Available online at www.sciencedirect.com
SCIENCE DIRECT
Operations Research Letters 33 (2005) 221–230
Operations Research Letters
www.elsevier.com/locate/dsw

An algorithm for source location in directed graphs

Mihály Bárász, Johanna Becker, András Frank*,[1]

Department of Operations Research, Eotvos University, Pazmany Peter setany 1/C, H-1117 Budapest, Hungary

Received 11 March 2004; received in revised form 4 July 2004
Available online 12 September 2004

Abstract

A polynomial time solution algorithm is described to find a smallest subset R of nodes of a directed graph $D=(V, A)$ such that, for every node $v \in V - R$, there are k edge-disjoint paths from R to v and there are l edge-disjoint paths from v to R.

Keywords: Source location; Edge-connectivity; Polynomial algorithm

1. Introduction

Discrete location problems are about finding an optimal placement of some facilities (shops, telecommunication centers, factories) in a network so as to satisfy certain customer demands. Typically it is the distance that matters in defining the constraints and the objective functions. For an annotated bibliography of the topic, see the work of Labbé and Louveaux [8]. Source location is a new type of location problem where the flow-amount or connectivity rather than the distance between facilities and customers is taken into consideration. Source location may serve as a useful optimization framework for designing fault-tolerant telecommunication networks. For example, imagine such a network in which a subset R of nodes is considered a suitable source-set if there are k edge-disjoint paths from R to every node not in R and the objective is to compute a smallest source-set.

There are several versions of source location problems, depending on the type of connectivity used in the constraints. Ito et al. [7] considered and analyzed the source location problem in directed graphs constrained with edge-connectivity or maximum flow-amounts. Their paper is a good overview of other models and results, as well, and it is the starting point of the present work. They proved a min–max theorem for the minimum cardinality of a subset R of nodes of an edge-capacitated digraph $D=(V, A)$ so that, for every node $v \in V - R$, the maximum flow-amount from R to v is at least k and from v to R is at least l. Based on this, they described an algorithm for computing such a minimum set R. The algorithm is polynomial provided that k and l are fixed. (That is, the running time of the algorithm depends

* Corresponding author.
E-mail addresses: barasz@cs.elte.hu (M. Bárász), beckerjc@cs.elte.hu (J. Becker), frank@cs.elte.hu (A. Frank).

[1] Research supported by the Hungarian National Foundation for Scientific Research, OTKA T037547 and by European MCRTN Adonet, Contract Grant No. 504438. A. Frank is also supported by Ericsson Hungary, Laborc u.1, Budapest, Hungary H-1037.

0167-6377/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.orl.2004.07.005

polynomially on the size of D but exponentially on k and l.) Throughout we will refer to this problem as the flow-constrained directed source location (FDSL) problem. To simplify our notation and discussions, we tipically work with the uncapacitated case (when the capacity function is identically one). In this case the maximum flow-amount from s to t is the same as the maximum number of edge-disjoint paths from s to t.

In the present paper, by developing further the ideas of [7], we describe a strongly polynomial algorithm for solving the FDSL problem. Around the same time as we did, van den Heuvel and Johnson [6] also developed a polynomial algorithm based on completely different ideas. For a comparison, see Section 5.

1.1. Preliminaries

Let $D=(V, A)$ be a digraph. For elements $s, t \in V$, a subset $X \subset V$ is called a $t\bar{s}$-set if $t \in X \subseteq V-s$. In D the in-degree $\varrho(X)=\varrho_D(X)$ denotes the number of edges entering X while the out-degree $\delta_D(X)=\delta(X)$ is the number of edges leaving X. If a nonnegative capacity function g is given on the edge set, $\varrho_g(X)$ (respectively, $\delta_g(X)$) denotes the sum of capacity values on the edges entering (resp., leaving) X.

Given nonnegative integers 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*. A deficient set Z is called *minimal* if no proper subset of Z is deficient. The hypergraph of minimal deficient sets will throughout be denoted by H_{kl}.

