Let $\pi_0 < \pi_1 < \cdots < \pi_k$ be the distinct values of π and let B_i denote $\{v \in V : \pi(v)$

$\geqslant \pi_i$) $(i \geqslant 1)$. Let $\mathscr{B} = \mathscr{B}(\pi) = \{B_1, B_2, \ldots, B_k\}$. By (4c) and the Claim, each B_i is tight. Define $y(X) = \pi_i - \pi_{i-1}$ if $X = B_i$ $(i \geqslant 1)$ and $y(X) = 0$ otherwise. Then y, x satisfy (3)(a)(b)(c). Furthermore y (and hence z, w) are integral if π and d are.

Therefore our purpose is to find vectors x, π satisfying (4)(a)(b)(c).

4. The algorithm. The optimality conditions (4)(a)(b)(c), of course, resemble and generalize network flow optimality conditions, and the optimal submodular flow algorithm can be viewed as extending classical network flow algorithms. Suppose that we are given a feasible flow x and a potential π such that (4c) is satisfied. (Any feasible x and $\pi \equiv 0$ can be used initially.) Choose an arrow j which violates (4b); the case in which (4a) is violated is similar. The *Inner Algorithm* is an algorithm which finds a new pair (x', π') such that j no longer violates (4b), such that every arrow e satisfying (4)(a)(b) before still does, and such that (4c) is maintained. It ensures that no new violations of (4)(a)(b) occur by imitating the classical primal-dual approach to minimum cost network flows, that is, it allows flow changes only on the distinguished arrow j and on arrows $e \in E$ for which $\bar{d}(e) = 0$. Similarly, it keeps (4c) by requiring that, for a given π, the tightness of each B_i be maintained.

The Inner Algorithm consists, as usual, of a sequence of flow changes and potential changes. We begin by concentrating on the attempt to remove the violation of (4b) by arrow j using *flow changes only*. (If this succeeds, the Inner Algorithm terminates. If it fails, it suggests a potential change.) Thus we are led to the following optimization problem (whose analog in the ordinary network flow case is a simple maximum flow problem):

$$\text{maximize} \quad \hat{x}_j \qquad \text{subject to} \tag{5}$$

$$\lambda_{\hat{x}}(F) \leqslant b(F) \qquad \text{for} \quad F \subseteq V; \tag{5a}$$

$$f \leqslant \hat{x} \leqslant g; \tag{5b}$$

$$\hat{x}(e) = x(e) \qquad \text{for} \quad e \neq j \quad \text{and} \quad \bar{d}(e) \neq 0; \tag{5c}$$

$$\lambda_{\hat{x}}(F) = b(F) \qquad \text{for} \quad F \in \mathscr{B}(\pi). \tag{5d}$$

As we have seen in §2 (Theorem 6), (5) defines a submodular flow polyhedron $Q = \{\hat{x} : \hat{f} \leqslant \hat{x} \leqslant \hat{g},\ \lambda_{\hat{x}}(F) \leqslant \hat{b}(F) \text{ for } F \subseteq V\}$ where $\hat{f}(e) = f(e)$ and $\hat{g}(e) = g(e)$ if $\bar{d}(e) = 0$ and $\hat{f}(e) = \hat{g}(e) = x(e)$ if $\bar{d}(e) \neq 0$. Therefore we can apply the maximum submodular flow algorithm given in [8]. We describe it here but without proof.

Given a feasible flow $\hat{x}$, construct an auxiliary digraph $H_{\hat{x}}$, having vertex-set V and three types of arrows uv with capacities $c(uv)$:

(i) A *forward* arrow uv for each arrow $uv \in E$ for which $\hat{x}(uv) < \hat{g}(uv)$; uv is given capacity $\hat{g}(uv) - \hat{x}(uv)$.

(ii) A *backward* arrow vu for every arrow $uv \in E$ for which $\hat{x}(uv) > \hat{f}(uv)$; vu is given capacity $\hat{x}(uv) - \hat{f}(uv)$.

