Algorithmica (2004) 38: 529–537
DOI: 10.1007/s00453-003-1054-1

Algorithmica

Minimum *k* Arborescences with Bandwidth Constraints[1]

Mao-cheng Cai,[2] Xiaotie Deng,[3] and Lusheng Wang[3]

Abstract. Let $G = (V, A)$ be a digraph with source r. A weight function w and bandwidth constraint function b (positive integer) on A are given. We present algorithms for finding k r-arborescences in G with minimum total weight and with bandwidth constraints.

Key Words. Digraph, Arborescence, Matroid, Polymatroid, Polymatroid intersection, Maximum flow, Algorithm and complexity.

1. Introduction.

Let $G = (V, A)$ be a finite digraph with vertex set V and arc set A. Parallel arcs are allowed, but loops are not. An arc from u to v is denoted by uv. A vertex $r \in V$ is called a (graphical) *source* of G if there is no arc entering r in G. An *r-arborescence* of G is defined as a spanning tree directed in such a way that each vertex of G, except r, has one arc entering it.

Consider a scenario where each arc $a \in A$ has an integer bandwidth $b(a)$, indicating the number of times such an arc can be used. The problem of *k arborescences with bandwidth constraints* is to find k arborescences rooted at the k given sources $r_1, r_2, \ldots, r_k$ in V, covering each arc $a \in A$ at most $b(a)$ times. We may assume that the k required arborescences are rooted at the same source r. Otherwise we add a new source r and k new arcs rr_i to G, and set $b(rr_i) = 1$ and $w(rr_i) = 0$ $(i = 1, 2, \ldots, k)$. Clearly, k required arborescences in G correspond to k arborescences rooted at r in the new digraph. This version is called the problem of *k r-arborescences with bandwidth constraints*.

In a more general setting, we consider the problem of *minimum k r-arborescences with bandwidth constraints*: In addition to the upper bound function b (positive integer) on A, a weight function w on A is given, we are to find k r-arborescences (not necessarily distinct) in G having minimum total weight and covering each arc $a \in A$ at most $b(a)$ times.

The problem of minimum k r-arborescences with bandwidth constraints has important applications in network communications. In group multicast, for example, each message (a plain text, a piece of speech or a frame of video image, etc.) multicast by a presenter must be received by all the participants and every participant can be a presenter. The multicast routing from a single source to a group of destinations is to find a tree rooted at the source and containing all destinations. Routing of a group multicast is to find a

[1] The research by the first author was partially supported by the National Natural Science Foundation of China. The second author's research was fully supported by a CERG grant [Project No. CityU 1047/01E]. The work of the third author was partially supported by a CERG Grant of HKRGC [Project No. CityU1056/01E].
[2] Institute of Systems Science, Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, Beijing 100080, People's Republic of China. caimc@mail.iss.ac.cn.
[3] Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong. {deng, lwang}@cs.cityu.edu.hk.

Received December 20, 2000; revised February 22, 2002. Communicated by Ming-Yang Kao.
Online publication December 5, 2003.

set of routing trees, one for each group number, so that each member can use its own routing tree to multicast messages [2]–[5], [12], [17].

We are interested in efficient algorithms for the problem of minimum k r-arborescences with bandwidth constraints. Several general approaches in combinatorial optimization can be applied to this problem, e.g., the approach of matroid intersection and the ellipsoid method. However, the resulting time complexity would be very high. In this paper we take the approach of the intersection of two polymatroids which requires solutions of several subprocedures, including a maximum flow algorithm for independence testing of subsets for the related polymatroids.

We first present the combinatorial algorithm using polymatroid intersection for the problem of minimum k r-arborescences with bandwidth constraints in Section 2. In Section 3 we prove the validity of the algorithm. In Section 4 we present an algorithm for constructing the required arborescences. Finally, we conclude our paper with some discussions.

