Egerváry Research Group
on Combinatorial Optimization

Technical reportS

TR-2004-06. Published by the Egrerváry Research Group, Pázmány P. sétány 1/C, H–1117, Budapest, Hungary. Web site: www.cs.elte.hu/egres. ISSN 1587–4451.

An algorithm for source location in directed graphs

Mihály Bárász, Johanna Becker, and András Frank

February 2004

An algorithm for source location in directed graphs

Mihály Bárász, Johanna Becker, and András Frank*

Abstract

Ito, Makino, Arata, Honami, Itatsu, and Fujishige [6] provided a theoretical answer to a source location problem by proving that the minimum cardinality of a subset R of nodes in an edge-capacitated directed graph $D = (V, A)$ so that the maximum flow-amount from R to every node $v \in V - R$ is at least k and the maximum flow amount from every node $v \in V - R$ to R is at least l is equal to the maximum number of pairwise disjoint deficient sets where a subset of nodes is deficient if its in-capacity is less than k or its out-capacity is less than l. They also showed how this theorem gave rise to a polynomial time algorithm to compute the optima in question in case the demands k and l are fixed, and posed as an open problem of developing an algorithm that is (strongly) polynomial not only in the size of the digraph but in k and l, as well. The present work describes such an algorithm.

1 Introduction

Source location problems in directed or undirected graphs concerning various connectivity type properties have recently been intensively investigated. In this paper we concentrate on the edge-connectivity version in digraphs. For non-negative integers k, l, we call a directed graph $D = (V, A)$ (k, l)**-edge-connected** with respect to a root node r if there are k-edge-disjoint paths from r to every other node and there are l edge-disjoint paths from every node to r. By the directed edge-version of Menger's theorem this is equivalent to requiring that the indegree and outdegree of all nonempty subsets of $V - r$ is at least k and l, respectively. When $k = l$, this notion is equivalent to k-edge-connectivity of D, while the case $l = 0$ corresponds to the rooted k-edge-connectivity of D.

Call a subset R of nodes of a digraph a (k, l)**-source** if the contraction of R into a node r results in a (k, l)-edge-connected digraph with respect to root node r. Equivalently, there are k edge-disjoint paths from R to every node and there are l edge-disjoint paths from every node to R. Yet another equivalent formulation is that R

*Egerváry Research Group of MTA-ELTE and Department of Operations Research, Eötvös University, Pázmány P. s. 1/c. Budapest, Hungary, H-1117. Supported by the Hungarian National Foundation for Scientific Research, OTKA T037547, and ADONET. The third author is also supported by Ericsson Hungary, Laborc u.1, Budapest, Hungary H-1037. e-mails: {barasz,beckerjc,frank}@cs.elte.hu

covers all deficient sets where a nonempty proper subset of nodes is **deficient** if its indegree is smaller than k or its outdegree is smaller than l.

Ito et al. [6] introduced and investigated the problem of finding a (k, l)-source of minimum cardinality which may be called the directed source location problem (DSL, in short). While they pointed out that several closely related problems are NP-complete, they also showed, by proving the following fundamental min-max theorem, that DSL belongs to **NP**∩co-**NP**.

Theorem 1.1 ([6]). *The minimum cardinality of a (k, l)-source is equal to the maximum number of pairwise disjoint deficient sets.*

They proved this result by observing first that in order to cover all deficient sets, it is sufficient to cover only the minimal ones. Second, they proved that the family of all minimal deficient sets form a subtree hypergraph H_{kl}. Finally, they invoked the known result that in any subtree hypergraph H the transversal number $\tau(H)$ is equal to the matching number $\nu(H)$.

A **subtree hypergraph** (sometimes called arboreal hypergraph) is one for which there is a spanning tree T on its node set so that each hyperedge induces a subtree of T. The tree is called a **basic** (or representative) **tree** for the hypergraph. Note that the hypergraph of all deficient sets is not necessarily a subtree hypergraph. In a hypergraph $H = (V, \mathcal{E})$ a family of pairwise disjoint hyperedges is called a **matching.** The largest cardinality $\nu(H)$ of a matching is the **matching number** of H. A subset of nodes Z intersecting each hyperedge is called a **transversal** of H if Z. The smallest cardinality $\tau(H)$ of a transversal is the **transversal number** of H.

There is a well-known algorithm, due to F. Gavril [3], to compute a maximum stable set in a chordal graph along with a minimum set of cliques covering all nodes. This algorithm can, in principle, be used for solving the DSL problem since the line graph $L(H)$ of an arbitrary subtree hypergraph H is chordal, and a stable set of nodes in $L(H)$ corresponds to a matching of H, while a clique covering of $L(H)$ corresponds to a transversal of H.