(iii) A *jumping arrow* uv for every pair $u, v \in V$ such that $\hat{\epsilon}_{\hat{x}}(u, v) > 0$; uv is given capacity $\hat{\epsilon}_{\hat{x}}(u, v)$.

The algorithm needs an oracle to compute $\hat{\epsilon}_{\hat{x}}(u, v)$. In our case by Theorem 5 this oracle is indeed available.

Where $j = ts$, a directed path P in $H_{\hat{x}}$ from s to t having as few edges as possible, together with j, yields a directed circuit in $H_{\hat{x}}$. Given such a dicircuit C, let Δ be the minimum capacity of its edges. Increase $\hat{x}(e)$ by Δ on arrows $e \in E$ corresponding to forward arrows of C and decrease $\hat{x}(e)$ by Δ on arrows $e \in E$ corresponding to backward arrows of C. The resulting $\hat{x}$ is feasible. (If $\Delta = \infty$, then there exist feasible solutions having $\hat{x}(j)$ arbitrarily large, so the maximum flow problem is unbounded.) The procedure is then repeated and terminates if unboundedness is discovered, or if no

more such augmentations can be performed because, for some $\hat{x}, H_{\hat{x}}$ has no directed path from s to t.

The main result of [8] is that the algorithm terminates after at most n^3 augmentations, independent of the values of $\hat{f}, \hat{g}, \hat{b}$, provided only that the successive paths P are found using a slight modification of the usual labelling algorithm. Namely, one scans the vertices in the order that they are labelled, and the vertices labelled from a vertex being scanned are labelled in an order consistent with a fixed linear order of V. (The vertex sequence of the resulting path is then lexicographically least with respect to this order among all shortest paths.) The latter rule would automatically be satisfied, for example, if the graph were represented by an adjacency matrix with scanning performed in the natural way. Thus, like the modification to the labelling method which yields a polynomial bound for the network maximum flow algorithm, this one is "so simple that it is likely to be incorporated innocently into a computer implementation" [4]. This lexicographic technique was introduced by Schönsleben [16] and Lawler and Martel [13]. It then was applied to the submodular flow problem in [8], and it also has found an application [1] on a quite different level. So the method may become a standard technique.

Theorem 7 shows that it is not necessary to deal explicitly with $\hat{b}$ or $\hat{\epsilon}_x(u,v)$, and the auxiliary digraph can be defined in terms of the original vectors x, π. Namely,

(i) uv is a forward arrow if $x(uv) < g(uv)$ and $\bar{d}(uv) = 0$; $c(uv) = g(uv) - x(uv)$.

(ii) vu is a backward arrow if $x(uv) > f(uv)$ and $\bar{d}(uv) = 0$; $c(uv) = x(uv) - f(uv)$.

(iii) uv is a jumping arrow if $\epsilon_x(u,v) > 0$ and $\pi(u) = \pi(v)$; $c(uv) = \epsilon_x(u,v)$.

We consider the two possible ways in which the algorithm terminates. First, suppose that there is a dicircuit C arising in the algorithm, all of whose arrow capacities are ∞. We shall prove in this case that (1) is unbounded. Such a dicircuit is a dicircuit in the graph H of Theorem 8. Moreover, it has cost $\sum(d'(uv) : uv \in C) = \sum(d'(uv) + \pi(u) - \pi(v) : uv \in C)$. But for every forward arrow $uv \neq j$, $d'(uv) + \pi(u) - \pi(v) = \bar{d}(uv) = 0$, and $d'(j) = \bar{d}(j) > 0$; for every backward arrow uv, $d'(uv) + \pi(u) - \pi(v) = -d(vu) + \pi(u) - \pi(v) = -\bar{d}(uv) = 0$; for every jumping arrow uv, $d'(uv) + \pi(u) - \pi(v) = \pi(u) - \pi(v) = 0$. Thus C has positive cost $\bar{d}(j)$ in H, and by the "only if" part of Theorem 8 already proved, the dual of (1) is infeasible so, since (1) is feasible, (1) is unbounded.