A digraph is called *(k, l)-edge-connected* with respect to a root node r if there are k edge-disjoint directed paths from r to every other node and there are l edge-disjoint directed paths from every node to r. It follows immediately from the directed edge-version of Menger's theorem that there are k edge-disjoint paths from r to *all* other nodes of D (that is, D is $(k, 0)$-edge-connected) if and only if the in-degree of all nonempty subsets of $V-r$ is at least k, and an analogous characterization holds for $(0, l)$ edge-connectivity. Therefore (k, l)-edge-connectivity of a digraph is equivalent to requiring that the in-degree and out-degree of all nonempty subsets of $V-r$ is at least k and l, respectively. When $k=l$, this notion is equivalent to the 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 a *(k, l)-source* if the contraction of R into a single node r results in a (k, l)-edge-connected digraph with respect to root node r. Equivalently, there are k edge-disjoint directed paths from R to every node and there are l edge-disjoint directed paths from every node to R. Yet another equivalent formulation is that R covers all deficient sets.

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 Z of nodes intersecting each hyperedge is called a *transversal* of H. The smallest cardinality $\tau(H)$ of a transversal is the *transversal number* of H. A hypergraph H is said to admit the *Helly property* or to be 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 every pair of hyperedges X, Y. A *subtree hypergraph* (sometimes called an arboreal hypergraph) is one for which there is a tree T on its node set such that each hyperedge induces a subtree of T. The tree is called a *basic* (or representative) *tree* for the hypergraph.

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 least 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.

An undirected 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. Chordal graphs are known to be perfect, and Gavril [4] constructed an algorithm that computes in a chordal graph a maximum stable set and a minimum clique-covering with the same cardinality.

The following simple theorem was discovered independently by several authors (for references, see [2]).

Theorem 1.1. *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.*

It follows that the matchings of a subtree hypergraph H correspond to the stable sets of $L(H)$ and the transversals of H correspond to the clique-coverings of $L(H)$. Therefore $\nu(H) = \tau(H)$ and Gavril's algorithm may be used to compute a maximum matching and a minimum transversal of H. This algorithm is polynomial in $|\mathcal{E}|$.

1.1.1. Flows and cuts

In later sections we need some basic properties of flows and cuts. For two disjoint nonempty 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 sets 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 family $\{Z : T \subseteq Z \subseteq V - S,\ \varrho(Z) = \lambda(S, T)\}$ of minimizer sets is closed under taking union and intersection. (Indeed, by using the submodularity of ϱ_g one has $\lambda_g(S, T) + \lambda_g(S, T) = \varrho_g(X) + \varrho_g(Y) \geqslant \varrho_g(X \cap Y) + \varrho_g(X \cup Y) \geqslant \lambda_g(S, T) + \lambda_g(S, T)$ 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 an 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, $Z_{\min}$ as well as $Z_{\max}$ 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 return $Z_{\max}$ as a minimizer set. The additional 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.

2. Flow-constrained directed source location (FDSL)

2.1. Known results

As mentioned above, Ito et al. [7] introduced and investigated the FDSL problem. While they pointed out that several closely related problems are NP-complete, they also showed, by proving a fundamental min–max theorem, that FDSL belongs to NP∩co-NP. We formulate the result for the uncapacitated case.

Theorem 2.1 (*Ito et al. [7]*). *The hypergraph H_{kl} of minimal deficient sets form a subtree hypergraph and (hence)*

$$\tau(H_{kl}) = \nu(H_{kl}). \tag{1}$$

The minimum cardinality of a (k, l)-source is equal to the maximum number of pairwise disjoint deficient sets.

What Ito et al. actually proved was that the hypergraph H_{kl} admits the Helly property and its line graph is chordal. This and Theorem 1.1 implied that H_{kl} is a subtree hypergraph from which (1) followed. The second part follows from the first one by observing that in order to cover all deficient sets, it is sufficient to cover only the minimal ones, that is, the hyperedges of H_{kl}. Note that the hypergraph of *all* deficient sets is not necessarily a subtree hypergraph indicating the value of working with the hypergraph of minimal deficient sets.