Therefore the approach of [6] gave rise to an algorithm for the directed source location problem which is polynomial *provided* that the number of minimal deficient sets (which is the number of nodes of $L(H)$) depends polynomially on the size of D. [6] concluded that this is the case when k and l are considered fixed since every deficient set Z is determined by the set of edges entering (or leaving) Z and the number of subsets of edges smaller than $\max\{k, l\}$ is polynomial in $|A|$. That is, the algorithm of [6] outlined above is polynomial for fixed k, l.

By the following example, however, they also showed that the number of minimal deficient sets may be exponential in the size of the digraph even if $l = 0$. Define a digraph on n nodes ($n > k$) with a special node s so that, for every other node v, there is one edge from v to s and there are k parallel edges from s to v. Then the minimal in-deficient sets are precisely those containing s and having $n - k + 1$ elements. In the concluding remarks of their paper, Ito at al. formulated the open problem of developing an algorithm that is (strongly) polynomial not only in the size of the digraph but in k and l, as well. The present work describes such an algorithm. While completing this work, we learnt that J. van den Heuvel and M. Johnson [5]

also developed a polynomial algorithm for the $(k,0)$-source problem. Their approach, which is completely different from ours, seems to be extendible to the general (k,l) case.

Actually, [6] proved Theorem 1.1 in the slightly more general capacitated case when a nonnegative capacity function is given on the edge-set and the requirement for a (k,l)-source set R is that the flow-amount from R to every node is at least k and from every node to R is at least l. Our algorithm is strongly polynomial in this edge-capacitated case, as well, but technically it is a bit easier to work with the uncapacitated case (when the capacity function is identically one). Therefore in the description below we consider the the edge-connectivity version of the problem formulated above.

1.1 Further notation and notions

For elements $s,t \in V$ a subset $X \subset V$ is called a $t\bar{s}$-set if $t \in X \subseteq V - s$. Throughout we denote $|V| = n$ and $|A| = m$. We will not distinguish between a one-element set and its only element.

A graph is called **chordal** if there is no induced circuit of length at least four, or in other words every circuit of length at least four admits a chord. It is well-known that every chordal graph contains a so-called **simplicial** node, a node whose neighbours form a clique. By an **arborescence** of root s we mean a directed tree in which every node is reachable from s.

A hypergraph is said to admit the **Helly property** or is of **Helly type** if any subset of pairwise intersecting hyperedges has a nonempty intersection. A hypergraph is **laminar** if at least one of the sets $X-Y, Y-X, X\cap Y$ is empty for any two members X, Y.

The **line graph** $L(H)$ of a hypergraph H is a graph in which the nodes correspond to the hyperedges two of them being adjacent if the corresponding hyperedges have a nonempty intersection. It follows from the definitions that $\nu(H)$ is the stability number $\alpha(L(H))$ of $L(H)$ while $\tau(H)$ is at most the clique-covering number of $L(H)$ (which is, by definition, the chromatic number of the complement of $L(H)$) with equality for hypergraphs of Helly type.

In a digraph $D = (V, A)$, the indegree $\varrho(X) = \varrho_D(X)$ denotes the number of edges entering X while the outdegree $\delta_D(X) = \delta(X)$ is the number of edges leaving X. If a nonnegative capacity function g is given on the edge set, $\varrho_g(X)$ denotes the **in-capacity** of X, that is, the sum of capacity values on the edges entering X. The **out-capacity** $\delta_g(X)$ of X is the sum of capacity values on the edges leaving X. For two disjoint non-empty sets S and T of V let $\lambda_g(S,T)$ denote the maximum flow-amount from s to t in the digraph arising from D by contracting S and T into nodes s and t, respectively. By the Max-flow Min-cut (MFMC) theorem, $\lambda_g(S,T)$ is equal to the minimum in-capacity of a subset Z with $T \subseteq Z \subseteq V - S$. In the uncapacitated case $\lambda(S,T)$ is the minimum in-degree of sets Z with $T \subseteq Z \subseteq V - S$. By the directed edge-version of Menger's theorem, $\lambda(S,T)$ is equal to the maximum number of edge-disjoint paths from S to T. It is known that the minimizer sets are closed under taking union and intersection. (Indeed, by using the submodularity of ϱ_g one has

$\lambda_g(S,T)+\lambda_g(S,T) = \varrho_g(X)+\varrho_g(Y) \geq \varrho_g(X\cap Y)+\varrho_g(X\cup Y) \geq \lambda_g(S,T)+\lambda_g(S,T)$ from which $\varrho_g(X\cap Y) = \lambda_g(S,T) = \varrho_g(X\cup Y)$.)