Now suppose that the algorithm terminates with a submodular flow x such that there is no directed path from s to t in H_x. If $x(j) = g(j)$, then we have succeeded in removing the violation by j of (4b); otherwise we show how this case leads to a potential change. Let T be the set of vertices reachable from s by a directed path in H_x. Then T has the properties:

$$x(e) = g(e) \quad \text{or} \quad \bar{d}(e) \neq 0 \qquad \text{for every} \quad e \in \delta(T); \tag{6a}$$

$$x(e) = f(e) \quad \text{or} \quad \bar{d}(e) \neq 0 \qquad \text{for every} \quad e \in \rho(T); \tag{6b}$$

$$\text{there exists a } b\text{-tight } u\bar{v}\text{-set} \quad \text{or} \quad \pi(u) \neq \pi(v), \qquad \text{for every} \quad u \in T, \quad v \notin T. \tag{6c}$$

Then we decrease $\pi(v)$ by δ for every $v \in \bar{T}$, where $\delta = \min(\delta_1, \delta_2, \delta_3, \delta_4)$ and

$\delta_1 = \min(\bar{d}(e) : e \in \rho(T),\ x(e) > f(e),\ \bar{d}(e) > 0)$;

$\delta_2 = \min(-\bar{d}(e) : e \in \delta(T),\ x(e) < g(e),\ \bar{d}(e) < 0)$;

$\delta_3 = \min(\pi(v) - \pi(u) : u \in T,\ v \in \bar{T},\ \pi(v) > \pi(u)$, there exists no b-tight $u\bar{v}$-set);

$\delta_4 = \bar{d}(j)$.

It is a consequence of the definitions that $\delta > 0$ and that either $\delta = \delta_4$ or the set of vertices reachable from s in the new auxiliary digraph properly contains T. (Hence there can be at most $n - 1$ consecutive potential changes.) We must check that no new violation of (4)(a)(b) is caused by the potential change, and that (4)(c) continues to

hold. (Note, however, that the potential change may make some old violations worse.) A new violation of (4)(a)(b) could occur only if some $\bar{d}(e)$ becomes positive or negative as a result of the change in π. Suppose that $\bar{d}(e)$ becomes negative. (The other case is similar.) Then $e \in \rho(T)\backslash\{j\}$. If $\bar{d}(e)$ was zero before the change, then by (6b), $x(e) = f(e)$, so e does not violate (4a) after the change. If $\bar{d}(e)$ was positive before the change, then $\bar{d}(e) < \delta \leqslant \delta_1$ implies $x(e) = f(e)$, and again, by the definition of δ_1, (4a) is not violated. Now suppose that (u, v) violates (4c) after the change in π. Then $u \in T$, $v \in \bar{T}$ and so by (6c) $\pi(u) < \pi(v)$ before the change. But then $\delta \leqslant \pi(v) - \pi(u)$ by the definition of δ_3 and (4c) is not violated after the change.

First, we are going to show that the procedure is finite regardless of the data. In §5 we present a scaling method to make the procedure polynomially-bounded if the cost function d is integer-valued. If $\delta = \delta_4$, the potential change ends j's violation of (4b) by lowering $\bar{d}(j)$ to zero. If $\delta < \delta_4$, we continue searching for flow augmentations. We know that there will be at most n^3 flow changes between potential changes and at most $n - 1$ potential changes between flow changes. Therefore, to show that the Inner Algorithm is finite, it is enough to show that the number of "breaking" potential changes, that is, those which are immediately followed by flow changes, is finite. During the whole procedure $\pi(s)$ never changes and $\pi(t)$ decreases monotonically. When a flow change occurs, the directed path P from s to t satisfies (by the same argument as was used to prove unboundedness) $\pi(t) - \pi(s) = \sum(d(e) : e$ corresponds to a forward arrow of $P) - \sum(d(e) : e$ corresponds to a backward arrow of $P)$. Since there are only finitely many paths and $\pi(t)$ is monotone decreasing, the number of breaking potential changes is finite, as required. Since we can begin with $\pi = 0$ and a flow x found by the method of [8], by using the Inner Algorithm at most $|E|$ times, we have a finite algorithm to solve (1).

