Transversals of Subtree Hypergraphs and the Source Location Problem in Digraphs [*]

Jan van den Heuvel [†] and Matthew Johnson [†]

CDAM Research Report LSE-CDAM-2004-10

May 2004

Abstract

A hypergraph $H = (V, E)$ is a subtree hypergraph if there is a tree T on V such that each hyperedge of E induces a subtree of T. To find a minimum size transversal for a subtree hypergraph is, in general, NP-hard. In this paper, we show that if it is possible to decide if a set of vertices $W \subseteq V$ is a transversal in time $S(n)$ (where $n = |V|$), then it is possible to find a minimum size transversal in $O(n^3\, S(n))$.

This result provides a polynomial algorithm for the Source Location Problem : a set of (k, l)-sources for a digraph $D = (V, A)$ is a subset K of V such that for any $v \in V \setminus K$ there are k arc-disjoint paths that each join a vertex of K to v and l arc-disjoint paths that each join v to K. The Source Location Problem is to find a minimum size set of (k, l)-sources. We show that this is a case of finding a transversal of a subtree hypergraph, and that in this case $S(n)$ is polynomial.

1 Introduction

A hypergraph $H = (V, E)$ is a *subtree hypergraph* if there exists a tree T on V such that each hyperedge of E induces a subtree of T. (Subtree hypergraphs are sometimes called *arboreal hypergraphs* or simply *hypertrees*; note that the latter term appears to have several unrelated definitions, and hence we will avoid using it.) A *transversal* of a hypergraph H is a set of vertices that contains at least one vertex of each hyperedge. It is, in general, NP-hard to find a minimum size transversal for a subtree hypergraph. The main result of this paper is that a minimum transversal can be found in polynomial time whenever there is an oracle that can decide in polynomial time if a set is a transversal or not.

Theorem 1 *Let $H = (V, E)$ be a subtree hypergraph and let $n = |V|$. If it is possible to check whether or not a subset $S \subseteq V$ is a transversal in time $S(n)$, then it is possible to find a minimum size transversal in time $O(n^3\, S(n))$.*

[*] Research supported by EPSRC MathFIT grant no. GR/R83514/01.

[†] Centre for Discrete and Applicable Mathematics, Department of Mathematics, London School of Economics, Houghton Street, London WC2A 2AE, U.K.; email : {jan, matthew}@maths.lse.ac.uk.

We demonstrate the value of Theorem 1 by presenting a problem where an oracle does exist.

The problem of selecting the location for facilities, subject to some constraints, in a given network is called a *location problem* [11]. For example, in a multimedia network, services for users must be located at specially selected nodes called sources. A user at a node u requests services from a source v by communicating along a path from u to v. Then the services are provided along a path from v to u. Thus, if the network is represented by a digraph $D = (V, A)$, then a measure of the robustness of the network is the number of arc-disjoint paths between the sources and each other vertex.

Definition 1 *A* set of (k, l)-sources *for D is a subset S of V such that for any $v \in V \setminus S$ there are k arc-disjoint paths that join S to v, and l arc-disjoint paths that join v to S.*

For disjoint sets of vertices U and W, let $\lambda(U, W)$ denote the maximum number of arc-disjoint paths joining U to W.

Source Location Problem

Input:	A digraph $D = (V, A)$ and nonnegative integers k and l.
Output:	$S \subseteq V$ of minimum size.
Constraint:	$\lambda(S, v) \geq k$ and $\lambda(v, S) \geq l$ for all $v \in V \setminus S$.

(For other examples of location problems with connectivity requirements see [1, 6, 7, 8, 9, 10, 12, 13]). Ito *et al.* [8] found an algorithm for the Source Location Problem that was polynomial for fixed k and l. They left open the problem of finding an algorithm that is polynomial when k and l are not fixed. The problem (with $l = 0$) was also posed by Tibor Jordán during the problem session of the *Graph Theory 2003* Conference, held in Nyborg, Denmark, 27–30 November 2003. It is easy to see (see also the next section) that the problem is equivalent to finding the minimum number of vertices required to cover all sets $P \subseteq V$ for which there are fewer than k arcs joining $V \setminus P$ to P or fewer than l arcs joining P to $V \setminus P$, a simple generalization of Problem 15 in [14].