Let Z_{min} and Z_{max} denote the unique minimal and maximal member of this family. With the help of a max-flow min-cut (MFMC) computation one can compute not only a maximum flow (or the edge-disjoint paths) from S to T and a minimizer set but the set Z_{min}, as well. For example, if x is a maximum flow, then Z_{min} is nothing but the set of nodes from which T is reachable in the auxiliary digraph defined by x in the Ford-Fulkerson algorithm. That is, once a maximum flow x is already computed Z_{min}, and analogously Z_{max}, is computable by a search algorithm in $O(|A|)$ time. Note that a typical MFMC algorithm based on alternating-paths can easily be modified so as to output a Z_{max} as a minimizer set. The extra search for Z_{min} or Z_{max} may be needed if another MFMC algorithm is applied. We will use these facts without any further reference.

Given nonnegative values k and l, a nonempty proper subset X of nodes is called **k-in-deficient** or simply **in-deficient** if $\varrho(X) < k$ (or in the capacitated case $\varrho_g(X) < k$) and **l-out-deficient** or simply **out-deficient** if $\delta(X) < l$ (or in the capacitated case $\delta_g(X) < l$). An in- or out-deficient set is called deficient. For a property P of subsets we say that a subset Z is minimal with respect to P if Z admits property P but its nonempty proper subsets do not. In this sense we will, for example, speak of minimal (in- or out-) deficient subsets.

2 Subtree hypergraphs

The algorithm suggested by [6] works on the line graph $L(H_{kl})$ of minimal deficient sets. Since this may be exponential in the size of the digraph D, one must avoid an algorithm running on the line graph. We will work directly with the hypergraph of minimal deficient sets (and even with an appropriate extension of it) since the complete list of the hyperedges will not really be needed. Instead, we must only be able to realize polynomially certain subroutines.

2.1 Computing a basic tree

The following simple but useful theorem was discovered by several authors. (For references, see [1].) For completeness, we include its proof, as well.

Theorem 2.1. *A hypergraph $H = (V,\mathcal{E})$ is a subtree hypergraph if and only if H admits the Helly property and the line graph $L(H)$ of H is chordal.*

Proof. The necessity is clear. To see the sufficiency we use induction on $|V|$. The case $|V| = 1$ is trivial so assume that $|V| > 1$. We may also assume that the cardinality of every hyperedge is at least 2. Since $L(H)$ is chordal, it contains a simplicial node. This corresponds to a hyperedge Z of H so that every two hyperedges intersecting Z intersect each other. By the Helly property it follows that Z must have an element z covering all hyperedges intersecting Z. Let u be another element of Z and $H' = (V-u,\mathcal{E}')$ be a hypergraph where $\mathcal{E}' := \{X-u : X\in\mathcal{E}\}$. By the choice of u, H'

also has the Helly property and its line graph is $L(H)$. By induction, there is a basic tree T' for H'. Let T be a tree arising from T' by adding a new edge zu. It follows from the construction that T is basic for H. •

The constructive nature of this proof allows one to turn it into an algorithm which is polynomial in $|\mathcal{E}|$. It is useful however to realize that deciding if a hypergraph admits a basic tree is a matroid optimization problem and thus a basic tree may be found by the greedy algorithm, as well. To this end, define a weight function $c(uv)$ on the edge-set of the complete graph on V as follows. For every pair $\{u, v\}$ of nodes,

$$\text{let } c(uv) \text{ be the number of hyperedges containing both } u \text{ and } v.$$

Theorem 2.2. *A hypergraph H admits a basic tree (that is, H is a subtree hypergraph) if and only if the maximum c-weight spanning tree is a basic tree for H.*

Proof. To see the non-trivial direction, let Z be a hyperedge and T an arbitrary spanning tree. Then Z induces at most $|Z| - 1$ edges of T. Therefore

$$c(T) := \sum_{uv \in E(T)} c(uv) = \sum_{uv \in E(T)} \sum [1 : Z \in \mathcal{E}, \{u, v\} \subseteq Z] \leq \sum_{Z \in \mathcal{E}} (|Z| - 1) \qquad (1)$$

and equality holds if and only if every hyperedge Z induces precisely $|Z| - 1$ edges of T, that is, if T is basic for H. •

It follows that Kruskal's algorithm can be used to compute a basic tree for a subtree hypergraph. Again, this algorithm is polynomial in the number of hyperedges. (In a matroid one may be interested in finding a basis B that spans some specified subsets $S_1, \ldots, S_t$, that is, $|B \cap S_i| = r(S_i)$ for all i. By defining $c(uv)$ to be the number of sets S_i containing both u and v, the maximum weight basis will do if there is such a basis at all. That is, the matroid greedy algorithm solves the problem and our approach above is just a specialization of the matroid case.)

3 Solid sets and partitions

How can one apply this algorithm to the hypergraph H_{kl} of minimal deficient sets where the number of hyperedges may be exponential and the running time is expected to be polynomial in the size of D? To answer this question we introduce a subtree hypergraph H_D on V which includes H_{kl} as a subhypergraph and show that there exists a small, well-computable subhypergraph H'_D of H_D such that any tree basic for H'_D is basic for H_D and hence for H_{kl}, as well. Therefore it will suffice to compute a basic tree for H'_D.