Although the computation bound provided by the above argument is not too attractive, it is valid without any assumptions on d, f, g, b. A modification described in the next section provides a polynomially-bounded algorithm in the case when d is integer-valued. Meanwhile, we can use the finiteness of the algorithm to prove sufficiency in Theorem 8.

Suppose that the dual of (1) is infeasible. Where M is a sufficiently large positive number, replace each $b(F)$ by $b(F) + M$, each $f(e)$ by $f(e) - M$, and each $g(e)$ by $g(e) + M$. The resulting version of (1) is now feasible but its dual is still infeasible, so this version of (1) is unbounded. The algorithm, therefore, will terminate with a directed circuit in an auxiliary graph, yielding a directed circuit of positive cost in H of Theorem 8. But the changes we have made to f, g, and b do not affect the properties of this circuit.

Notice that to state the algorithm we do not need (explicitly) to mention maximal submodular flows. But because the steps of the algorithm can be regarded as maximal submodular flow calculations, we were able to use the results of [8] to provide a relatively short justification of the algorithm. Another proof, more direct but more complicated, can be found in [9].

5. Scaling. In this section we assume that d is integer-valued. For simplicity, we require that no arrow $e = uv$ exists with $f(e) = -\infty$. Otherwise, if $g(e) \neq \infty$, we can replace e by $e' = vu$ with $f(e') = -g(e)$, $g(e') = \infty$, and $d(e') = -d(e)$; if $g(e) = \infty$, we can replace e by $e_1 = uv$ and $e_2 = vu$ with $f(e_1) = f(e_2) = 0$, $g(e_1) = g(e_2) = \infty$, $d(e_1) = d(e)$, and $d(e_2) = -d(e)$.

The idea behind the scaling method is the following. Suppose that d is integer-valued and we have a submodular flow x and integer-valued potential π satisfying (4c), and we apply the Inner Algorithm to $j \in E$ violating (4b) with $\bar{d}(j) = 1$ (equivalently, violating (4a) with $\bar{d}(j) = -1$). Then the Inner Algorithm will perform at most

one potential change, since the amount δ of any such change will be 1, and so $\bar{d}(j) = 0$ after the change. So in this case the Inner Algorithm requires at most n^3 flow changes. It follows that if flow x and integer-valued potential π satisfy (4)(a)(b)(c) with respect to integer-valued objective vector d, and d' differs from d by a change of 1 in exactly one component, then we can find flow x' and integer-valued potential π' satisfying (4)(a)(b)(c) with respect to d' after at most n^3 flow augmentations. Moreover, if x, π satisfy (4)(a)(b)(c) with respect to d, then $x, 2\pi$ satisfy (4)(a)(b)(c) with respect to $2d$.

Now it is quite easy to reach the vector d from the zero vector by a sequence of operations of the two kinds mentioned: changing a component by 1 and doubling the vector. Namely, let us suppose that each component of d is given in base 2 and that the biggest nonzero digit is 2^{K-1}; that is, $\max(|d(e)| : e \in E)$ consists of K digits. (The complexity of the algorithm will be proportional to this K.) Let $d^0 = d$ and for $1 \leqslant i \leqslant K$ let d^i be defined by $d^i(e) = [d^{i-1}(e)/2]$. Thus

$$d^K(e) = \begin{cases} -1 & \text{if } d(e) < 0; \\ 0 & \text{if } d(e) \geqslant 0. \end{cases}$$

Moreover, for each k, d^i can be obtained from $2d^{i+1}$ by changing a single component by 1 at most $|E|$ times.

We remark that the scaling technique was introduced by Edmonds and Karp [5] to solve the minimum cost flow problem. They scaled the capacities and demands. It is not difficult (see [18], for example) to scale the objective function instead, but this seems to have advantages only in the special case of irrational capacities. In the present context, however, a straightforward attempt to scale b will destroy its submodularity.