2. Algorithms.

In this section we give a combinatorial algorithm for the problem. First, we give some definitions. For any $B \subseteq A$ and $S \subseteq V$, let $V(B)$ denote the set of end-vertices of B, let $E(S)$ be the set of arcs with both end-vertices in S, let $\Delta^-(S)$ be the set of arcs in A entering S, let $\Delta^+(S) = \Delta^-(\bar{S})$, let $\bar{S} = V \backslash S$ and let $V \backslash v = V \backslash \{v\}$ for $v \in V$. For disjoint subsets $S, S' \subset V$ we write $E(S, S')$ for the set of arcs from S to S'.

For a function f defined on set S and $T \subseteq S$, we write $f(T) = \sum_{t \in T} f(t)$, and set $f(\emptyset) = 0$. For a family $\mathcal{F}$ of specified subsets of A and a weight function w on A, $F \in \mathcal{F}$ is said to be w-minimal in $\mathcal{F}$ if $w(F) \leq w(X)$ for all $X \in \mathcal{F}$.

If we construct a new digraph $G^* = (V, A^*)$ from $G = (V, A)$ by replacing each arc $a \in A$ by $b(a)$ parallel arcs with weight $w(a)$ each, then the k required r-arborescences correspond to k arc-disjoint r-arborescences with minimum total weight in G^*, and they are also a minimum weight $k(n-1)$-arc intersection of two matroids, one is the union of k graphic matroids of G^* considered as undirected and the other is the partition matroid $(A^*, \mathcal{I})$ where $\mathcal{I} = \{I \subseteq A^* : |I \cap D^-_{G^*}(v)| \leq k, \forall v \in V \backslash r\}$. Hence the problem can be solved by matroid intersection algorithms, see, for instance, [10] and [13]. However, this approach has two drawbacks: the number of the arcs is largely increased by replacing every arc $a \in A$ by $b(a)$ parallel arcs and it repeatedly calls a matroid partitioning algorithm of complexity $O(k^3 n^4)$ [6], [13] as a subroutine for independence testing in the matroid union. The complexity of the matroid intersection approach for this problem is $O(k^4 (b(A))^2 n^5)$ [10].

To avoid these drawbacks, we consider the problem as an optimal intersection of two weighted polymatroids on A. The first one is

$$P_1 = \{x \in Q^A : x \geq 0, x(B) \leq k\rho(B), \forall B \subseteq A\},$$

where ρ is the rank function of the graphic matroid of G. It is easily shown that

$$P_1 = \{x \in Q^A : x \geq 0, x(E(S)) \leq k(|S| - 1), \forall \emptyset \neq S \subseteq V\}.$$

The second polymatroid is

$$P_2 = \{x \in Q^A : x \geq 0,\ x(\Delta^-(v)) \leq k, \forall v \in V \backslash r,\ x(a) \leq b(a), \forall a \in A\}.$$

Suppose that G has k required r-arborescences, let $x(a)$ denote the times of $a \in A$ covered by the r-arborescences, then $x \in P_1 \cap P_2$ has minimum total weight under condition $x(A) = k(n-1)$.

Conversely, suppose the linear program

$$\min\{wx : x \in P_1 \cap P_2, x(A) = k(n-1)\} \tag{1}$$

has an optimal solution, then it has an integral optimal solution x by the extended polymatroid intersection theorem [16]. Note that $k(n-1) = x(A) = \sum\{x(\Delta^-(v)) : v \in V\backslash r\} \le k(n-1)$, we have $x(\Delta^-(v)) = k$ for all $v \in V\backslash r$. Thus for any nonempty $S \subseteq V\backslash r$, $x(\Delta^-(S)) = k|S| - x(E(S)) \ge k$. Based on Edmonds' arc-disjoint arborescence theorem [7], G has k r-arborescences covering each $a \in A$ at most $x(a)$ times and hence exact $x(a)$ times because $x(A) = k(n-1)$.

To solve the problem, we first find an optimal solution to (1) by a weighted polymatroid intersection algorithm, which is an adaptation of Frank's weighted matroid intersection algorithm [10], and then decompose the intersection into k required r-arborescences by an adaptation of Lovász' arborescence packing algorithm [14].

