Annals of Discrete Mathematics 19 (1984) 135–146
© Elsevier Science Publishers B.V. (North-Holland)

ALGEBRAIC FLOWS

A.M. Frieze

Department of Computer Science & Statistics
Queen Mary College
University of London
Mile End Road
London E1 4NS, G.B.

We consider a natural generalisation of the maximum value flow problem, where flow values are elements of an ordered d-monoid. Assuming conservation of flow at vertices and capacity constraints on the arcs we are able to prove a Max-Flow Min-Cut Theorem using a flow augmenting path algorithm. If the monoid is weakly cancellative then we can make the algorithms polynomially bounded.

INTRODUCTION

We consider here the problem studied by Hamacher [3]: we are given

a digraph $D = (V,A)$ with vertices V and arcs $A \subseteq V \times V$ which is loopless and symmetric (1.1a)

a totally ordered commutative d-monoid $(H,*,\leqslant)$ with identity e (1.1b)

i.e. a set H totally ordered by $<$ and an associative, commutative binary operation $*$ satisfying

(1) $a \leqslant b$ implies $a*c \leqslant b*c$ for all $a,b,c \in H$.

(2) $a < b$ implies there exists $c > e$ such that $a*c = b$ for $a,b \in H$.

a capacity function $c:A \to H$ such that $c(u) > e$ for $u \in A$. (1.1c)

2 special vertices s and t. (1.1d)

An s-t flow is a function $f:A \to H$ satisfying

$$e \leqslant f(u) \leqslant c(u) \text{ for } u \in A \tag{1.2a}$$

$$f(v:V) = f(V:v) \text{ for } v \neq s,t \tag{1.2b}$$

where for sets $X,Y \subseteq V$ $X:Y = \{(v,w) \in A : v \in X, w \in Y\}$ and for $S \subseteq A$

$$f(S) = \mathop{\divideontimes}_{(v,w)\in S} f(v,w)$$

Naturally $v:V$ is an abbreviation of $\{v\}:V$.
The value val(f) of flow f is defined to be $f(s:V)$.

The problem of finding the s-t flow that maximises val(f) is a natural generalisation of the classical maximum flow problem of Ford and Fulkerson [2] and examples can be found in [3]. Hamacher made the point that in practice, minimum capacity cuts are more important than maximum flows, which stresses the importance of a Max-Flow Min-Cut Theorem.

Unfortunately, 1.2 is not quite restrictive enough to make a 'sensible' problem and further restrictions are required. Indeed we can see from the following example that the given definition of flow leaves problems:

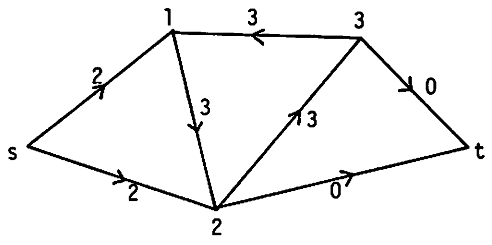

Figure 1

Here $H = (R+, \max, \leqslant)$. Note that 1.2b is satisfied and yet $f(s:V) = f(V:t)$ as one might expect. We will consider 2 essentially different ways of overcoming this problem.

Cuts

As usual a set $X \subseteq V$ such that $s \in X$, $t \in \bar{X} = V-X$ generates a cut $X:\bar{X}$ which has capacity $c(X:\bar{X})$. Note that if in the above example we assume $f(u) = c(u)$ for each arc, then where $X = (s,1,2,3)$ we have $\mathrm{val}(f) = 2 > 0 = c(X:\bar{X})$ and so there is no Max-Flow Min-Cut Theorem for arbitrary flows.

Hamacher dealt with this problem by putting a return arc (t,s) and replacing 1.2b by

$$f(X:\bar{X}) = f(\bar{X}:X) \qquad \text{for all } X \subseteq V \tag{1.3}$$

and assuming that $(H,*)$ has a weakly cancellative property (see Section 3). Thus to make any progress we must restrict our attention to flows with additional properties.

In Section 2 we consider a class of flows for which we are able to prove, constructively, a Max-Flow Min-Cut Theorem without making any extra assumptions about the d-monoid H. Unfortunately, this algorithm is not (proved to be) polynomially bounded.