We can now state the algorithm.

Scaling Algorithm for (1).

Step 1. Using Theorem 8 and a shortest path method, test (1) for dual feasibility. If it is not dual feasible, stop.

Step 2. Use the algorithm of [8] to find a feasible solution x of (1). If none exists, stop.

Step 3. Put $\pi = 0$, $i = K$, $d = 0$.

Step 4. While there exists $j \in E$ with $d(j) \neq d^i(j)$, replace $d(j)$ by $d^i(j)$ and (if x, π violate (4)(a)(b)), apply the Inner Algorithm.

Step 5. If $i = 0$, stop. Otherwise, replace i by $i - 1$, π by 2π, and d by $2d$, and go to Step 4.

There is one difficulty which may seem to arise in this approach. Namely, what if the original problem is dual feasible but, with respect to an intermediate cost function d', the problem becomes dual infeasible? We show that this cannot be the case. Since we assume that $f(e) \neq -\infty$ for every $e \in E$, for nonpositive d', $d'x$ will be bounded above, that is, $\max d'x$ exists. (This assumption was made for just this purpose.) This means that this difficulty cannot occur while $i = K$. Furthermore, $2d^i \leqslant d^{i-1}$ for $1 \leqslant i \leqslant K$; therefore if $\max d^{i-1}x$ exists then $\max 2d^i x$ exists, so $\max d^i x$ exists, too, and the difficulty cannot occur for $0 \leqslant i < K$ either.

The scaling algorithm requires at most n^3 augmentations in Step 2 and at most $|E|n^3$ in Step 4 for $0 \leqslant i \leqslant K$, so it needs at most $(K + 1)|E|n^3 + n^3$ augmentations altogether.

6. Crossing submodular functions. Let b'', b', and b be as in Lemmas 3 and 4. The algorithm developed for fully submodular functions needs an oracle to compute $\epsilon_x(u, v)$. Theorem 5 shows that in order to compute a vector $x \in P$ and a potential satisfying the optimality criteria in (4) that algorithm can be used without any change for crossing submodular functions as well. There is a little complication however in

getting the dual solution y from π. The next lemma, from [7], follows simply from the definition of b, b' and b''.

LEMMA 9. *For $x \in P$, a b-tight set X partitions into b'-tight sets A_i. The complement of a b'-tight set A_i partitions into the complements of b''-tight sets A_{ij}.*

As a consequence a b-tight set $B_k = \bigcup_i A_i = \bigcup_i \bigcap_j A_{ij}$, where each A_{ij} is b''-tight. Let $\mathcal{K}(B_k) = \{A_{ij}\}$. Define $y(X)$ to be $\sum \pi_k - \pi_{k-1}$, where the sum is over subscripts k for which $X \in \mathcal{K}(B_k)$ (the empty sum is 0). It is not hard to see that $ya(uv) = \pi(v) - \pi(u)$ and hence that x, y satisfy (3). From the algorithmic point of view we have to be able to construct the family $\mathcal{K}(B)$. Here we outline the method of [7].

We need a slightly stronger oracle which computes not only $\epsilon''_x(u, v) = \min(\gamma''_x(F) : F$ a $u\bar{v}$-set) but also determines a set F where the minimum is attained.

By Lemma 9 for $v \in B$, $B(v) := \bigcap(X : X$ is b''-tight, $v \in X)$ is b'-tight and $B(v)$ can be computed. The components of the hypergraph $\{B(v) : v \in B\}$ partition B into b'-tight sets A_i.

LEMMA 10 [7]. *A b'-tight set A, for which the hypergraph $H = \{B(v) : v \in A\}$ is connected, can be obtained constructively as the intersection of pairwise co-disjoint b''-tight sets.*

PROOF. This is obvious for a set $A = B(v)$ and for $A = V$. Suppose that X and Y are crossing b'-tight sets and we have co-disjoint b''-tight sets X_i and co-disjoint b''-tight sets Y_j such that $X = \bigcap X_i$, $Y = \bigcap Y_j$. Then $X \cup Y = \bigcap(Z : Z = X_i \cup Y_j$, X_i and Y_j are crossing). Here any set Z is b''-tight and the minimal sets Z are co-disjoint, that is, we have obtained $X \cup Y$ in the required form. The lemma now follows from a simple induction. ∎

Applying Lemma 10 to the components A_i, we get the members of $\mathcal{K}(B)$.