In the next section we show that Theorem 1 provides a polynomial algorithm for the Source Location Problem even when k and l are not fixed. In the sections after that we present and analyse algorithms to prove Theorem 1.

2 The Source Location Problem

In this section we use results on the Source Location Problem from Ito *et al.* [8] to present the problem in terms of subtree hypergraphs. Throughout, let $D = (V, A)$ be a digraph and let k and l be nonnegative integers.

For $P \subseteq V$, $d^-_D(P)$ is the number of arcs joining $V \setminus P$ to P, and $d^+_D(P)$ is the number of arcs joining P to $V \setminus P$.

Definition 2 *A non-empty set $P \subseteq V$ is (k, l)-*critical *if $d^-_D(P) < k$ or $d^+_D(P) < l$, and no proper subset of P has this property.*

It is clear that S is a set of sources if and only if it intersects each (k, l)-critical set. Let the *critical hypergraph* of D be $H_D = (V, E)$ where the hyperedges E are the (k, l)-critical sets of D. A *transversal* of H_D is a set of vertices that contains at least one vertex of each hyperedge. So the Source Location Problem is the problem to to find a transversal of H_D of minimum size.

An undirected graph is *chordal* if it contains no induced cycles of length more than three (that is, if every cycle of length at least four has a *chord*, an edge joining two vertices that are not adjacent in the cycle). A collection of sets has the *Helly property* if any subcollection of pairwise intersecting sets has non-empty intersection.

Proposition 2 (Ito *et al.* [8]) *For any digraph D, the line graph of the critical hypergraph H_D is chordal and the hyperedges have the Helly property.*

Note that Proposition 2 implies that any induced subgraph of H_D also has a chordal line graph and has the Helly property.

We will use the following property of chordal graphs (see, for example, [3])

- for any chordal graph G, there exists a *simplicial vertex* $v \in V(G)$ such that v and its neighbours form a clique.

Therefore, from this property and the Helly property,

- for any collection $\mathcal{P}$ of hyperedges in the critical hypergraph of a digraph, there exists a simplicial hyperedge $P \in \mathcal{P}$ such that P and all hyperedges in $\mathcal{P}$ that intersect P have non-empty intersection.

Using this property, Theorem 3 follows easily from Proposition 2.

Theorem 3 (Ito *et al.* [8]) *A minimum size set of (k, l)-sources for a digraph D is the same size as a maximum size family of disjoint (k, l)-critical sets.*

Next we see how the Source Location Problem is solved using Theorem 1 : the following characterization of subtree hypergraphs was found independently by several authors; see [4, Theorem 3.8].

Proposition 4 *A hypergraph is a subtree hypergraph if and only if its line graph is chordal and its hyperedges have the Helly property.*

Thus, by Proposition 2, a critical hypergraph $H_D = (V, E)$ of a digraph $D = (V, A)$ is a subtree hypergraph. To check that a subset W of V is a transversal of H_D means checking that the vertices are a set of (k, l)-sources for D. Recalling Definition 1, we see this is easily done by contracting the sources into a vertex s and checking that $\lambda(s, v) \geq k$ and $\lambda(v, s) \geq l$ for all $v \in V$. This can be done using an algorithm of Hao and Orlin [5] in time $O(n\, m \log(n^2/m))$, where m is the number of arcs. Thus, by Theorem 1, a minimum size set of (k, l)-sources for a digraph can be found in polynomial time.

We remark that Bárász, Becker and Frank [2] have also recently solved the Source Location Problem using a completely different approach and giving different algorithms.

3 Transversers in Hypergraphs

In the next section we present algorithms that find a minimum transversal to prove Theorem 1. First we need several definitions and preliminary results. The algorithm will work with a generalization of transversals.