In Section 3 we consider a different class of flows and assume H is weakly cancellative so that we again have a Max-Flow Min-Cut Theorem and this time a polynomially bounded algorithm for constructing a maximum flow.

DECOMPOSABLE FLOWS

For a flow f led D(f) be the digraph (V,A(f)) where $A(f) = \{u \in A: f(u) > e\}$. A flow f is a P-flow if

D(f) is a simple path P(f) from s to t (plus isolated vertices) (2.1a)

there exists $a = a(f)$ H such that $f(u) = a$ for $u \in A(f)$ and $f(u) = e$ for $u \in A-A(f)$. (2.1b)

A flow f is decomposable if there exist P-flows $f_1, f_2, \ldots, f_k$ ($k = k(f)$) such that $f = f_1 * f_2 * \ldots f_k$(i.e. $f(u) = f_1(u) * f_2(u) * \ldots f_k(u)$ for $u \in A$).

The main result of this section is the following

Theorem 2.1. The maximum value of a decomposable s-t flow is equal to the minimum capacity of a cut separating s and t.

This will be proved using a flow augmenting path argument, but first we need to introduce some notation and prove some simple lemmas.

For $a \in H$ $\mathrm{dom}(a) = \{b \in H: a*b = a\}$ and for a flow f $\mathrm{dom}(f) = \mathrm{dom}(\mathrm{val}(f))$.

Lemma 2.1. $a \in \mathrm{dom}(b)$ and $c > b$ implies $a \in \mathrm{dom}(c)$

Proof. Let $c = b*d$, then $a*c = a*b*d = b*d = c$

Lemma 2.2. If f is a decomposable flow then $\mathrm{val}(f) \succcurlyeq f(u)$ for all $u \in A$.

Proof. Straightforward by induction on k(f).

(note that our example of Figure 1 shows this is not true in general for d-monoid flows).

For a decomposable flow let $K = K(f) = [k(f)]$ where for positive integer n $[n] = \{1,2,\ldots n\}$. For $I \subseteq K$ let $f_I = \underset{i \in I}{\Join} f_i$.

Lemma 2.3. Let f be a decomposable flow. If there exists $u \in A$ and sets I,J $I \subset J \subset K$ such that

(1) $f_i(u) > e$ for $i \in L = J-I$

(2) $f_I(u) = f_J(u)$

then $\mathrm{val}(f) = \mathrm{val}(f_M)$ where $M = K-L$.

Proof. Let $a = f_L(u) = \mathrm{val}(f_L)$ by (1). Let $b = f_I(u)$ and $c = \mathrm{val}(f_M)$. Then

$b \leqslant f_M(u) \leqslant c$ by Lemma 2.2. $a \in \text{dom}(b)$ by (2). Applying Lemma 2.1 gives $a \in \text{dom}(c)$.

Lemma 2.4. Let f be a decomposable flow and $X:\bar{X}$ a cut. Then

$$\text{val}(f)*f(\bar{X}:X) = f(X:\bar{X}) \tag{2.2}$$

Proof. Equation 2.2 holds for P-flows using the fact that an s-t path has one more arc in $X:\bar{X}$ than it has in $\bar{X}:X$. The truth of 2.2 in general follows by combining the separate equations for each path.

Note that 2.2 is essentially Hamacher's condition and generally speaking one has to add extra conditions to 1.2b in order that 2.2 or 1.3 holds.

Corollary 2.5. Let f be a decomposable flow and $X:\bar{X}$ a cut. Then

$$\text{val}(f) \leqslant c(X:\bar{X}) \tag{2.3}$$

Proof.

$$\begin{aligned}\text{val}(f) &\leqslant \text{val}(f)*f(\bar{X}:X)\\ &= f(X:\bar{X})\\ &\leqslant c(X:\bar{X})\end{aligned}$$

Corollary 2.6. If f is a decomposable flow then

$$f(s:V) = f(V:t)$$

Proof. Put $X = V-\{t\}$ in 2.2.

We define next the incremental graph $G(f) = (V,E(f))$ with respect to a given flow f. For $(v,w) \in V \times V$ let $\rho(v,w) = (w,v)$ then

$$\begin{aligned}E(f) = \{u = (v,w) \in V \times V):\ &(a)\ v = t \text{ and } w = s\\ &\text{and}\\ &(b)\ u \in A, f(u) < c(u) \text{ and}\\ &\qquad \{x : x*f(u) = c(u)\} \cap \text{dom}(f) = \emptyset\\ \text{or}\qquad&\\ &(c)\ \ \rho(u) \in A \text{ and } f(\rho(u)) \notin \text{dom}(f)\ \}\end{aligned}$$

The set of arcs EF of G(f) defined in (b) are called forward arcs and the set of arcs EB defined in (c) are called backward arcs.