The main concern of the FDSL problem is to construct an efficient algorithm to compute a (k, l)-source of smallest cardinality. In this respect, Theorem 2.1 may be viewed as a stopping rule since it can equivalently be formulated as follows: *A (k, l)-source R is of minimum cardinality if and only if there is a family $\mathcal{M}$ of $|R|$ pairwise disjoint deficient sets.* Such an $\mathcal{M}$ may serve as a certificate for the minimality of a proposed (k, l)-source R.

By applying Gavril's algorithm to the line graph of H_{kl}, Ito et al. [7] obtained an algorithm that computes both a minimum cardinality (k, l)-source and a maximum matching $\mathcal{M}$ of deficient sets. The running time is polynomial in the size of D and of H_{kl}. Unfortunately, the number of minimal deficient sets may be exponential in $|V|$ even if $l = 0$ as shown by the

following example of [7]. 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.

Therefore the algorithm for the FDSL problem is polynomial in the size of D but not necessarily in k and l. For fixed k and l, however, the number of minimal deficient sets depends polynomially on the size of D 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 [7] outlined above is polynomial for fixed k, l.

2.2. *Strategy of a strongly polynomial algorithm*

The goal of the present work is to describe an algorithm that is polynomial for the FDSL problem in k and l (and strongly polynomial in the capacitated case). A crucial idea is the introduction of the new concept of solid sets (for a definition, see Section 3). This notion depends only on D and not on k and l and it proves to be a fortunate generalization of that of minimal deficient sets. On one hand, Theorem 2.1 nicely extends to solid sets (Theorem 3.3), on the other hand, serious algorithmic difficulties with minimal deficient sets can be overcome by working with solid sets.

Since the line graph of H_{kl} may be exponentially large in the size of D, one must avoid its usage in an algorithm. Ref. [3] described an algorithm for computing a minimum transversal and a maximum matching of an arbitrary subtree hypergraph H that works directly on the basic tree T for H, rather than using the line graph of H.

Therefore we must be able to compute a basic tree for H_{kl} as well as to run the algorithm of [3] in a situation where the hypergraph H_{kl} cannot be given explicitly. This difficulty will be overcome by introducing a 'surrogate' subtree hypergraph H'_D that has just a few hyperedges (at most n^2, by a straightforward estimation, and, in fact, at most $2n - 2$, by a tricky one [1]). H'_D will have the property that any tree basic for H'_D is automatically basic for H_{kl}. Details will be discussed in Section 3. Once a basic tree for H_{kl} is available, the algorithm of [3] can be mimicked with the help of MFMC computations, see Section 4 for details.

3. Computing a basic tree for H_{kl}

In this section we show how to compute a basic tree for H_{kl} without using an explicit list of the hyperedges of H_{kl}.

3.1. *Computing a basic tree for a subtree hypergraph*

How can one compute a basic tree for an arbitrary subtree hypergraph $H = (V, \mathcal{E})$? Although the known inductive proof of Theorem 1.1 may easily be turned into an algorithm that is polynomial in $|\mathcal{E}|$, we outline here another approach, based on the greedy algorithm, for constructing a basic tree. We do so for completeness and because we have not found in the literature this pretty link to the well-known maximum weight spanning tree problem.

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,

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

Theorem 3.1. *A hypergraph H admits a basic tree (that is, H is a subtree hypergraph) if and only if a spanning tree of maximum c-weight 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

$$\begin{aligned} c(T) &:= \sum_{uv \in E(T)} c(uv) \\ &= \sum_{uv \in E(T)} \sum [1 : Z \in \mathcal{E}, \{u, v\} \subseteq Z] \\ &\leqslant \sum_{Z \in \mathcal{E}} (|Z| - 1) \end{aligned} \tag{2}$$

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.