Given a digraph $D = (V, A)$, we call a nonempty proper subset Z of V **in-solid** (respectively, **out-solid**) if $\varrho(X) > \varrho(Z)$ (respectively, $\delta(X) > \delta(Z)$) for every nonempty proper subset X of Z. An in- or out-solid set is called **solid**. Singletons are always in-

and out-solid, and a minimal k-in-deficient set is in-solid (for any k). A k-in-deficient in-solid set is not necessarily a minimal k-in-deficient set. Let $H_D = (V, \mathcal{E}_D)$ denote the hypergraph of all solid sets. Note that the definition of deficient sets depends on parameters k and l while that of solid sets does not. We remark that an analogous notion for undirected graphs, under the name of extreme sets, was introduced and successfully used to solve edge-connectivity augmentation problem by Watanabe and Nakamura [9]. But the structure of extreme sets of undirected graphs, as being laminar, is much simpler than that of solid sets of digraphs.

As mentioned, [6] proved that minimal deficient sets form a subtree hypergraph. We can use their proof-technique almost word for word to show that the hypergraph H_D of solid sets is also a subtree hypergraph. Let us start with the following useful observation.

Lemma 3.1. *If X is in-solid and Y is out-solid, then at least one of the subsets $A := X - Y, B := Y - X, C := A \cap B$ is empty.*

Proof. Let $\alpha, \beta, \gamma, \gamma'$ denote, respectively, the number of edges from C to A, from B to C, from $V - (X \cup Y)$ to C, and from C to $V - (X \cup Y)$. If, indirectly, none of A, B, C is empty, then $\varrho(A) > \varrho(X)$ and $\delta(B) > \delta(Y)$. Therefore $\alpha > \beta + \gamma$ and $\beta > \alpha + \gamma'$ from which the impossible $0 > \gamma + \gamma'$ follows. •

Theorem 3.2. *The hypergraph $H_D = (V, \mathcal{E}_D)$ of solid sets is a subtree hypergraph.*

Proof. We claim that the line graph of H_D is chordal. If, indirectly, it induces a chordless circuit of length at least 4, then there are solid sets $X_1, \ldots, X_h$ $(h \geq 4)$ so that $X_i \cap X_j \neq \emptyset$ if and only if i and j are consecutive integers where we use the notational convention $X_{h+1} = X_1$. Lemma 3.1 implies that either all X_i's are in-solid or all X_i's are out-solid, say the first case occurs. It follows that the h intersections $X_i \cap X_{i+1}$ are pairwise disjoint and hence

$$\sum_{i=1}^{h} \varrho(X_i \cap X_{i+1}) \leq \sum_{i=1}^{h} \varrho(X_i) \tag{2}$$

Since X_i is in-solid, $\varrho(X_i) < \varrho(X_i \cap X_{i+1})$ for $i = 1, \ldots, h$ and hence $\sum_i \varrho(X_i) < \sum_i \varrho(X_i \cap X_{i+1})$, contradicting (2).

We claim that H_D admits the Helly property. If it does not, then there is a smallest number h along with h solid sets $X_1, \ldots, X_h$ so that any two of these sets intersect each other but the intersection $M = X_1 \cap \ldots \cap X_h$ is empty. Again, by Lemma 3.1 either the sets X_i's are all in-solid or all out-solid. By symmetry we may assume that each X_i is in-solid. Let $Y_i = X_1 \cap X_2 \cap \ldots \cap X_{i-1} \cap X_{i+1} \cap \ldots \cap X_h$ $(i = 1, \ldots, h)$. By the minimal choice of h, $Y_i \neq \emptyset$, while $M = \emptyset$ implies that $Y_i \cap Y_j = \emptyset$ $(1 \leq i < j \leq h)$. If an edge enters one of the sets Y_i, then it enters at least one of the sets X_j. Therefore $\sum_i \varrho(Y_i) \leq \sum_i \varrho(X_i)$. On the other hand $\varrho(Y_i) > \varrho(X_{i+1})$ for each i as X_{i+1} is in-solid and hence $\sum_i \varrho(Y_i) > \sum_i \varrho(X_{i+1}) = \sum_i \varrho(X_i)$, a contradiction.

Theorem 2.1 implies that H_D is indeed a subtree hypergraph. •

By a **maximal s-avoiding in-solid** (**out-solid**) set we mean an in-solid (out-solid) subset of $V - s$ not included in any other s-avoiding in-solid (out-solid) subset of $V - s$.