A simple path from s to t in G(f) is called a flow augmenting path with respect to f. We next prove

Lemma 2.5. Let f be a decomposable flow for which G(f) has no flow augmenting paths. Then f is a maximum flow.

Proof. Let $X = \{v \in V$: v is reachable from s by a path in $G(f)\}$. Then $s \in X$ and $t \in \bar{X}$ by assumption.

For $u \in X:\bar{X}$ let $g(u) \in \mathrm{dom}(f)$ be such that $f(u)*g(u) = c(u)$. g(u) exists else we can reach a vertex of X. Note also that similarly $u \in \bar{X}:X$ implies $f(u) \in \mathrm{dom}(f)$. Thus

$$\begin{aligned} \mathrm{val}(f) &= \mathrm{val}(f)*f(\bar{X}:X)*g(X:\bar{X}) \\ &= f(X:\bar{X})*g(X:\bar{X}) \\ &= c(X:\bar{X}) \end{aligned}$$

Thus f is a maximum flow and X generates a minimum cut.

Proving the converse result i.e. that given a flow augmenting path we can actually augment the flow has proved somewhat more difficult. This is in effect why we are looking at decomposable flows.

Lemma 2.6. If f is a decomposable flow and G(f) has a flow augmenting path then there is a decomposable flow $\hat{f}$ with $\mathrm{val}(\hat{f}) > \mathrm{val}(f)$.

Proof. Let P be a flow augmenting path with respect to f and let the arcs X of P be divided into forward arcs XF and backward arcs XB (it simplifies things slightly to refer to the arcs in XB as they are in A instead of in $\rho(A)$). For $u \in XF$ let $g(u) \notin \mathrm{dom}(f)$ be such that $f(u)*g(u) = c(u)$. Let $\theta = \min(\min(g(u): u \in XF), \min(f(u): u \in XB)$.

As expected we are going to construct a flow $\hat{f}$ for which $\mathrm{val}(\hat{f}) = \mathrm{val}(f)*\theta > \mathrm{val}(f)$ as $\theta \notin \mathrm{dom}(f)$ by construction.

A problem arises for $u \in XB$ if we want to 'subtract' θ from f(u) i.e. choose $\hat{f}(u)$ such that $\hat{f}(u)*\theta = f(u)$, as one expects to do if one follows an analogous procedure to the classical real case. The problem arises from the possibility of there being several choices for $\hat{f}(u)$ and it is not clear (to the author) which, if any, maintain conservation of flow. This we hope will justify the rather complex procedure that we now describe. There are 3 phases to the update of f.

Phase 1. At the end of this phase, the decomposition of f will have been amended (but not f itself) so that for each $u \in XB$

$$\text{there exists } r = r(u) \text{ such that } \theta = f_1(u)*f_2(u)*\dots*f_r(u). \qquad (2.3)$$

Suppose there exists $u \varepsilon XB$ for which 2.3 fails, then for some $p \leqslant k(f)$ we have

$$f_{\lceil p-1\rceil}(u) < \theta < f_{\lceil p\rceil}(u)$$

Define $x,y \varepsilon H$ by $\theta = f_{\lceil p-1\rceil}(u)*x$ and $x*y = f_{\lceil p\rceil}(u)$ which will be possible as $f_{\lceil p\rceil}(u) > x$ necessarily.

Now renumber $f_{p+1},\ldots,f_k$ as $f_{p+2},\ldots,f_{k+1}$ to leave a gap for a new f_{p+1}.

Let $Q = P(f_p)$. Replace $a(f_p)$ by x and add a new P-flow f_{p+1} to the decomposition with $P(f_{p+1}) = Q$ and $a(f_{p+1}) = y$. Clearly 2.3 now holds for u with $r = p+1$ and if 2.3 held for $u' = u$ before this change, it will still hold but $r(u')$ may have increased by 1.

We shall use the term splitting f_p using x,y to denote the above construction.