The polymatroid intersection algorithm starts with $x \equiv 0$, $w_1 \equiv 0$ and $w_2 \equiv w$. In a general step, suppose we have $x \in P_1 \cap P_2$, w_1 and w_2 such that $w_1 + w_2 = w$, $x(A) < k(n-1)$ and x is w_i-minimal in P_i with $x(A)$ being fixed (i.e., $w_i x \le w_i x'$ for all $x' \in P_i$ with $x'(A) = x(A))(i = 1, 2)$. We construct new $x' \in P_1 \cap P_2$, w_1' and w_2' such that $w_1' + w_2' = w$, $x'(A) \ge x(A) + 1$ and x' is w_i'-minimal in P_i with $x'(A)$ being fixed.

Given an $x \in P_1 \cap P_2$, $S \subseteq V$ is called *P_1-tight* with respect to x if $x(E(S)) = k(|S| - 1)$. Clearly, $\emptyset$ cannot be P_1-tight. We say $S \subseteq V$ contains arc $a \in A$, denoted by $a \in S$, if $a \in E(S)$.

LEMMA 1. *If S and S' are intersecting P_1-tight subsets (with respect to x), then*

(1) *$S \cap S'$ and $S \cup S'$ are also P_1-tight and*
(2) $x(E(S\backslash S', S'\backslash S)) + x(E(S'\backslash S, S\backslash S')) = 0.$

PROOF. Indeed, write μ for the left-hand side of the last equality, then we have

$$\begin{aligned} x(E(S \cap S')) &+ x(E(S \cup S')) \\ &= x(E(S)) + x(E(S')) + \mu \\ &= k(|S| + |S'| - 2) + \mu \\ &= k(|S \cap S'| - 1) + k(|S \cup S'| - 1) + \mu \\ &\ge x(E(S \cap S')) + x(E(S \cup S')) + \mu, \end{aligned}$$

implying the assertions. □

Let $e_a \in Q^A$ denote the unit vector with 1 only in a's component. Suppose an integral $x \in P_1 \cap P_2$ and an $a \in A$ with $x(a) < b(a)$. If $x + e_a \notin P_1$, then by the definition of P_1 there exists a P_1-tight $S \subseteq V$ containing a. We denote by $S_1(x, a)$ the intersection of all P_1-tight subsets containing a. Then $S_1(x, a)$ is the unique minimal P_1-tight subset (under inclusion) containing a. Obviously, $S_1(x, a') \subseteq S_1(x, a)$ if $S_1(x, a)$ contains $a' \in A$ and $x(a') < b(a')$.

For $a = uv \in A$ with $x(a) < b(a)$, if $x + e_a \notin P_2$, clearly, $x(\Delta^-(v)) = k$ by the definition of P_2, put $S_2(x, a) = \Delta(v)$.