Definition 3 *A disjoint collection of non-empty sets* $\mathcal{T} = \{X_1, \ldots, X_t\}$ *is a* set of transversers *for a hypergraph* H *if* $X_i \subseteq V$ *for* $1 \leq i \leq t$*, and* $\bigcup_{i=1}^{t} X_i$ *is a transversal for* H*.*

A transverser $X \in \mathcal{T}$ is *redundant* if $\mathcal{T} \setminus \{X\}$ is also a set of transversers. A set of transversers containing no redundant set is *minimal.*

Definition 4 *For* $u \in V$ *and* $X \in \mathcal{T}$*, if* $(\mathcal{T} \setminus \{X\}) \cup \{\{u\}\}$ *is a set of transversers, then* u *is an* alternative *to* X*. The* unrestricted set of alternatives *to* X *contains all such* u *and is denoted* $A(X)$*. The* restricted set of alternatives *to* X *is* $A(X) \cap X$ *and is denoted* $B(X)$*.*

In the algorithms, a common operation is to alter a set of transversers by replacing one of the transversers X by its (restricted or unrestricted) set of alternatives X'. Notice that if $X' \neq \varnothing$, then $(\mathcal{T} \setminus \{X\}) \cup \{X'\}$ is also a set of transversers.

Definition 5 *For* $X \in \mathcal{T}$*, a hyperedge is an* essential hyperedge *of* X *if it is covered by* X*, but not by any other transverser.*

A vertex is an alternative to X if and only if it covers its essential hyperedges. Thus if X is not redundant, $A(X)$ is equal to the intersection of the essential hyperedges of X; if X is redundant, it has no essential hyperedges and $A(X) = V$.

Definition 6 *If a set* $X \subseteq V$ *is a subset of a hyperedge* P*, then* P *is a* confining hyperedge *of* X*. If* X *is equal to the intersection of its confining hyperedges, then it is* confined*.*

If a set $X \in \mathcal{T}$ is not redundant, then $A(X)$ is confined by the essential hyperedges of X. If X is confined, then $B(X)$ is confined by the confining hyperedges of X and the essential hyperedges of X.

Definition 7 *A set of transversers* $\mathcal{T}$ *is* stable *if for each* $X \in \mathcal{T}$*,* $X = A(X)$*. It is* consistent *if for each* $X \in \mathcal{T}$*,* $X = B(X)$*.*

Notice that if $\mathcal{T}$ is stable, then it is also consistent and each $X \in \mathcal{T}$ is confined.

Proposition 5 *If* H *has a minimum size transversal* T *and a stable set of transversers* $\mathcal{T}$*, then* $|T| \geq |\mathcal{T}|$*.*

In the next section, we will see that from a stable set, it is possible to find a transversal that contains one vertex from each transverser. This will prove that $|T| \leq |\mathcal{T}|$. Thus $|T| = |\mathcal{T}|$.

Proof of Proposition 5: Let $\mathcal{T} = \{X_1, \dots, X_t\}$. We will prove that $|T| \geq |\mathcal{T}|$ by finding a disjoint collection of hyperedges $P_1, \dots, P_t$.

As each transverser in $\mathcal{T}$ is its own set of alternatives, it is equal to the intersection of its essential hyperedges, and each of these essential hyperedges intersects only one transverser. Let H_1 be the subhypergraph of H containing all the essential hyperedges. Let P_1 be a simplicial hyperedge of H_1, and suppose that P_1 is an essential hyperedge for X_1. Thus $X_1 = A(X_1) \subseteq P_1$, and the essential hyperedges of X_1 are P_1 and none, some or all of its neighbours in the line graph of H_1. Recall from Proposition 2 that the intersection of P_1 with all these neighbours is non-empty. Each vertex in this intersection is certainly an alternative to X_1 — it covers all the essential hyperedges — and thus a member of X_1. Therefore, as the essential hyperedges each intersect only one of the transversers, P_1 and *all* of its neighbours are essential hyperedges of X_1.

Now consider the hypergraph H_2 of the essential hyperedges except those of X_1. Let P_2 be a simplicial hyperedge and suppose it is an essential hyperedge for X_2. Note that P_2 does not intersect P_1 as no essential hyperedge that intersects P_1 is included in H_2. By the same argument as before, P_2 and all of its neighbours in the line graph of H_2 are essential hyperedges of X_2.

Then we look for a simplicial hyperedge in the hypergraph containing all essential hyperedges except those of X_1 and X_2. From this and further repetitions we find $P_3, \dots, P_t$. $\square$

4 The Algorithms

We present two polynomial algorithms (see next page): StableTransversers finds a stable set of transversers for a digraph D, and MinimumTransversal takes a stable set and finds a transversal containing a single vertex from each transverser. By Proposition 5, this transversal will have minimum size.