References

[1] Cunningham, W. H. (1984). Testing Membership in Matroid Polyhedra. *J. Combinatorial Theory* **B36** 161–188.

[2] Dunstan, F. D. J. (1976). Matroids and Submodular Functions. *Quart. J. Math. Oxford* **27** 339–347.

[3] Edmonds, J. (1970). Submodular Functions, Matroids, and Certain Polyhedra. R. K. Guy et al., eds., *Combinatorial Structures and Applications*. Gordon and Breach, New York, 69–87.

[4] ——— and Giles, R. (1977). A min-max Relation for Submodular Functions on Directed Graphs. *Ann. Discrete Math.* **1** 185–204.

[5] ——— and Karp, R. M. (1972). Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems, *J. ACM* **19** 248–264.

[6] Ford, Jr., L. R. and Fulkerson, D. R. (1962). *Flows in Networks*. Princeton University Press, Princeton, N.J.

[7] Frank, A. (1982). An Algorithm for Submodular Functions on Graphs. *Ann. Discrete Math.* **16** 97–120.

[8] ———. (1984). Finding Feasible Vectors of Edmonds-Giles Polyhedra. *J. Combinatorial Theory* **B36** 221–239.

[9] ———. (1984). Submodular Flows. in: W. R. Pulleyblank, ed. *Progress in Combinatorial Optimization*. Academic Press, 147–165.

[10] ———. (1984). Generalized Polymatroids. in: A. Hajnal et al., eds. *Finite and Infinite Sets, Eger 1981*. North Holland, 285–294.

[11] Fujishige, S. (1978). Algorithms for Solving the Independent Flow Problems. *J. Oper. Res. Soc. Japan* **21** 189–203.

[12] ———. (1984). Structures of Polytopes Determined by Submodular Functions on Crossing Families. *Math. Programming* **29** 125–141.

[13] Grötschel, M., Lovász, L. and Schrijver, A. (1981). The Ellipsoid Method and its Consequences in Combinatorial Optimization. *Combinatorica* **1** 169–197.

[14] Hassin, R. (1982). Minimum Cost Flow with Set Constraints. *Networks* **12** 1–21.

[15] Lawler, E. L. and Martel, C.U. (1982). Computing Maximal Polymatroidal Network Flows. *Math. Oper. Res.* **7** 331–347.

[16] Lovász, L. (1977). Flats in Matroids and Geometric Graphs, P. J. Cameron ed., *Combinatorial Surveys*, Academic Press, New York, 45–86.

[17] Lucchesi, C. and Younger, D. H. (1978). A Minimax Theorem for Directed Graphs. *J. London Math. Soc.* **17** 369–374.

[18] Röck, H. (1980). Scaling Techniques for Minimal Cost Network Flows. U. Pape ed., *Discrete Structures and Algorithms*, Carl Hanser, Munchen, 181–191.

[19] Schönsleben, P. (1980). Ganzzahlige Polymatroid—Intersektions—Algorithmen. thesis, ETH, Zürich.

[20] Schrijver, A. (1984). Total Dual Integrality from Directed Graphs, Crossing Families, and Sub- and Supermodular Functions. in: W. R. Pulleyblank (ed.) *Progress in Combinatorial Optimization*, Academic Press.

CUNNINGHAM: DEPARTMENT OF MATHEMATICS AND STATISTICS, CARLETON UNIVERSITY, OTTAWA, ONTARIO, CANADA K1S 5B6

FRANK: RESEARCH INSTITUTE FOR TELECOMMUNICATION, BUDAPEST, HUNGARY