Theorem 3.3. *For any element $s \in V$, the maximal s-avoiding in-solid (out-solid) sets are pairwise disjoint.*

Proof. Suppose indirectly that there are two maximal s-avoiding in-solid sets X, Y with a nonempty intersection. By the maximality of X and Y, none of $X - Y$ and $Y - X$ is empty and $X \cup Y$ is not in-solid. Hence there is a maximal nonempty subset $Z \subset X \cup Y$ with $\varrho(Z) \leq \varrho(X \cup Y)$.

If Z includes one of X and Y, say X, then $Z \cap Y \subset Y$, $X \cup Y = Z \cup Y$ and hence $\varrho(Z \cap Y) > \varrho(Y)$, $\varrho(X \cup Y) = \varrho(Z \cup Y) \geq \varrho(Z)$ from which $\varrho(Y) + \varrho(Z) \geq \varrho(Z \cap Y) + \varrho(Z \cup Y) > \varrho(Y) + \varrho(Z)$ would follow. Therefore Z can include neither X nor Y.

If Z is disjoint from X or Y, say from X, that is, $Z \subseteq Y - X$, then $\varrho(Z) > \varrho(Y)$ which is not possible since $\varrho(X) + \varrho(Y) \geq \varrho(X \cap Y) + \varrho(X \cup Y) > \varrho(X) + \varrho(X \cup Y)$ implies $\varrho(Y) > \varrho(X \cup Y)$ from which we would have $\varrho(Z) > \varrho(X \cup Y)$ contradicting the assumption $\varrho(Z) \leq \varrho(X \cup Y)$. Therefore Z must intersect both X and Y.

It follows that $X \cap Z \neq \emptyset$ and $X \cap Z \subset X$ from which $\varrho(X \cap Z) > \varrho(X)$ as X is in-solid. Since $Z \subset X \cup Z$, the maximal choice of Z implies $\varrho(X \cup Z) \geq \varrho(Z)$. Therefore we have $\varrho(X) + \varrho(Z) \geq \varrho(X \cap Z) + \varrho(X \cup Z) > \varrho(X) + \varrho(Z)$, a contradiction.

That is, the maximal s-avoiding in-solid sets are indeed disjoint. The proof for out-solid sets is analogous. •

Since each singleton is in-solid, the maximal s-avoiding in-solid sets partition $V - s$. This will be called the **in-solid partition** of $V - s$. The out-solid partition of $V - s$ is defined analogously. It follows from Theorem 3.3 and Lemma 3.1 that:

Corollary 3.4. *The family of maximal s-avoiding solid sets is a partition of $V - s$.* •

We call this partition the **solid partition** of $V - s$.

3.1 Computing the solid partition of $V - s$

By Corollary 3.4 the members of the in-solid partition and the out-solid partition of $V - s$ form a laminar family $\mathcal{L}$. Therefore the solid partition of $V - s$ consists of the maximal members of $\mathcal{L}$. Hence, in order to compute the maximal solid partition of $V - s$, it suffices to compute separately the in-solid and the out-solid partition of $V - s$. Since the two computations are analogous, we describe only the first one to compute the in-solid partition of $V - s$.

As mentioned in the introduction the maximum number $\lambda(t) := \lambda(s, t)$ of edge-disjoint paths from s to a node $t \in V - s$ is equal to the minimum indegree of the $t\bar{s}$-sets, and the minimizer sets are closed under taking union and intersection. Let N_t denote the unique minimal member of this family.

Theorem 3.5. *If N is a minimal member of the family $\{N_t : t \in V - x\}$, then N is a maximal s-avoiding in-solid set.*

Proof. We claim that $z \in N_t$ implies that $N_z \subseteq N_t$ for any $z, t \in V - s$. Indeed, if we had, indirectly, $N_z - N_t \neq \emptyset$, then $\varrho(N_z \cap N_t) > \varrho(N_z)$ from which $\varrho(N_z) + \varrho(N_t) \geq \varrho(N_z \cup N_t) + \varrho(N_z \cap N_t) > \lambda(t) + \varrho(N_t) \geq \varrho(N_z) + \varrho(N_t)$ would follow.

This and the minimality of N imply that $N = N_t$ for every element $t \in N$ and hence N is in-solid. Furthermore there are $\varrho(N)$ edge-disjoint paths from s to t, therefore $\varrho(Z) \geq \varrho(N)$ whenever $N \subseteq Z \subseteq V - s$, that is, N is maximally in-solid in $V - s$. •

Based on this, the in-solid partition of $V - s$ can be computed as follows. First compute all sets N_t ($t \in V - s$) and choose the smallest of them, denoted by N_1. By Theorem 3.5, N_1 is a maximal s-avoiding in-solid set. Contract s and N_1 into a node s_1 and compute in a similar manner a maximal s_1-avoiding in-solid set N_2 in the contracted digraph. Since the maximal s-avoiding in-solid sets in D are disjoint, N_2 is a maximal s-avoiding in-solid set in D. At the general step, contract s and the already computed maximal s-avoiding in-solid sets $N_1, \ldots, N_h$ into a node s_h and compute a maximal s_h-avoiding solid set N_{h+1} of the contracted digraph. The algorithm terminates when the union of the current sets $N_1, \ldots, N_h$ is $V - s$.