We must prove the efficacy of the two algorithms, which we will do one by one.

To begin, StableTransversers considers the vertex set of V as a set of transversers $\mathcal{T}$. Redundant sets are discarded until $\mathcal{T}$ is minimal.

The main part of the algorithm contains three nested **while** loops labelled L1, L2 and L3. We say that the algorithm *enters* a loop if the *loop condition* is satisfied. For example, the loop condition for L2 is that $\mathcal{T}$ contains a redundant set. Inside the loops $\mathcal{T}$ is altered by replacing transversers by their sets of alternatives or by discarding transversers. We shall show that after each alteration $\mathcal{T}$ is still a set of transversers. Thus if the algorithm does not enter L1, then for each $X \in \mathcal{T}$, $X = A(X)$. Hence $\mathcal{T}$ is a stable set of transversers and we have the required output.

We shall prove the following stronger claims. (Recall that a set of transversers $\mathcal{T}$ is consistent if $X = B(X)$ for each $X \in \mathcal{T}$.)

```
Algorithm StableTransversers
Input: A hypergraph $H = (V, E)$ where $V = \{v_1, \ldots, v_n\}$.
Output: A stable set of transversers $\mathcal{T}$ for $H$.

let $\mathcal{T} = \{\{v_1\}, \ldots, \{v_n\}\}$;
while there exists a redundant set $R \in \mathcal{T}$ do
  let $\mathcal{T} = \mathcal{T} \setminus \{R\}$;
end /* while */
while there exists $Y \in \mathcal{T}$, $Y \neq A(Y)$ do                /* L1 */
  let $\mathcal{T} = (\mathcal{T} \setminus \{Y\}) \cup \{A(Y)\}$;
  while there exists a redundant set $R \in \mathcal{T}$ then           /* L2 */
    let $\mathcal{T} = \mathcal{T} \setminus \{R\}$;
    while there exists $Z \in \mathcal{T}$, $Z \neq B(Z)$ do            /* L3 */
      let $\mathcal{T} = (\mathcal{T} \setminus \{Z\}) \cup \{B(Z)\}$.
    end /* while */
  end /* while */
end /* while */
Output $\mathcal{T}$.
```

```
Algorithm MinimumTransversal
Input: A hypergraph $H = (V, E)$ with a stable set of transversers $\mathcal{T}$.
Output: $T$, a minimum size transversal for $H$.

while there exists $R' \in \mathcal{T}$, $|R'| \geq 2$ do               /* L4 */
  choose $u \in R'$;
  let $\mathcal{T} = (\mathcal{T} \setminus \{R'\}) \cup \{\{u\}\}$;
  while there exists $Z \in \mathcal{T}$, $Z \neq B(Z)$ do              /* L5 */
    let $\mathcal{T} = (\mathcal{T} \setminus \{Z\}) \cup \{B(Z)\}$.
  end /* while */
end /* while */
let $T = \bigcup_{X \in \mathcal{T}} X$.
Output $T$.
```

Claim 6 *(a) Each time the algorithm considers the loop condition for L1, $\mathcal{T}$ is a consistent set of transversers, and each $X \in \mathcal{T}$ is confined or a singleton, but not redundant.*
(b) Each time the algorithm considers the loop condition for L2, $\mathcal{T}$ is a consistent set of transversers, and each $X \in \mathcal{T}$ is confined or a singleton.

We prove the claim by induction. The first time the algorithm considers the loop condition for L1, $\mathcal{T}$ is a set of singletons and minimal. Thus each transverser is non-redundant and its own restricted set of alternatives, and hence $\mathcal{T}$ is consistent.

Suppose that $X = B(X)$ and X is non-redundant for each $X \in \mathcal{T}$ and that L1 is entered. A set Y is replaced by $Y' = A(Y)$ (and $Y \subsetneqq Y'$ since $Y = B(Y) \subseteq A(Y) \neq Y$). Clearly $B(Y') = Y'$, Y' is confined, and $\mathcal{T}$ is still a set of transversers. For each $X \in \mathcal{T}$, $X \neq Y'$, X remains a singleton or confined, and also the set of alternatives to X will, if anything, have grown when Y was replaced by $A(Y)$. Thus for all $X \in \mathcal{T}$, the claim that X is confined or a singleton and $X = B(X)$ still holds when the loop condition for L2 is considered. So if L2 is not entered, none of the transversers is redundant; the algorithm returns to consider the condition for L1 and Claim 6 (a) holds.