Phase 2. At the end of this phase the decomposition of f will have been amended so that

> there is a sequence $a_1,a_2,\ldots,a_p$ of members of $H-\{e\}$ such that for each $u \varepsilon XB$ there exists a permutation of the non-identity members of the sequence $f_1(u),f_2(u),\ldots,f_r(u)$, $r = r(u)$, which is identical to $a_1,a_2,\ldots,a_p$ (thus $\theta = a_1{}^*a_2{}^*\ldots a_p$). (2.4)

Suppose then that $XB = \{u_1,u_2,\ldots,u_l\}$ and assume inductively that 2.4 holds for $u \varepsilon \{u_1,u_2,\ldots,u_{m-1}\}$, which holds trivially for $m = 2$. We show now how to extend 2.4 to include u_m.

Suppose then that the non-identity members of the sequence $f_1(u_m),f_2(u_m),\ldots f_r(u_m)$, $r = r(u_m)$ are $b_1,b_2,\ldots,b_q$.

We then iteratively do the following:

```
if $a_1 = b_1$: $c_1 := a_1$; continue with $a_2,\ldots a_p,b_2,\ldots b_q$.
if $a_1 > b_1$: let $a_1 = a_1'*b_1$; $c_1 := b_1$; for $i := 1$ to $m-1$ let $j(i)$ be such that $a_1$
        corresponds to $f_{j(i)}(u_i)$ in the given permutation of the non-identity
        members of $f_1(u_1)$; for $j\epsilon\{j(1),\ldots j(m-1)\}$ split $f_j$ using $a_1',b_1$;
        continue with $a_1',a_2,\ldots a_p,b_2,\ldots b_q$.
if $a_1 < b_1$: let $b_1 = b_1'*a_1$; $c_1 := a_1$; split the P-flow $f_j$ such that $f_j(u_m)$
        corresponds to $b_1$, using $b_1',a_1$; continue with $a_2,\ldots a_p,b_1',b_2,\ldots b_q$.
```

`After at most p+q-1 iterations of the above we will have produced a sequence

$c_1, c_2, \ldots c_{p'}$ and have exhausted one (or both) of the sequences $a_1, \ldots a_p$ or $b_1, \ldots b_q$. Let $d_1, \ldots d_{q'}$ denote the remainder of the unexhausted sequence. For convenience assume that $b_1, \ldots b_q$ gets exhausted first, the other case is similar.

We now find that for $i = 1, \ldots, m-1$ the non-identity members of the sequence $f_1(u_i), \ldots f_r(u_i)$, $r = r(u_i)$ are a permutation of $c_1, \ldots c_{p'}, d_1, \ldots d_{q'}$ and that the non-identity members of the sequence $f_1(u_m), \ldots f_r(u_m)$, $r = r(u_m)$ are a permutation of $c_1, \ldots c_{p'}$ and further that $d_1 * d_2 * \ldots * d_{q'} \in \mathrm{dom}(c_1 * c_2 * \ldots c_{p'})$. Note that the P-flows corresponding to $d_1, \ldots d_{q'}$ are distinct from those corresponding to $c_1, \ldots c_{p'}$ as the former have not been affected by the above procedure. We can then apply Lemma 2.3 to remove the flows corresponding to $d_1, \ldots d_{q'}$ and we find that 2.4 holds with u_m included.

Phase 3. Let $a_1, \ldots a_p$ be as in 2.4 and let $S = \{a_1, \ldots a_p\}$. For each $a \in S$ do the following: let $I = \{i: a(f_i) = a\}$. Construct an integral flow g by sending one unit of flow along each path $P(f_i)$ for $i \in I$. Let m be equal to the number of times a occurs in $a_1, \ldots a_p$. Augment g by an amount m using the flow augmenting path P to create a new integral flow h. Decompose h into a set of flows of value 1 along paths from s to t as follows: find a path Q from s to t using only arcs u for which $h(u) > 0$. Decrease h by 1 on each arc of Q. Store Q. Repeat until there is no path from s to t with positive h flow in all its arcs. Let $Q_1, \ldots Q_q$ be the paths stored. Replace the P-flows f_i for $i \in I$ by the set of P-flows with paths $Q_1, \ldots Q_q$ and flow value a.

It should be clear that the above 3 phase procedure does in fact augment f to a flow $\hat{f}$ with $\mathrm{val}(\hat{f}) = \mathrm{val}(f) * \theta$.