Write $C_i(x, a) = \{a' \in S_i(x, a) : x(a') > 0\},\ i = 1,\ 2.$

Algorithm A

```
Step 0. For all $a \in A$, set $x(a) = 0$, $w_1(a) = 0$ and $w_2(a) = w(a)$.
Step 1. Let $m_i = \min\{w_i(e) : x(e) < b(e), x + e_a \in P_i\}$ $(i = 1, 2)$,
          $A_i = \{a \in A : x(a) < b(a), x + e_a \in P_i, w_i(a) = m_i\}$ $(i = 1, 2)$.
        Label each $a = uv \in A_2$ with $(\emptyset, \pi)$ where $\pi = \min\{b(a) - x(a), k - x(\Delta^-(v))\}$.
        Set $L^+ = A_2$ and $L^- = \emptyset$.
Step 2. Set $L = L^+ \cup L^-$.
        If some $a \in A_1$ is labeled, go to Step 3. If there are no
          unscanned labeled arcs, go to Step 4, else by the first-labeling-
          first-scanning method we label arcs as follows.
        Suppose $a \in L$ is the first labeled but unscanned arc and has label
          $(a', \pi)$.
        If $a \in L^+$, give each $a'' \in C_1(x, a) \backslash L$ with $w_1(a'') = w_1(a)$ label
          $(a, -\pi'')$ where $\pi'' = \min\{\pi, x(a'')\}$ and put $a''$ into $L^-$.
        If $a \in L^-$, give each $a'' \in A \backslash L$ such that $a \in C_2(x, a'')$ and
          $w_2(a'') = w_2(a)$ label $(a, \pi'')$ where $\pi'' = \min\{-\pi, b(a'') - x(a'')\}$ and put $a''$ into $L^+$.
        Go to Step 2.
Step 3. Augment $x$. Using backtracking find an augmenting arc sequence
          $Q = a_1 a_2 \cdots a_{2\ell+1}$ where $a_1 \in A_2$ and $a_{2\ell+1} \in A_1$ (notice that
          no arcs $a \in A_1$ can get a label from $L^+$ since no $P_1$-tight subsets
          contain $a$, hence $a_{2\ell+1} \in L^+$). Suppose $a_{2\ell+1}$ has label $(a_{2\ell}, \pi)$.
          Find $\sigma(x, a_{2\ell+1})$ by Subroutine A with $a_{2\ell+1}$ and $x$.
        Set $\theta = \min\{\pi, \sigma(x, a_{2\ell+1})\}$ and modify $x$ along $Q$ as follows:

          $$x'(a) = \begin{cases} x(a) + \theta & \text{if} \quad a = a_{2i+1}, \\ x(a) - \theta & \text{if} \quad a = a_{2i}, \\ x(a) & \text{otherwise.} \end{cases}$$

        If $x'(A) < k(n - 1)$, set $w'_i = w_i$ $(i = 1, 2)$ and remove all labels,
          go to Step 1, else then $x'$ is a required intersection, stop.
Step 4. Modify weights $w_1$ and $w_2$. Set
        $\theta = \min\{\theta_1, \theta_2, \theta_3, \theta_4\}$ where
        $\theta_1 = \min\{w_1(a) - w_1(a') : a \in L^+, x + e_a \notin P_1,$
                $a' \in C_1(x, a) \backslash L\}$,
        $\theta_2 = \min\{w_1(a) - m_1 : a \in L^+, x + e_a \in P_1\}$,
        $\theta_3 = \min\{w_2(a) - w_2(a') : a \notin L, x(a) < b(a), x + e_a \notin P_2,$
                $a' \in C_2(x, a) \cap L^-\}$,
```

$\theta_4 = \min\{w_2(a) - m_2 : a \notin L, x(a) < b(a), x + e_a \in P_2\}$.
(The minimum is $+\infty$ when it is taken over the empty set.)
If $\theta = +\infty$, then the problem is infeasible, that is, there exists such a nonempty subset $S \subseteq V \backslash r$ that $b(\Delta^-(S)) < k$, stop. Otherwise revise

$$w_1'(a) = \begin{cases} w_1(a) & \text{if } \quad a \in L, \\ x(a) + \theta & \text{otherwise,} \end{cases}$$

and $w_2' = w - w_1'$. Go to Step 1.

Subroutine A

Step 1. For given arc $a \in A$ and $x \in P_1 \cap P_2$, construct a network N from $G = (V, A)$ as follows. For every $v \in V \backslash r$ with $x(\Delta^-(v)) < k$, add a new arc rv with capacity $k - x(\Delta^-(v))$, and let x be the capacities of arcs in A.

Step 2. By a maximal flow algorithm find an r-cut of minimum capacity separating r from $V(a)$ in N [1], [8], whose value is denoted by $\sigma(x, a)$.

The complexity of the subroutine is bounded by $O(kn^2)$ (see Theorem 7.22 of [1]). (Note that all the arcs concerned in this subroutine, except a, are in $\{a' \in A : x(a') > 0\}$, hence the number of such arcs is less than kn.)

3. Proof for Algorithm A.

In this section we prove the validity of Algorithm A. We need several lemmas.