Suppose that L2 is entered and a redundant set R is removed from $\mathcal{T}$. If L3 is not entered, then Claim 6 (b) holds.

So suppose that L3 is entered. We call the process encoded by L3 *reduction*: to *reduce* a set of transversers is to arbitrarily choose a transverser X and replace it by $B(X)$, and to repeat this until a consistent set is obtained (note that after each alteration, the restricted sets of alternatives of each transverser must be recalculated). A set of transversers is *reducible* if this process is possible, that is, if each time we replace a transverser X by $B(X)$ we obtain a set of transversers (clearly the process must eventually terminate as the transversers are continually getting smaller). We need the following result.

Proposition 7 *Let $\mathcal{T}$ be a consistent set of transversers such that each $X \in \mathcal{T}$ is confined or a singleton. Let $Z \in \mathcal{T}$ and let $Z' = \varnothing$ or $Z' = \{z\}$ for some $z \in Z$. If $(\mathcal{T} \setminus \{Z\}) \cup \{Z'\}$ is a set of transversers, then it is reducible. Moreover, each X in the set of transversers obtained after the reduction is confined or a singleton.*

The proof is at the end of the paper.

Apply the proposition to the set $\mathcal{T}$ obtained just before R is discarded with $Z = R$ and $Z' = \varnothing$. Thus after the algorithm has finished looping through L3, Claim 6 (b) holds. Then another redundant set may be discarded and the algorithm may begin to loop through L3 again. When no more redundant sets can be found, the algorithm has finished its run through L1 and Claim 6 (a) holds. We have shown that StableTransversers will output a stable set of transversers.

MinimumTransversal contains two nested **while** loops, L4 and L5. If the algorithm does not enter L4, then each transverser contains one vertex and from this we obtain a minimum size transversal

As before, we must show that as $\mathcal{T}$ is altered, it is always a set of transversers.

Claim 8 *Each time the algorithm considers the loop condition for L4, $\mathcal{T}$ is a consistent set of transversers, and each $X \in \mathcal{T}$ is confined or a singleton.*

We use induction to prove the claim. The first time the algorithm considers the loop condition for L4, $\mathcal{T}$ is stable and the claim holds (see the remark after Definition 7). Assume the claim holds as L4 is entered. After replacing a transverser R' by one of its alternatives, $\mathcal{T}$ is still a set of transversers (by Definition 4). If the algorithm does not enter L5, then the claim is true. All that remains to be proved is that once the algorithm finishes going through L5, the resulting $\mathcal{T}$ satisfies the conditions in the claim. Notice that L5 is identical to L3 : so applying Proposition 7 with $Z = R'$ and $Z' = \{u\}$ will guarantee that this is the case. We have shown that MinimumTransversal will output a minimum transversal.

5 Running Time of the Algorithms

The algorithms StableTransversers and MinimumTransversal contain three consecutive steps that must be taken to obtain a minimum size transversal.

1. Find a minimal set of transversers.
2. Find a stable set of transversers (L1).
3. Find a minimum size transversal (L4).

We will show that Step 2 is the bottleneck and determine the overall running time.

Let $S(n)$ be the complexity of an algorithm that checks whether or not a set is a transversal.

The complexity of Step 1 is $O(n\, S(n))$: for each of the n transversers $\{v_i\} \in \mathcal{T}$, we check whether $\mathcal{T} \setminus \{\{v_i\}\}$ is a set of transversers.

Note that each **while** loop in the algorithms has two consecutive steps : checking the loop condition and, if the condition is satisfied, performing the loop content. To find the complexity of a loop, it is necessary to find the complexity of each step and how many times the algorithm may cycle through the loop. In fact, since we can assume that adding or removing elements from a set can be done in constant time, we only have to consider the complexity of checking the loop conditions.