To complete the proof of Theorem 2.1 we must show that we need only augment a finite number of times. This is not difficult because after an augmentation along a path P either a forward arc of P becomes saturated or a backward arc of P becomes flowless. Thus if we apply the obvious analogue of the Dinic Algorithm [1], the same arguments can be used to show that no more than $O(|V|^4)$ augmentations are needed until G(f) has no flow augmenting paths.

Complexity of the algorithm. Although finite, the algorithm above is not polynomial as the size k(f) of the decomposition seems to be capable to growing exponentially. The problem occurs in Phase 2 where k could double (we can easily ensure that only |A| P-flows are created for each $a \in S$ in Phase 3 by reducing h along $a \in Q$ by enough to create at least one new h flowless arc).

In some cases the algorithm can be made polynomial by ensuring that all the paths $P(f_i)$, $i = 1,\dots k$ are distinct - if $P(f_i) = P(f_j)$ we can replace $a(f_i)$ by $a(f_i)*a(f_j)$ and delete f_j. Thus if we consider a class of digraphs in which the number of s-t paths is bounded by a polynomial in $|V|$ then the algorithm becomes polynomial.

ACYCLIC FLOWS

Recall that for a flow f satisfying 1.2 we define $A(f) = \{u \in A: f(u) > e\}$. We say that a flow is acyclic if the digraph $D(f) = (V,A(f))$ has no (directed) cycles. Note that our 'problem' flow of Figure 1 is not acyclic. We now assume that $t:V = V:s = \emptyset$ without any real loss of generality.

Lemma 3.1. Let f be an acyclic flow and let $X:\bar{X}$ be a cut separating s and t. Then

$$f(s:V)*f(\bar{X}:X) = f(X:\bar{X}) \tag{3.3}$$

Proof. For the purposes of the Lemma we assume $V = \{1,\dots n\}$, $s = 1$ and $t = n$ and since D(f) is acyclic we can assume that $f(i,j) > e$ implies $i < j$. We temporarily augment A with those arcs (i,j) where $1 \leqslant i < j \leqslant n$ and $(i,j) \notin A$. We put $f(i,j) = e$ for such arcs and note that f is still an acyclic flow.
Let $p = p(X) = \max(i:i \in X)$, $q = q(X) = \min(i:i \in \bar{X})$

Case 1:$q > p$. Thus $X = \{1,\dots p\}$, $\bar{X} = \{p+1,\dots n\}$. Note that $f(\bar{X}:X) = e$ in this case. We verify 3.1 by induction on p.

If $p = 1$ then $s:V = X:\bar{X}$ and so there is nothing to prove.

Suppose then we have verified this case for $|X| < p$ and suppose now that $|X| = p$. Let $Y = \{1,\dots p-1\}$ then

$$\begin{aligned} f(Y:\bar{Y}) &= f(Y:p)*f(Y:\bar{X}) \\ &= f(p:\bar{X})*f(Y:\bar{X}) \qquad \text{using 2.1b} \\ &= f(X:\bar{X}) \end{aligned}$$

and the induction step is easily completed.

Case 2: $q < p$. We proceed by induction on p-q. Case 1 provides the base $p-q < 0$. Now

$$\begin{aligned} f(s:V)*f(\bar{X}:X) &= f(s:V)*f(\bar{X}-\{q\}:X)*f(q:X) \\ &= f(s:V)*f(\bar{X}-\{q\}:X \cup \{q\})*f(q:X) \end{aligned}$$

since $f(\bar{X}-\{q\}:=q) = e$. Also

$$f(X:\bar{X}) = f(X:q)*f(X:\bar{X}-\{q\})$$
$$= f(q:X)*f(q:\bar{X}-\{q\})*f(X:\bar{X}-\{q\})$$

since $f(X:q) = f(V:q) = f(q;V)$. Thus

$$f(X:\bar{X}) = f(q:X)*f(X \bigcup \{q\}:\bar{X}-\{q\}).$$

Thus 3.1 holds if

$$f(s:V)*f(\bar{X}-\{q\}:X \bigcup \{q\}) = f(X \bigcup \{q\}:\bar{X}-\{q\})$$

But this can be assumed if we use induction on p-q, since if $p(X) > q(X)$ we have $p(X) = p(X \bigcup \{q(X)\})$ and $q(X \bigcup \{q(X)\}) > q(X)$.