LEMMA 2. *$x \in P_i$ is w_i-minimal with $x(A)$ being fixed if and only if for any $a \in A$ with $x(a) < b(a)$,*

(1) *$x + e_a \notin P_i$ implies $w_i(a) \geq w_i(a')$ for every $a' \in C_i(x, a)$ and*
(2) *$x + e_a \in P_i$ implies $w_i(a) \geq w_i(a')$ for every $a' \in A$ with $x(a') > 0$.*

LEMMA 3. *Let $x \in P_1 \cap P_1$ and x is w_i-minimal with $x(A)$ being fixed where $w_1 + w_2 = w$.*

(1) *x is also w_i'-minimal with $x(A)$ being fixed after modifying weights in Step* 4.
(2) *x' is w_i-minimal with $x'(A)$ being fixed after augmenting x in Step* 3. *Furthermore, x' is also w-minimal in $P_1 \cap P_2$ with $x'(A)$ being fixed.*

The proofs of Lemmas 2 and 3 are very similar to those given in [10] and are omitted.

LEMMA 4. *There is an algorithm of complexity $O(kn^2)$ to decide, for an $x \in P_1 \cap P_2$ and an $a \in A$ with $x(a) < b(a)$, whether there exists a P_1-tight subset S containing a or not, and in the former case, determine $S_1(x, a)$.*

PROOF. Let S be a P_1-tight subset, we examine two cases according to whether $r \in S$ or not.

Case 1: $r \in S$. Then S is P_1-tight if and only if $x(\Delta^-(S)) = 0$ and $x(\Delta^-(v)) = k$ for all $v \in S\backslash r$ by noting that $x(E(S)) = k(|S| - 1)$, $\Delta^-(r) = \emptyset$ and $x(\Delta^-(v)) \leq k$ for all $v \in S\backslash r$.

Given $a \in A$ with $x(a) < b(a)$, it is easy to detect such a P_1-tight S containing a. Indeed, let S^* be the set of vertices $v' \in V$ from which there is path to $V(a)$ using only such arcs a' that $x(a') > 0$. If $r \notin S^*$ or $x(\Delta^-(v')) < k$ for some $v' \in S^*\backslash r$, then there are no P_1-tight subsets containing r and a; otherwise S^* is a P_1-tight subset containing r and a. The running time of such a detection is bounded by kn.

Furthermore, if S is a P_1-tight subset containing r and an arc, we contract S into a new source r' and fix x on the arcs $uv \in A$ with $v \in S$. To see this, we show that no P_1-tight subsets S containing r contain labeled arcs. Assuming it was not true, let S be a maximal one among such tight subsets and let $a = uv \in L$ be the first labeled arc in S and have label (a', π). Obviously, $a \notin L^-$ for otherwise the P_1-tight subset $S_{a'}$ contains a' and a. Thus $S \cup S'$ is P_1-tight, $t \in S \cup S' \supset S$ as $a' \notin S$, contradicting the choice of S. Hence, $a \in L^+$. However, $a \notin A_2$ otherwise S contains v with $x(\Delta^-(v)) < b(a)$. Then $a \in L^+\backslash A_2$, $a' \in L^- \cap \Delta^-(v) \cap \Delta^-(S)$, implying $x(\Delta^-(S)) > 0$, a contradiction.

Case 2: $r \notin S$. Given $a \in A$, find $\sigma(x, a)$ and a maximal flow f by Subroutine A. If $\sigma(x, a) > k$, there are no P_1-tight subsets containing r and a. Otherwise, let S be the set of vertices $v' \in V$ from which there is path to $V(a)$ using only such arcs a' that $x(a') > f(a) \geq 0$. The running time of such a detection is bounded by $O(kn^2)$.

For an $x \in P_1 \cap P_2$ and an $a \in A$ with $x(a) < b(a)$, we first detect whether there exists a P_1-tight subset containing r and a; if not, then detect whether there exists a P_1-tight subset containing a or not, in the former case, determine $S_1(x, a)$. The running time is also bounded by $O(kn^2)$. □

LEMMA 5. *The problem is infeasible if* $\theta = +\infty$ *in Step* 4 *of Algorithm A*.