It takes time $O(n\, S(n))$ to check the loop condition of L3 : for each $Z \in \mathcal{T}$ and each vertex $v \in Z$, we must see if v is an alternative to Z, that is, if $(\mathcal{T} \setminus \{Z\}) \cup \{\{v\}\}$ is a set of transversers. The algorithm may pass through L3 at most n times within each run through L2 (as during each pass through L3 at least one vertex is removed from the sets of transversers). Thus L3 has complexity $O(n^2\, S(n))$. It takes time $O(n\, S(n))$ to check the loop condition of L2 : we check whether $\mathcal{T} \setminus \{X\}$ is a set of transversers for each $X \in \mathcal{T}$. The content of L2 is the loop L3 and, within each run through L1, the number of times the algorithm may consider the loop conditions for L2 and L3 is, in total, n (since both loops have an operation that removes vertices). Thus L2 has complexity $O(n^2\, S(n))$. The loop condition of L1 is checked in time $O(n^2\, S(n))$: for each vertex v and each transverser $Y \in \mathcal{T}$, we must see if v is an alternative to Y, that is, if $(\mathcal{T} \setminus \{Y\}) \cup \{\{v\}\}$ is a set of transversers. The content of L1 is loop L2, and the algorithm may enter L1 at most n times (as the number of transversers is reduced each time). Thus the complexity of L1 is $O(n^3\, S(n))$.

The loop condition for L5 has complexity $O(n\, S(n))$. The loop condition for L4 has complexity $O(n)$. Since loops L4 and L5 together are checked at most n times, we find that the third step has complexity $O(n^2\, S(n))$.

Thus the total running time of the algorithms is $O(n^3\, S(n))$.

6 Proof of Proposition 7

We require one further result on chordal graphs before we prove Proposition 7.

Lemma 9 *Let u and v be non-adjacent vertices in a chordal graph G, and suppose that W_1, the set of vertices adjacent to both u and v, is non-empty. Let W_2 contain each vertex (other than u and v) that is adjacent to every vertex in W_1. Then u and v are in different components in $G - (W_1 \cup W_2)$.*

Proof : We show that if there is a u-v path in $J = G - (W_1 \cup W_2)$, then there is an induced cycle of length greater than 3 in G.

Let $up_1 \cdots p_r v$ be the shortest u-v path in J ($r \geq 2$ as $p_1 \notin W_1$). As $p_1 \notin W_2$, there exists $w \in W_1$ such that p_1 and w are not adjacent in G. If possible, choose the smallest $i \geq 2$ such that p_i is adjacent to w in G : then $\{u, p_1, \ldots, p_i, w\}$ induces a cycle in G. If no p_i is adjacent to w, then $\{u, p_1, \ldots, p_r, v, w\}$ induces a cycle. □

Proof of Proposition 7 : Let $\mathcal{T}$, Z and Z' be as in the proposition. During the reduction of $(\mathcal{T} \setminus \{Z\}) \cup \{Z'\}$ we repeatedly select a transverser X to replace with its restricted set of alternatives $B(X)$. A singleton will never be selected as it is equal to its restricted set of alternatives. If X is confined, then $B(X)$ is also confined, and if $B(X) \neq \varnothing$, then replacing X by $B(X)$ will give a new set of transversers. Hence the proposition holds if $B(X) \neq \varnothing$ for all transversers X encountered during reduction. Thus we will assume that we find a transverser with an empty restricted set of alternatives and show that this leads to a contradiction.

First some terminology : if a set X' is obtained from X by any number of replacements, then we say that X is an *ancestor* of X' and X' is a *descendant* of X (a set is its own ancestor and descendant).

Every transverser obtained during reduction is a subset of a transverser of $\mathcal{T}$. Thus if X and X^* are unrelated confined transversers (neither is the ancestor of the other), then they are disjoint and there is a confining hyperedge of X that does not intersect all the confining hyperedges of X^* (since if the confining hyperedges of X and X^* were pairwise intersecting, there would, by the Helly property, be a vertex contained in all of them, hence in $X \cap X^*$). So there is a confining hyperedge of X that does not intersect X^*.