To describe the algorithm more formally, let $\mathcal{N}$ denote the current family of maximal disjont in-solid subsets of $V - s$. Instead of carrying out the contractions we will maintain a subset S that is the union of the members of $\mathcal{N}$ plus s.

Algorithm for computing the in-solid partition of $V - s$.

INPUT Digraph $D(V, A)$ and a node $s \in V$.
OUTPUT The in-solid partition $\mathcal{N}$ of $V - s$

(P1) Set $\mathcal{N} := \emptyset$ and $S := \{s\}$.

(P2) If $V - S$ is empty, output $\mathcal{N}$. STOP. (The algorithm terminates.)

(P3) For each $t \in V - S$, with the help of an MFMC routine, compute $\lambda(S, t)$ and the unique smallest set N_t for which $t \in N_t \subseteq V - S$ and $\varrho(N_t) = \lambda(S, t)$. Let N be a smallest member of the family $\{N_t : t \in S - V\}$. Add $\{N\}$ to $\mathcal{N}$. Set $S := S \cup N$. Go to (P2).

3.2 Computing a basic tree for H_D

Given the solid partition of $V - s$ for every node $s \in V$, let H'_D be the subhypergraph of H_D consisting of those hyperedges which occur in the solid partition of $V - s$ for some $s \in V$. Note that H'_D has at most n^2 hyperedges, that is, H'_D is small even if H_D has exponentially many hyperedges. Therefore one can compute a basic tree for H'_D in polynomial time with the greedy algorithm described in the previous section. That is, for each pair of nodes $u, v \in V$, let $c(uv)$ denote the number of hyperedges of $\mathcal{H}'_{\mathcal{D}}$ containing both u and v. With the help of the greedy algorithm, compute a maximum weight tree T. The good thing is that this tree is automatically basic for the whole H_D.

Theorem 3.6. *If T is a basic tree for H'_D, then T is basic for the hypergraph H_D of all solid sets (and hence for its subhypergraph H_{kl}) of deficient sets.*

Proof. Suppose indirectly that there is solid set Z that does not induce a subtree of T. Then there are two elements a, b of Z so that the unique path P in T connecting a and b contains a node s not belonging to Z. That is, Z is an s-avoiding solid set and hence there is a maximal s-avoiding solid set Z' including Z. But T is basic for H'_D and hence the whole P must belong to Z', a contradiction. •

4 Computing a minimum transversal and a maximum matching

Let $H = (V, \mathcal{E})$ be an arbitrary subtree hypergraph and T a basic tree for H. [2] describes an algorithm for computing a minimum transversal and a maximum matching of H that works directly on the basic tree T for H, rather than using the line graph of H. Actually, that algorithm settles a weighted case as well but we need only its unweighted version. We exhibit first this Generic Algorithm in which it does not matter how the input hypergraph is 'given' and the main step is described in a general form. We show then a more detailed and specialized version that applies to the hypergraph H_{kl} of minimal deficient sets.

We need some notation. Choose an arbitrary node s of T as a root node. Let $\vec{T}$ denote the arborescence arising from T by orienting each edge of T away from s. Define the **height** of a node v to be the distance of v from s in $\vec{T}$. A node v is said to be **above** a node $u \neq v$ if there is a path in $\vec{T}$ from u to v. For a hyperedge Z of H, the **bottom node** $b(Z)$ of Z is the (unique) lowest node of Z. The height of Z is defined to be the height of its bottom node. We say that a hyperedge Z of H is **independent** from a matching $\mathcal{M}$ if Z is disjoint from the members of $\mathcal{M}$, that is, if $\mathcal{M} + \{Z\}$ is a matching.

Starting with the empty matching $\mathcal{M}$, in each step of the algorithm, we find a highest hyperedge which is independent from the current matching $\mathcal{M}$ and add it to $\mathcal{M}$. The algorithm teminates when there is no more hyperedge independent from $\mathcal{M}$. We will prove that the bottom nodes of the members of $\mathcal{M}$ is a transversal of H. Let us describe the algorithm a bit more formally.

Generic Algorithm for computing a maximum matching and a minimum transversal of a subtree hypergraph.

INPUT: A subtree hypergraph $H = (V, \mathcal{E})$ along with a basic tree T for H.
OUTPUT: A matching $\mathcal{M}$ and a transversal R of H so that $|\mathcal{M}| = |R|$.

(GA1) Set $\mathcal{M}$ and R to be empty.

(GA2) Find a highest hyperedge Z which is independent from $\mathcal{M}$. If there is no hyperedge, output $\mathcal{M}$ and R. STOP. (The algorithm terminates).