PROOF. To prove the lemma, it suffices to show there is a $S \subseteq V\backslash r$ such that $S \neq \emptyset$ and $b(\Delta^-(S)) < k$. Put $U = \{v \in V\backslash r \colon x(\Delta^-(v)) < k\}$. Then $U \neq \emptyset$ since $x(A) < k(n-1)$ at that time. We may assume $L \neq \emptyset$ otherwise $b(\Delta^-(v)) = x(\Delta^-(v)) < k$ for all $v \in U$.

First we claim that each arc $a \in L$ is contained in some $S_1(x, a')$. Indeed, if $a \in L^+$, then $x + e_a \notin P_1$ as $\theta_2 = +\infty$, thus $a \in S_1(x, a)$ by the definition of $S_1(x, a)$. If $a \in L^-$, say a has label (a', π), then $a \in S_1(x, a')$.

Now consider a hypergraph H with vertex set V and hyperedge set $\{S_1(x, a) : a \in L^+\}$. Then each connected component C of H is P_1-tight by Lemma 1. Moreover, $C \cap U \neq \emptyset$. To see this, suppose $a_1 = uv$ has label $(\emptyset, *)$, a_{i+1} has label $(a_i, *)$ $(i = 1, 2, \ldots, 2j)$, $a_{2i} \in L^-$ and $a_{2i+1} \in L^+$. Then $v \in U \cap S_1(x, a)$, $a_{2i} \in S_1(x, a_{2i-1})$, $a_{2i}, a_{2i+1} \in \Delta^-(v')$ for some $v' \in V\backslash r$, implying $v' \in S_1(x, a_{2i-1}) \cap S_1(x, a_{2i+1})$ and $v \in C$. Furthermore, every $a \in L$ is contained in some connected component of H.

We show $x(\Delta^-(C)) = b(\Delta^-(C))$. Suppose to the contrary, we take any $a = uv \in \Delta^-(C)$ with $x(a) < b(a)$. Clearly, $a \notin L$ as no component of H covers it.

Hence $x(\Delta^-(v)) = k$ otherwise $\theta \leq \theta_4 < \infty$. Notice that $x(E(C)) + x(\Delta^-(C)) = \sum\{x(\Delta^-(v')) : v \in C\} < k|C|$ as $C \cap U \neq \emptyset$, we have $x(\Delta^-(C)) < k$ since $x(E(C)) = k(|C| - 1)$ by Lemma 1, yielding there is some $a' \in \Delta^-(v) \cap E(C)$ with $x(a') > 0$. By Lemma 1, a' is contained in some $S_1(x, a'')$. Then $a' \in L$ follows from $\theta_1 = \infty$. Clearly, $a' \notin L^-$ otherwise $\theta_3 < \infty$. However, if $a' \in L^+$, say a' has label $(a'', *)$, then a can also get label $(a'', *)$ since $\theta_3 = +\infty$, $\{a, a', a''\} \subseteq \Delta^-(v)$ and $x(a) < b(a)$, a contradiction. Therefore $b(\Delta^-(C)) = x(\Delta^-(C)) < k$. □

THEOREM 1. *Algorithm A can solve the problem in $O(k^2m^2n^3)$ time.*

PROOF. The labeling process requires at most m^2 steps to find an augmenting path or the subset T. Each step calls Subroutine A at most once. If the augmenting path is found, $x(A)$ increases by at least one. Otherwise modify weights w_1 and w_2, each modification results in at least one arc being labeled and the current labels can be used again. Thus the complexity of the algorithm can be bounded by $O(k^2m^2n^3)$. □

REMARK 1. We use Subroutine A of complexity $O(kn^2)$ instead of a matroid partitioning algorithm of complexity $O(k^3n^4)$ for independence testing to reduce the algorithm's complexity.

REMARK 2. Algorithm A can be slightly modified to solve the following generalized problem:

Let $G = (V, A)$ be a digraph with source r. A weight function w and upper and lower bound functions $b \geq b'$ (positive integer) on A are given. Find k r-arborescences in G covering each arc a at least $b'(a)$ and at most $b(a)$ times such that each of the k r-arborescences contains exactly one arc leaving r and the total weight is minimized.