3.2. Solid sets and partitions

In order to introduce the small surrogate hypergraph for H_{kl} promised above, first we need to define a hypergraph that is actually larger than H_{kl}. Given a digraph $D = (V, A)$, we call a nonempty 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 the parameters k and l while that of the solid sets does not. Since an in-solid set Z is a minimal k'-in-deficient set with respect to the parameter $k' := \varrho(Z) + 1$, the set of in-solid sets is exactly the union of all k-in-deficient sets ($k = 1, 2, \ldots$). An analogous statement holds for out-solid and solid sets. In other words, H_D may be viewed as the union of hypergraphs H_{kl} for all possible values of k and l.

We remark that an analogous notion for undirected graphs, under the name of extreme sets, was introduced and successfully used to solve the undirected edge-connectivity augmentation problem by Watanabe and Nakamura [10]. But the structure of extreme sets of undirected graphs, as they are laminar, is much simpler than that of the solid sets of digraphs.

As mentioned, [7] 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 3.2. *If X is in-solid and Y is out-solid, then at least one of the subsets $A := X - Y$, $B := Y - X$, $C := X \cap Y$ 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 $0 > \gamma + \gamma'$ would follow. □

Theorem 3.3. *The hypergraph $H_D = (V, \mathcal{E}_D)$ of solid sets is a subtree hypergraph, that is, for every directed graph $D = (V, A)$ there is a spanning tree on the groundset V such that each solid set of D induces a subtree.*

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 $X_1, \ldots, X_h$ ($h \geqslant 4$) 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 3.2 implies that either all X_i's are in-solid or all X_i's are out-solid. By symmetry, we may assume that the first case occurs. It follows that the h intersections $X_i \cap X_{i+1}$ are pairwise disjoint and hence

$$\sum_{i=1}^{h} \varrho(X_i \cap X_{i+1}) \leqslant \sum_{i=1}^{h} \varrho(X_i). \tag{3}$$

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 (3).

We claim that H_D admits the Helly property. If it does not, then there is a smallest number $h \geqslant 3$ along with h solid sets $X_1, \ldots, X_h$ such that any two of these sets intersect each other while the intersection $M = X_1 \cap \ldots \cap X_h$ is empty. Again, by Lemma 3.2 either the sets $X_1, \ldots, X_h$ are all in-solid or they are 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 $Y_i \cap Y_j = \emptyset$ ($1 \leqslant i < j \leqslant h$). If an edge enters one of the sets Y_i, then it enters at least one of the sets X_j. Therefore $\sum_i \varrho(Y_i) \leqslant \sum_i \varrho(X_i)$. On the other hand $\varrho(Y_i) > \varrho(X_{i+1})$ for each i as X_{i+1} is in-solid and $Y_i \subset X_{i+1}$. Hence $\sum_i \varrho(Y_i) > \sum_i \varrho(X_{i+1}) = \sum_i \varrho(X_i)$, a contradiction.

By Theorem 1.1 H_D is indeed a subtree hypergraph. □

We call a basic tree for H_D a *solid tree* for D. In order to be able to compute a solid tree, we need some further properties of solid sets.

Lemma 3.4. *If the intersection of two in-solid (out-solid) sets X and Y is nonempty, then $X \cup Y$ is in-solid (out-solid).*

Proof. If indirectly $X \cup Y$ is not in-solid, then there is a maximal nonempty subset $Z \subset X \cup Y$ with $\varrho(Z) \leqslant \varrho(X \cup Y)$.