Note that the conclusions of Corollaries 2.5 and 2.6 thus hold for acyclic flows.

Flow augmenting paths are defined exactly as in Section 2, (except that we can re-define $EF = \{u \in A: c(u)-f(u) \notin \text{dom}(f)\}$ which has the advantage of being simpler and computable in $O(|A|)$ time).

Lemma 3.2. If f is an acyclic flow and G(f) has no flow augmenting paths then f is a maximum flow.

Proof (identical to Lemma 2.5).

We now assume that our d-monoid obeys the weak cancellative rule

$$a*b = a*c \text{ implies } b = c \text{ or } a*b = a \text{ for } a,b,c \in H. \qquad (3.2)$$

Then it can be shown that there exists another ordered set $(I,\leqslant)$ and a surjective function in: $H \to A$ satisfying

$$a \leqslant b \text{ implies } in(a) \leqslant in(b) \qquad (3.3a)$$

$$in(a*b) = \max(in(a),in(b)) \qquad (3.3b)$$

$$in(a) < in(b) \text{ implies } a*b = b \qquad (3.3c)$$

$$a*b = a*c \text{ and } in(a) = in(b) = in(c) \text{ implies } b = c \qquad (3.3d)$$

It follows that the element c defined in 1.1(b)(2) is unique. We shall denote this by a-b and extend the definition of - to a-a = e.

Note that $in(a-b) = in(a)$ for $a > b$.

For $i \in I$ let $H(i) = \{a \in H:in(a) = i\}$

Let $K = \{i \in I:|H(i)| = 1\}$.

Proofs of all these results can be found in Zimmermann [5].

Given a flow augmenting path P and XB,XF and θ as defined in Lemma 2.5 we look in this case at the much more straightforward way of updating the flow:

SIMPLE UPDATE

$$\text{Let } \hat{f}(a) = f(a)*\theta \qquad a \in XF$$
$$= f(a)-\theta \qquad a \in XB$$
$$= f(a) \qquad a \notin P$$

We assume that we start the algorithm with $f(a) = e$ for $a \in A$. Now SIMPLE UPDATE does not in fact guarantee that f is a flow, let alone acyclic. We next define a quasi-flow $f:A \to H$ to be one that satisfies 1.2a and

$$\text{in}(f(a)) \preccurlyeq \text{in}(f) \qquad \text{where in}(f) = \text{in}(\text{val}(f)) \qquad (3.4a)$$

$$f(v:V) = f(V:v) \qquad (3.4b)$$

for all $v \in V$ such that $\text{in}(f(v:V)) = \text{in}(f)$ or $\text{in}(f(V:v))) = \text{in}(f)$.

$$f(s:V) = f(V:t) \qquad (3.4c)$$

It is easy to prove

Lemma 3.3. If f is a quasi-flow then after SIMPLE UPDATE f is also a quasi-flow. (Some proofs will be omitted because they are obvious or a similar result has been proved in Hamacher. Indeed Hamacher used SIMPLE UPDATE but assumed that $H(i)$ had its own identity e_i and let $a-a = e_i$ for a $H(i)$. This means that f remains a flow but at the 'expense' of introducing e_i).

Thus from now on flow augmenting paths are defined in terms of quasi-flows.

Lemma 3.3. If f is a quasi-flow and if G(f) has no flow augmenting paths then there exists a cut $X:\bar{X}$ such that $\text{val}(f) = c(X:\bar{X})$.

Lemma 3.4. Let f be a quasi-flow. Then there exists an acyclic flow f" such that val(f) = val(f").

Proof. First define f' by

$$f'(a) = f(a) \qquad \text{if } (f(a) \notin \text{dom}(f)) \text{ or } (\text{in}(f) \in K \text{ and } f(a) = \text{val}(f))$$
$$= e \qquad \text{otherwise}$$

(we recognise $\text{in}(f) \in K$ by $\text{val}(f) \neq e$ and $\text{val}(f)*\text{val}(f) = \text{val}(f)$).

It is easy to see that f' is a flow because of 3.4b. Note that val(f') = val(f). However f' may not be acyclic.

If in(f) ε K find an s-t path Q in D(f) and let f" be the P-flow with path Q and value val(f). Such a path exists by 3.4b and 3.4c.