4. Algorithm of Decomposition.

Suppose that an optimal integral $x \in P_1 \cap P_2$ with $x(A) = k(n-1)$ is found. Thus $x(\Delta^+(S)) \geq k$, $\forall S \subset V$ with $r \in S$. Now we recursively construct k r-arborescences as follows.

Algorithm B

```
Step 0. Set $F = \{a = rv\}$ with $x(a) > 0$ and $x = x - e_a$.
        Let $F$ be an $r$-subarborescence satisfying

        (2)    $x(\Delta^+(S)) \geq k - 1$    for all  $S \subset V$  with  $r \in S$.

Step 1. If $V(F) \subset V$, go to Step 2, else $F$ is a required arborescence.
Step 2. Choose an arc $a \in \Delta^+(V(F))$ such that $x(a) \geq 1$ and $x - e_a$ satisfies
          (2), set $F = F \cup \{a\}$ and $x = x - e_a$, if $x(a) = 0$, remove $a$. Go
          to Step 1.
```

THEOREM 2. *Algorithm B is correct and runs in $O((kn)^3)$ time.*

PROOF. We first show the existence of the chosen arc in Step 2. Suppose it was not true, then for each arc $a \in \Delta^+(V(F))$ with $x(a) \geq 1$ there exists $S_a \subset V$ such that $r \in S_a$, $a \in \Delta^+(S_a)$ and $x(\Delta^+(S_a)) = k-1$. Put $S^* = \bigcup_{a\in\Delta^+(V(F))} S_a$. Then by (2) and induction on $|\Delta^+(V(F))|$, it is easily shown that $x(\Delta^+(S^*)) = k - 1$, implying $S^* \subset V$. Clearly, $r \in S^*$, $\{a \in \Delta^+(S \cup S^*) : x(a) \geq 1\} \subseteq \{a \in \Delta^+(S^*) : x(a) \geq 1\}$ since $v \in S^*$ for each $a = uv \in \Delta^+(V(F))$ with $x(a) \geq 1$. Hence $x(\Delta^+(V(F) \cup S^*)) \leq x(\Delta^+(S^*)) = k-1$. On the other hand, as $V(F) \subseteq V(F) \cup S^*$, we have $(x(\Delta^+(V(F) \cup S^*)) \geq k$, a contradiction.

In order to determine the arc being chosen in Step 2, we check the arcs $a \in \Delta^+(V(F))$ with $x(a) > 0$ one by one by solving $n - 1$ maximal flow problems from r to all other vertices with arc capacity $x - e_a$. Choose a if the minimum value of $n - 1$ maximal flows is great than or equal to $k - 1$.

Note that during the construction time of an r-arborescence, each arc in $\{a \in A : x(a) > 0\}$ is checked at most once since, when it is checked, either it is put into F or it cannot become an element of F anymore. Hence the complexity of Algorithm B is bounded by $O((kn)^3)$ [14]. □

5. Discussions. The problem can also be solved using the ellipsoid method [11]. However, the time complexity is high and it is not practical for the application area that motivated our study [2]–[5], [12] for which b is bounded by a constant because of the technology constraints. On the other hand, the ellipsoid method results in a polynomial time algorithm when the size of b is taken into consideration of the input size. This is of independent interest theoretically. Therefore, we include a sketch of the proof here.

THEOREM 3. *The k arborescence problem is polynomially solvable by the ellipsoid method* [11].