If Z includes one of X and Y, say X, then $Z \cap Y \subset Y$, $X \cup Y = Z \cup Y$ and hence $\varrho(Z \cap Y) > \varrho(Y)$, $\varrho(Z \cup Y) = \varrho(X \cup Y) \geqslant \varrho(Z)$ from which $\varrho(Y) + \varrho(Z) \geqslant \varrho(Z \cap Y) + \varrho(Z \cup Y) > \varrho(Y) + \varrho(Z)$ 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 $\varrho(X) + \varrho(Y) \geqslant \varrho(X \cap Y) + \varrho(X \cup Y) > \varrho(X) + \varrho(X \cup Y)$ implies $\varrho(Y) > \varrho(X \cup Y)$ from which we would have $\varrho(Z) > \varrho(X \cup Y)$, contradicting the assumption $\varrho(Z) \leqslant \varrho(X \cup Y)$. 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 $\varrho(X \cup Z) \geqslant \varrho(Z)$. Therefore we have $\varrho(X) + \varrho(Z) \geqslant \varrho(X \cap Z) + \varrho(X \cup Z) > \varrho(X) + \varrho(Z)$, a contradiction. The proof for out-solid sets is analogous. □

By an *s-avoiding in-solid* (*out-solid*) set Z we mean an in-solid (out-solid) subset of $V - s$. The adjective maximal is used if Z is not included in any other s-avoiding in-solid (out-solid) subset of $V - s$. By Lemma 3.4 the maximal s-avoiding in-solid sets are disjoint. 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 Lemmas 3.4 and 3.2 that:

Corollary 3.5. *The family of maximal s-avoiding solid sets is a partition of* $V - s$.

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

3.3. Computing the solid partition of $V - s$

By Corollary 3.5 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 solid partition of $V - s$, it suffices to compute separately the in-solid and the out-solid partitions 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 in-degree 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.

Lemma 3.6. *If N is a minimal member of the family $\{N_t : t \in V - s\}$, then N is a maximal s-avoiding in-solid set.*

Proof. We claim that $z \in N_t$ implies $N_z \subseteq N_t$ 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 $\varrho(N_z) + \varrho(N_t) \geqslant \varrho(N_z \cup N_t) + \varrho(N_z \cap N_t) > \lambda(t) + \varrho(N_z) \geqslant \varrho(N_t) + \varrho(N_z)$ 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 $\varrho(Z) \geqslant \varrho(N)$ 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 these sets N_t, denoted by N_1. By Lemma 3.6, N_1 is a maximal s-avoiding in-solid set. Second, 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 a 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 in-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 disjoint 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).

3.4. Computing a solid tree for 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. (In fact, Bernáth [1] proved that H'_D has at most $2n - 2$ hyperedges.) Therefore one can compute a basic tree T for H'_D as described in Section 3.1 and this algorithm is polynomial in the size of D. The nice thing is that T will automatically be a basic tree for H_D and hence for H_{kl}, too.

Theorem 3.7. *If T is a basic tree for H'_D, then T is basic for the hypergraph H_D of all solid sets (and, in particular, for its subhypergraph H_{kl} of deficient sets).*

Proof. Suppose indirectly that there is a 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. Ref. [3] describes an algorithm for computing a minimum transversal R and a maximum matching $\mathcal{M}$ of H that works directly on the basic tree T for H. (Actually, that algorithm settles a weighted case as well but here we need only the unweighted version.) First we exhibit and justify the correctness of the generic form of the algorithm where it does not matter how the input hypergraph is given. A more specific version is then described and shown how it 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 $\mathcal{M} \cup \{Z\}$ is a matching.

The generic algorithm starts with the empty matching $\mathcal{M}$. In each step, it chooses any of the highest hyperedges that is independent from the current matching $\mathcal{M}$ and adds it to $\mathcal{M}$. The algorithm terminates when no such hyperedge exists anymore. It returns the final $\mathcal{M}$ and the set R of bottom nodes of the members of $\mathcal{M}$. Clearly, $|\mathcal{M}| = |R|$. The correctness of the algorithm as well as a proof of the min–max relation $\nu(H) = \tau(H)$ for subtree hypergraphs follow from the following lemma.