(GA3) Add $\{Z\}$ to $\mathcal{M}$. Add the bottom node $b(Z)$ to R. Go to **(GA2)**.

The correctness of the algorithm along with the proof of the min-max relation $\nu(H) = \tau(H)$ for subtree hypergraphs follow from the following observation.

Lemma 4.1. *The set R of bottom nodes output by the algorithm covers all hyperedges.*

Proof. Suppose indirectly that there is a hyperedge Y not covered by R. Since the algorithm terminates when there is no more hyperedge independent from the current $\mathcal{M}$, Y must intersect a member of $\mathcal{M}$. Among these members, let Z be the one which was added earliest to $\mathcal{M}$. Then Y is disjoint from each member of $\mathcal{M}$ that has been added to $\mathcal{M}$ previous to Z. Since $b(Z)$ is not in Y but $Z \cap Y \neq \emptyset$, it follows that $b(Y)$ is above $b(Z)$ contradicting the 'highest' rule of (GA2). •

We describe now more specifically how Step (GA2) can be carried out. Instead of trying to find directly a highest hyperedge in Step (GA2) which is independent from $\mathcal{M}$, the algorithm considers the nodes of H in a decreasing order according to their height, and checks whether or not the current node is the bottom node of a hyperedge Z which is independent from $\mathcal{M}$. In the first case Z is added to $\mathcal{M}$ and the algorithm turns to the next node.

To describe more formally this algorithm, we let $A(X)$ denote, for a subset $X \subseteq V$, the set of nodes reachable from X in $\vec{T}$. For a singleton $\{v\}$ we write $A(v)$ and let $B(v) := V - A(v)$. Then $v \in A(v)$ and $V = A(s)$. In addition to the matching $\mathcal{M}$ and the set R of the bottom nodes of the member of $\mathcal{M}$, the algorithm maintains a label marked-unmarked assigned to the nodes.

Specific Algorithm for computing a maximum matching and a minimum transversal of a subtree hypergraph H

INPUT: A subtree hypergraph $H = (V, \mathcal{E})$ along with a basic tree for T for H.
OUTPUT: A matching $\mathcal{M}$ and a transversal R of H so that $|\mathcal{M}| = |R|$

(SA1) Set $\mathcal{M}$ and R to be empty, and let each node be unmarked.

(SA2) If there is no unmarked node, output $\mathcal{M}$ and R. STOP. (The algorithm terminates).

(SA3) Choose a highest unmarked node v and mark it. Let $S(v) := B(v) \cup A(R)$.

(SA4) Find a hyperedge Z for which $v \in Z \subseteq V - S(v)$. If no such a hyperedge exists go to Step (SA2).

(SA5) Add Z to $\mathcal{M}$ and add $b(Z)$ to R. Go to Step (SA2).

The correctness of this algorithm follows from that of the Generic Algorithm since a node v get marked only when $A(v) - v$ includes no hyperedge disjoint from R. Hence we have:

PROPERTY $(*)$ Every hyperedge included in $V - S(v)$ must contain v.

4.1 Realizing Step (SA4) for H_{kl}

Let us return to our initial problem of computing a minimum (k, l)-source set, that is, a minimum transversal of the hypergraph H_{kl} of minimal deficient sets along with a maximum matching. In the preceding section we showed how to compute a basic tree T for H_{kl}. Now we want to apply the algorithm above to H_{kl}, a situation where the list of hyperedges is not explicitly listed. The only task is to realize Step (SA4). To this end, let v be the node considered in Steps (SA3) and (SA4).

(SA4.1) Compute $\lambda(S(v), v)$ along with the unique minimal set Z' for which $v \in Z' \subseteq V - S(v)$ and $\varrho(Z') = \lambda(S(v), v)$. Compute $\lambda(v, S(v))$ along with the unique minimal set Z'' for which $v \in Z' \subseteq V - S(v)$ and $\delta(Z'') = \lambda(v, S(v))$.

(SA4.2) If $\lambda(S(v), v) \geq k$ and $\lambda(v, S(v)) \geq l$, then (by Property $(*)$) the hyperedge for (SA4) does not exist. Go to Step (SA2).

(SA4.3) If $\lambda(S(v), v) < k$ and $\lambda(v, S(v)) < l$, then let Z be the smaller of Z' and Z''. If exactly one of $\lambda(S(v), v) < k$ and $\lambda(v, S(v)) < l$ holds, then let Z be, accordingly, Z' or Z''. Turn to Step (SA5) with this Z.

The only property we have to check is that the subset Z constructed this way is hyperedge of H_{kl}.