PROOF. Indeed, based on Edmonds' arborescence theorem [7] and Frank's kernel theorem [9], the k arborescence problem has the following linear programming formulation:

$$\begin{array}{lll} & \min \sum_{a\in A} w(a)x(a) & \\ (3) & \text{s.t.} & \\ & x(\delta^-(S)) \geq k, & \forall \emptyset \neq S \subseteq V\backslash r, \\ & 0 \leq x(a) \leq b(a), & \forall a \in A. \end{array}$$

Note that the number of constraints is exponential in n. However, the problem can be solved in polynomial time by the ellipsoid method since its strong separation problem can be done in polynomial time with a maximal flow algorithm. To see this, let x be a vector in Q^A. We first test whether $0 \leq x(a) \leq b(a)$ for all $a \in A$. If one of these inequalities is violated, we have a separating hyperplane. We further test whether $x(\delta^-(S)) \geq k$ for all $\emptyset \neq S \subseteq V\backslash r$. The latter test can proceed as follows. Considering x as a capacity function on A, for each $v \in V\backslash r$ we determine an r-cut $\Delta(S_v)$ of minimum capacity separating r from v by a maximal flow algorithm and determine whether $x(\Delta(S_v)) \geq k$ or not. If $x(\Delta(S_v)) \geq k$ for all $v \in V\backslash r$, then x is an optimal solution. Otherwise some inequality $x(\Delta(S_v)) \geq k$ is violated and a separating hyperplane is found. □

A more general problem arising in network communications is the following *k directed Steiner trees with bandwidth constraints* problem: Let $G = (V, A)$ be a digraph. Each arc $a \in A$ has a bandwidth $b(a)$ indicating the number of times such an arc can be used. We are given a group of processes (a set of vertices) $D \subseteq V$ and a set of sources $S = \{s_1, s_2, \ldots, s_k\} \subseteq V$, the problem here is to find k Steiner trees rooted at the k sources in S and containing every node in D such that the total number of times (bandwidth) used for each arc a is at most $b(a)$. This problem was shown to be NP-hard and the weighted version cannot be approximated within any ratio polynomial to the input size [15].

References

[1] R.K. Ahuja, T.L. MaGnanit and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[2] M.H. Ammar, S.Y. Cheung, and C.M. Scoglio, Routing multipoint connections using virtual paths in an ATM network, *Proc. IEEE INFOCOM '93*, pp. 98–105, 1993.

[3] F. Bauer and A. Varma, Distributed algorithms for multicast path setup in data networks, *IEEE/ACM Trans. Networking*, **4** (1996), 181–191.

[4] K. Bharath-Kumar and J.M. Jaffe, Routing to multiple destinations in computer networks, *IEEE Trans. Commun.* **COM-31** (1983), 343–351.

[5] S.E. Deering and D.R. Cheriton, Multicast routing in datagram internetworks and extended LANs, *ACM Trans. Comput. Systems*, **8** (1990), 85–110.

[6] J. Edmonds, Minimum partition of a matroid into independent subsets, *J. Res. Nat. Bur. Standards*, **69B** (1965), 67–72.

[7] J. Edmonds, Edge-disjoint branchings, in: *Combinatorial Algorithm* (R. Rustin, ed.), Academic Press, New York, 1973, pp. 91–96.

[8] L.R. Ford and D.R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, NJ, 1962.

[9] A. Frank, Kernel systems of digraphs, *Acta Sci. Math. (Szeged)*, **41** (1979), 63–76.

[10] A. Frank, A weighted matroid intersection algorithm, *J. Algorithms*, **2** (1981), 328–336.

[11] M. Grötschel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin, 1988.

[12] X. Jia and L. Wang, A group multicast routing algorithm by using multiple minimum Steiner trees, *Comput. Commun.*, **20** (1997), 750–758.

[13] E.L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart, & Winston, New York, 1975.

[14] L. Lovász, On two minimax theorems in graph theory, *J. Combin. Theory Ser. B*, **21** (1976), 96–103.

[15] B. Ma and L. Wang, On the inapproximability of disjoint paths and minimum Steiner forest with bandwidth constraints, *J. Comput. System Sci.*, **60** (2000), 1–12.

[16] C. McDiarmid, Blocking, Antiblocking and pairs of matroids and polymatroids, *J. Combin. Theory Ser. B*, **25** (1978), 313–325.

[17] J. Moy, Multicast routing extensions for OSPF, *Comm. ACM*, **37** (1994), 61–66.