Now we assume that X is the first transverser obtained during reduction with $B(X) = \varnothing$, and find a contradiction. We can assume that $|X| \geq 2$ and X is confined. Thus $B(X)$ is equal to the intersection of the confining hyperedges of X and the essential hyperedges of X. As $B(X) = \varnothing$, by Proposition 2, two of these sets, say V_1 and V_2, are disjoint (and V_1 and V_2 must be essential hyperedges of X, as X is a subset of each of its confining hyperedges). Let G be the line graph of H. Apply Lemma 9 with $u = V_1$ and $v = V_2$ (W_1 is non-empty as it includes all the confining hyperedges of X) : we obtain a graph $J = G - (W_1 \cup W_2)$ such that V_1 and V_2 are in separate components of J, say J_1 and J_2.

We shall find a path in J from J_1 to J_2, a contradiction. We need the following result.

Assertion 10 *Suppose that a transverser Y is replaced by $B(Y)$ during reduction and that P is an essential but not confining hyperedge of Y. Then either*

- *$P \cap Z \neq \varnothing$, or*
- *there is a transverser T that was replaced by $B(T)$ earlier during reduction (i.e., before Y is replaced by $B(Y)$), and $P \cap T \neq \varnothing$ but $P \cap B(T) = \varnothing$.*

Proof: Let Y' be the ancestor of Y in $\mathcal{T}$. As P is not a confining hyperedge of Y, it is not a confining hyperedge of Y'. Thus P was covered by another transverser in $\mathcal{T}$ (else $Y' \neq B(Y')$, contradicting that $\mathcal{T}$ is consistent). If this was Z, then $P \cap Z \neq \varnothing$. Otherwise suppose a transverser W covered P. When Y is replaced by $B(Y)$, no descendant of W covers P. Thus at some point before Y was replaced, a descendant of W that does cover P was replaced by its restricted set of alternatives that does not cover P. Let this descendant be T. □

We use the assertion to find a sequence $X_1P_1X_2 \cdots X_rP_r$ where,

- $X_1 = X$ and $P_1 = V_1$;
- for $1 \leq j \leq r$, X_j is a confined transverser, P_j is an essential but not confining hyperedge of X_j;
- for $1 \leq j \leq r-1$, P_j is covered by X_{j+1} but not by $B(X_{j+1})$;
- $P_r \cap Z \neq \varnothing$;
- sets that are not consecutive in the sequence are disjoint.

The first two terms of the sequence are given. When the first $2j$ terms, $X_1P_1X_2 \cdots X_jP_j$, are known, apply Assertion 10 with $Y = X_j$ and $P = P_j$. If $P_j \cap Z \neq \varnothing$, then the sequence is found. Otherwise let $X_{j+1} = T$. Since X was the first transverser encountered with an empty restricted set of alternatives, $B(X_{j+1}) \neq \varnothing$ and is confined. As $B(X_{j+1})$ does not cover P_j, X_{j+1} must have an essential hyperedge that does not cover P_j: let this set be P_{j+1}. Note that P_{j+1} is not a superset of X_j and thus not a confining hyperedge of it. We must show that X_{j+1} and P_{j+1} are each disjoint from the sets they are not consecutive to in the sequence. By the choice of P_{j+1} and X_{j+1}, $P_j \cap P_{j+1} = \varnothing$; and X_{j+1} is unrelated to X_j so they are disjoint. Let Q_j and Q_{j+1} be disjoint confining hyperedges of X_j and X_{j+1}, respectively. Then $P_{j+1} \cap X_j = \varnothing$, else $\{Q_j, P_j, Q_{j+1}, P_j\}$ induces a 4-cycle in G. If $j > 1$, we must show that, for $1 \leq j' \leq j-1$, X_{j+1} and P_{j+1} do not intersect $X_{j'}$ or $P_{j'}$. Apply Lemma 9 with $u = P_j$ and $v = P_{j-1}$ (W_1 is not empty as it includes all the confining hyperedges of X_j): we obtain a graph $J = G - (W_1 \cup W_2)$ such that P_j and P_{j-1} are in separate components of J, say J_1 and J_2. As Q_{j+1} covers P_j and does not intersect X_j, it must be in J_1. For $1 \leq j' \leq j-1$, let $Q_{j'}$ be a confining hyperedge of $X_{j'}$ that does not intersect X_j. Note that $P_{j-1}Q_{j-1}P_{j-2} \cdots P_1Q_1$ is a path in G and must be in J_2 since none of these hyperedges intersect X_j. Thus, for $1 \leq j' \leq j-1$, P_{j+1} and Q_{j+1} (and so X_{j+1}) are disjoint from $P_{j'}$ and $Q_{j'}$.