Lemma 4.1 (*Frank [3]*). *The set R of bottom nodes output by the algorithm outlined above covers all hyperedges.*

Proof. Suppose indirectly that there is a hyperedge Y not covered by R. By the termination rule of the algorithm 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 $\mathcal{M}$ prior 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 'choose-the-highest' rule of the algorithm. □

We describe now more specifically how a highest hyperedge independent from $\mathcal{M}$ can be found. Instead of trying to find it directly, 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 $\mathcal{M}$. If it is, Z is added to $\mathcal{M}$.

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 members of $\mathcal{M}$, the algorithm maintains a label assigned to each node. The content of the label is 'marked' or 'unmarked'.

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 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 set each node 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 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 cannot get marked as long as $A(v) - v$ includes a 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 given. 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 $\lambda(S(v), v) \geqslant k$ and $\lambda(v, S(v)) \geqslant l$, then (by Property (*)) a 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 a hyperedge of H_{kl}.

Claim 4.2. *The 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 3.2 one of them includes the other, hence Z is minimally deficient. If exactly one of $\lambda(S(v), v) < k$ and $\lambda(v, S(v)) < l$ holds, then by Property (*) again, Z is minimal deficient. □

5. Running time and conclusions

In the following estimation of running times we use the notation $n := |V|$, $m := |A|$. The algorithm outlined above for solving the FDSL 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 FDSL problem on the same digraph for several values of k and l, only the third step should be repeated.

Let $F(n, m)$ denote the complexity of an 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 $O(n^2F(n,m))$. Since we need this for all nodes, the total time of Phase 1 is $O(n^3F(n,m))$.

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 $O(nF(n,m))$.

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 [5] invented an $O(nm\log(n^2/m))$ time algorithm to compute the minimum cuts 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 $\lambda_g(s,t)$ in-capacity, 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 $O(n^3m\log(n^2/m))$.

Finally, we remark that the algorithm may be applied to the capacitated case without any change. Since the time bound for the Hao–Orlin algorithm concerns capacitated digraphs anyhow, the complexity bound given before remains valid.

van den Heuvel and Johnson [6] also developed a polynomial algorithm based on completely different ideas. Actually, their algorithm can compute a smallest transversal for *any* subtree hypergraph provided a subroutine is available for deciding whether a subset of nodes is a transversal of H. On the other hand the algorithm does not compute a maximum matching of H_{kl}. The complexity of the algorithm of [6] is $O(n^3S(n))$ where $S(n)$ denotes the complexity of the subroutine, and such a subroutine is indeed available for H_{kl} via a Hao–Orlin computation. Therefore the algorithm of van den Heuvel and Johnson, when specialized to the FDSL problem, is of complexity $O(n^4\, m\, \log(n^2/m))$.

5.2. Conclusion

We developed a strongly polynomial time algorithm for the FDSL problem introduced and analyzed in [7]. A useful feature of our approach is that it can be used to solve the following inverse problem: given the digraph D and an integer 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 Megiddo [9] can be used to show that $k(C)$ can be computed by n applications of our algorithm.

References

[1] A. Bernáth, A note on the directed source location algorithm, Egerváry Research Report, 2004-12.

[2] P. Duchet, Hypergraphs (Theorem 3.8), in: R. Graham, M. Grötschel, L. Lovász (Eds.), Handbook of Combinatorics, Elsevier, Amsterdam, 1995, pp. 381–432.

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

[4] F. Gavril, Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph, SIAM J. Comput. 1 (1972) 180–187.

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

[6] J. van den Heuvel, M. Johnson, Transversals of subtree hypergraphs and the source location problem in digraphs, CDAM Research Report, LSE-CDAM-2004-10.

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

[8] M. Labbé, F.V. Louveaux, Location problems, in: M. Dell' Amico, F. Maffioli, S. Martello (Eds.), Annotated Bibliographies in Combinatorial Optimization, Wiley, 1997, pp. 261–281.

[9] N. Megiddo, Combinatorial optimization with rational objective functions, Math. Oper. Res. 4 (1979) 414–424.

[10] T. Watanabe, A. Nakamura, Edge-connectivity augmentation problems, Comput. Syst. Sci. 35 (1) (1987) 96–144.