Claim 4.2. *Subset Z is minimal deficient, that is, Z is a hyperedge of H_{kl}.*

Proof. If $\lambda(S(v), v) < k$ and $\lambda(v, S(v)) < l$, then by Property $(*)$, Z' is minimal in-deficient and Z'' is minimal out-deficient. By Lemma 3.1 one of them includes the other, hence Z is minimally deficient. If exactly one of $\lambda(S(v), v) < k$ and $\lambda(v, S(v))$ holds, then by Property $(*)$ again, Z is minimal deficient. •

5 Running time and conclusions

The algorithm we outlined above for solving the (k, l)-source location problem consists of three consecutive phases:

1. Computing the solid partition of $V - s$ for each $s \in V$.

2. Computing a basic tree T for H_D.

3. Computing a minimum (k, l)-source (and a maximum matching) using T.

Note that only the last step depends on k and l, so in order to solve the (k, l)-source location problem for several values of k and l, only the third step should be repeated.

Let $S(n, m)$ denote the complexity of a MFMC algorithm on a digraph with n nodes and m edges. As we mentioned in Section 3, one member of the solid partition of $V - s$ may be obtained by running an MFMC algorithm n times. Thus the in-solid partition of $S - v$, and analogously the out-solid partition as well, can be

computed in $O(n^2 S(n,m))$. Since we need this for all nodes, the total time of Phase 1 is $O(n^3 S(n,m))$.

To compute a basic tree for H_D, we first have to determine the weight function c corresponding to H'_D, and find then a maximum weight spanning tree T. So this phase can be bounded by $O(n^3)$.

The third phase of the algorithm applies the MFMC algorithm twice for every node v (to get the maximum flow-amount and the min-cut from v to $S(v)$ and from $S(v)$ to v). Hence this is doable in $O(nS(n,m))$.

We can conclude that the bottleneck of the whole algorithm is Phase 1, therefore we want to improve on this.

5.1 Computing the solid partition via the algorithm of Hao and Orlin

Hao and Orlin [4] invented an $O(nm\log(n^2/m))$ algorithm to compute a minimum cut in a digraph between a given node s and all the other nodes $t \in V - s$. With a slight modification of their algorithm (which does not increase its complexity), one can obtain the unique minimal minimizer set N_t. Namely, the Hao-Orlin algorithm maintains a feasible preflow, so when it finds a $t\bar{s}$-set with $\lambda_g(s,t)$ out-capacity, then a maximum flow can immediately be obtained from the available preflow, and then one more search algorithm gives rise to N_t. That is, the additional time we need is $O(mn)$ which does not affect the total complexity of the Hao-Orlin algorithm.

Summing up, when the algorithm of Hao and Orlin is used in Phase 1, the total complexity of our algorithm is $O(n^3 m\log(n^2/m))$.

5.2 Conclusion

We developed a strongly polynomial time algorithm for the (k,l)-source location problem introduced and analyzed in [6]. An advantage of this approach is that it can be used to solve the following inverse problem: given the digraph D and a number C, what is the maximum value $k = k(C)$ so that there is a C-element subset of V whose contraction to a node gives rise to a k-edge-connected digraph (or in another version, to a $(k,0)$-edge-connected digraph). In the uncapacitated case this question can be easily answered: simply run the algorithm above for all possible values $1, 2, \ldots, M$, where M is the maximum of the in-degrees and the out-degrees of the nodes, and choose the largest k for which the resulting minimum (k,k)-source set has at most C elements. This approach is certainly not strongly polynomial in the capacitated case but an elegant idea of N. Megiddo [7] can be used to show that n applications of our algorithm $k(C)$ can be obtained.

References

[1] P. Duchet, Hypergraphs, (Theorem 3.8) in: Handbook of Combinatorics (eds. R. Graham, M. Grötschel, L. Lovász), Elsevier Science B. V. (1995), pp. 381-432.

[2] A. Frank, Some polynomial algorithms for certain graphs and hypergraphs, in: Proceedings of the Fifth British Combinatorial Conference, (1975) Congressus Numerantium XV, Eds. C. Nash-Williams and J. Sheehan, pp. 211-226.

[3] F. Gavril, Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph, SIAM Journal on Computing 1 (1972) 180-187.

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

[5] J. van den Heuvel and M. Johnson, A polynomial algorithm for the source location problem in digraphs, CDAM Research Report, LSE-CDAM-2004-02.

[6] Hiro Ito, Kazuhisa Makino, Kouji Arata, Shoji Honami, Yuichiro Itatsu, and Satoru Fujishige, Source location problem with flow requirements in directed networks, Optimization Methods and Software, Vol. 18, No. 4, August 2003, pp. 427-435.

[7] N. Megiddo, Combinatorial optimization with rational objective functions, Mathematics of Operations Research, Vol. 4 (1979) 414-424.

[8] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, Springer, 2003. Vol. 24 of the series Algorithms and Combinatorics.

[9] T. Watanabe and A. Nakamura, Edge-connectivity augmentation problems, Computer and System Sciences, **35** No.1, (1987) 96–144.