Every time in the construction above we find a set X_j that is replaced by its restricted set of alternatives $B(X_j)$ during reduction earlier than X_{j-1} was replaced by $B(X_{j-1})$. Therefore after a finite number of steps the sequence must end with a suitable P_r.

Once the sequence is found, let Q'_j, $2 \leq j \leq r$, be a confining hyperedge of X_j that does not intersect $X = X_1$. Then $P_1 Q'_2 P_2 \cdots Q'_r P_r$ must be a path in J_1. Thus we have found a hyperedge P_r in J_1 that intersects Z.

Use the same argument to find a hyperedge W in J_2 that intersects Z (find a path from V_2). If $|Z| \geq 2$, then there is a confining hyperedge U of Z that does not intersect every confining hyperedge of X. Thus $P_{r-1}UW$ is a path in J from J_1 to J_2. If $|Z| = 1$, then $P_{r-1} \cap W \supseteq Z$. Hence $P_{r-1}W$ is an edge in J. This final contradiction completes the proof of Proposition 7. □

Acknowledgements:
The authors thank Alex Berg, András Frank, Bill Jackson, Tibor Jordán and Anders Yeo for stimulating discussions and correspondence. Alex and Tibor are also thanked for directing us to many of the relevant references to the source location problem.

References

[1] K. Arata, S. Iwata, K. Makino, and S. Fujishige, *Locating sources to meet flow demands in undirected networks.* J. of Algorithms, **42** (2002), 54–68.

[2] M. Bárász, J. Becker and A. Frank, *An algorithm for source location in directed graphs.* Preprint (2004).

[3] R. Diestel, *Graph Theory.* Springer-Verlag, New York (1997).

[4] P. Duchet, *Hypergraphs.* In: R. Graham, M. Grötschel, and L. Lovász, eds., *Handbooks of Combinatorics*, Elsevier Science B.V. (1995), 381–432.

[5] J. Hao and J. B. Orlin, *A faster algorithm for finding the minimum cut in a graph.* J. of Algorithms, **17** (1994), 424–446.

[6] H. Ito, M. Ito, Y. Itatsu, K. Nakai, H. Uehara, and M. Yokoyama, *Source Location Problems considering vertex-connectivity and edge-connectivity simultaneously.* Networks, **40** (2002), 63–70.

[7] H. Ito, M. Ito, Y. Itatsu, H. Uehara, and M. Yokohama, *Location problems based on node-connectivity and edge-connectivity between nodes and node-subsets.* Lecture Notes in Computer Science, **1969** (2000), 338–349.

[8] H. Ito, K. Makino, K. Arata, S. Honami, Y. Itatsu, and S. Fujishige, *Source location problem with flow requirements in directed networks.* Optimization Methods and Software, **18** (2003), 427–435.

[9] H. Ito, H. Uehara, and M., Yokoyama, *A faster and flexible algorithm for a location problem on undirected flow networks.* IEICE Trans., **E83-A** (2000), 704–712.

[10] H. Ito and M. Yokoyama, *Edge-connectivity between nodes and node-subsets*, Networks, **31** (1998), 157–164.

[11] M. Labbe, D. Peeters, and J.-F. Thisse, *Location on networks.* In : M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, eds., *Handbooks in Operations Research and Management Science, 8 : Network Routing*, North-Holland (1995), 551–624.

[12] H. Nagamochi, T. Ishii, and H. Ito, *Minimum cost source location problem with vertex-connectivity requirements in digraphs.* Info Process Letters, **80** (2001), 287–294.

[13] H. Tamura, M. Sengoku, S. Shinoda, and T. Abe, *Location problems on undirected flow networks.* IEICE Trans., **E73** (1990), 1989–1993.

[14] Open Problems page of the Egerváry Research Group on Combinatorial Optimization, `http://www.cs.elte.hu/egres/`, `http://www.cs.elte.hu/egres/design/mf_problems.html`.