Otherwise if D(f') has a cycle C let θ = min(f'(a): a ε C). Replace f"(a) by f"(a)-θ for a ε C. One can show fairly easily that f' remains a flow and that val(f') is unchanged. We continue until we obtain an acyclic flow f".

We can again use the Dinic algorithm to search for flow augmenting paths and again because after augmentation along a path P, one forward arc of P becomes saturated or one backward arc of P becomes flowless, the algorithm will run in $O(|V|^4)$ time. We are thus led to claim the following result:

Theorem 3.1. The maximum value of an acyclic flow is equal to the minimum capacity of a cut if the d-monoid is weakly cancellative.

Other algorithms e.g. Malhotra, Kumar and Maheshwari [4] use augmenting paths in a less direct manner and it is worth checking that they do not create problems: these algorithms proceed in a sequence of stages. The aim of each stage is to find a flow $\tilde{f}$ in the layered subgraph LG(f) = (V,E(f)) made up of the shortest s-t paths in D(f). The flow $\tilde{f}$ is chosen to saturate each s-t path in LG(f). These algorithms have straightforward algebraic analogues where we add and subtract and compare as if H was the set of reals (we never need to compute a-b where a < b).

Having computed $\tilde{f}$, a new flow f' is computed by

$$f'(a) = f(a)*\tilde{f}(a) \qquad a \in EF \qquad (3.5a)$$

$$f'(a) = f(a)-\tilde{f}(a) \qquad a \in EB \qquad (3.5b)$$

We need to check that 3.4 holds for f'. We note first that it can easily be shown that in the algebraic analogue of the algorithm of [4] that f' satisfies 3.4c in LF(g). It follows from V:s = t:V = 0 and the definition of E(f) that f' satisfies 3.4c in G also, and that

$$f'(s:V) = f(s:V)*\tilde{f}(s:V) \qquad (3.6)$$

We can consider two cases:

Case 1: in(f') > in(f). By considering only those arcs a ε E(f) for which in (f(a)) = in(f) we see that they must be forward arcs and these are the only arcs that need be considered in confirming 3.4a and 3.4b, which follows as f is a flow.

Note that this necessarily includes the case in(f') ε K.

Case 2: i = in(f') = in(f) $\notin$ K. In this case H(i) is either an ordered group or the positive cone of an ordered group [5] and by considering the same set of arcs as in the first case we can reduce to the group case which is essentially the same as the real case.

It is suggested that one works with quasi-flows until no more flow augmenting paths can be found. Only then do we reduce the quasi-flow to an acyclic flow. It only remains to check that we can do this in $O(|V||A|)$ time. We outline next how this can be done.

We use depth-first search on $D(f)$, stacking vertices as they are visited and removing them from the stack after all neighbours of a vertex have been visited. If the next neighbour of the vertex currently being visited is on the stack then a cycle has been found. In $O(|V|)$ time we can examine the cycle, reduce the flow in it, remove one (or more) arcs from $D(f)$ and restart the search at the tail of the first arc (in the order in which used in the search) removed. We continue until no more cycles are found in this manner and the search finishes.

To bound the total time taken we apportion the work done into work done (i) between finding cycles and restarting the search, (ii) traversing arcs between finding cycles that do not lie on the next cycle found and (iii) traversing arcs between finding cycles that lie on the next cycle found.

For each cycle found the time spent doing (i) and (iii) is $O(|V|)$ and as no more than A cycles can be found, because we delete at least one arc after finding one.

REFERENCES

[1] Dinic, E.A., Algorithm for solution of a problem of maximum flow in a network with power estimation, Soviety Mathematics Doklady 11 (1970) 1277-1280.

[2] Ford, L.F., and Fulkerson, D.R., Flows in networks (Princeton University Press, Princeton, 1962).

[3] Hamacher, H., Maximal algebraic flows: algorithms and examples, in: Pape, U. (ed.), Discrete Structures and Algorithms (Hansser, Munich, 1980) 153-166.

[4] Malhotra, V.M., Kumar, M.P., and Maheshwari, S.N., An $O(|V|^3)$ algorithm for finding maximum flows in networks, Information Processing Letters 7 (1978) 277-278.

[5] Zimmermann, U., Linear and combinatorial optimization in ordered algebraic structures, Annals of Discrete Mathematics 10 (North-Holland Publishing Co., Amsterdam, 1981).