Egerváry Research Group
on Combinatorial Optimization

Technical reportS

TR-2006-12. Published by the Egerváry Research Group, Pázmány P. sétány 1/C, H–1117, Budapest, Hungary. Web site: www.cs.elte.hu/egres. ISSN 1587–4451.

Source location in undirected and directed hypergraphs

Attila Bernáth

December 2006

Source location in undirected and directed hypergraphs

Attila Bernáth*

Abstract

In this paper we generalize source location problems with edge-connectivity requirements on undirected and directed graphs to similar problems on undirected and directed hypergraphs. In the undirected case we consider an abstract framework which contains the source location for undirected hypergraphs as a special case and which can be solved in polynomial time. For the directed case, the asymmetry of the results we get reflects the asymmetry of the model of directed hypergraph we consider.

1 Introduction and preliminaries

A wide range of source location problems can be formulated the following way: given a graph (either directed or undirected) find a nonempty subset S of the nodes (called *source set*) such that contracting the set S will result in a (di)graph with "good connectivity properties". The objective is to minimize the total weight of the source set to be found (where nodes have nonnegative weights). Tractability, of course, depends heavily on our definition for "good connectivity properties".

We will mainly concentrate on the following special cases ($\mathbb{R}_\oplus$ denotes the set of nonnegative reals):

Problem 1 (Source Location in Graphs)**.** *Given a graph $G = (V, E)$, a weight function $w : V \to \mathbb{R}_\oplus$ and a requirement function $r : V \to \mathbb{R}_\oplus$ on the nodes (we will also call r as the* demand function*). Find a nonempty subset of nodes S such that for every node $v \in V - S$ there are at least $r(v)$ edge-disjoint paths each starting at v and ending in a node of S and $w(S) := \sum_{s\in S} w(s)$ is minimum.*

This problem was raised in [1] where they show the NP-completeness of this problem. However, it is shown in [1] that the special case when w is constant can be solved in $O(|V|M)$ time, where M is the time needed for one maximum flow computation in our network. A minor observation shows that the same algorithm solves another

*Dept. of Operations Research, Eötvös University, Pázmány P. s. 1/C, Budapest, Hungary H-1117. The author is a member of the Egerváry Research Group (EGRES). Research supported by European MCRTN Adonet, Contract Grant No. 504438, by OTKA grant K 060802 and by the Mobile Innovation Centre, Hungary. E-mail: bernath@cs.elte.hu

special case, namely when r is constant, but the authors of [1] give another algorithm for this case that improves the running time to $O(|V|(|E| + |V| \log |V|))$.

To introduce the problems we want to consider for directed graphs, we need a definition.

Definition 1. *Given a digraph $D = (V, A)$, a nonempty subset of the nodes S and a node $v \in V - S$, then $\lambda_D(S, v)$ denotes the maximum number of arc-disjoint directed paths, each starting at a vertex of S and ending at v. Similarly, $\lambda_D(v, S)$ denotes the maximum number of arc-disjoint directed paths starting at v and ending at a vertex of S. If $u, v \in V$ are distinct vertices then we write $\lambda_D(u, v)$ instead of $\lambda_D(\{u\}, v)$. If no confusion can arise then we will leave out D from the subscript. A digraph is called* (k, l)-arc-connected *from node s if $\lambda(s, v) \geq k$ and $\lambda(v, s) \geq l$ for every $v \in V - \{s\}$.*

Problem 2 (Source Location in Digraphs)**.** *Given a directed graph $D = (V, A)$ and nonnegative integers k, l, find a nonempty subset S of the nodes such that $\lambda_D(S, v) \geq k$ and $\lambda_D(v, S) \geq l$ for every node $v \in V - S$ and $|S|$ is minimum. (In other words: find a smallest possible nonempty set S such that contracting S to a node s results in a (k, l)-arc-connected graph from s).*

This problem was raised in [8] where it was shown that it lies in $NP \cap \text{co}(NP)$. Later, Bárász, Becker and Frank in [3] gave a strongly polynomial time algorithm that solves this problem. We have to mention that the weighted version of this problem is NP-hard, as was shown in [8].

In this paper we generalize the above given source location problems for hypergraphs and directed hypergraphs. We should note that the above given references actually solve the capacitated versions of the above given problems (that is, edges or arcs have nonnegative capacities and we consider flow value instead of edge- or arc-connectivity), but this only means a minor technical difficulty in both cases.

Let us conclude this section with some more notations and definitions. For general graph-theoretic terms we refer the reader to [14].

Definition 2. *A set-function $b : 2^V \to \mathbb{R}$ is said to be* submodular *if*

$$b(X) + b(Y) \geq b(X \cap Y) + b(X \cup Y) \ \forall X, Y \subseteq V.$$

It is called symmetric *if*

$$b(X) = b(V - X) \text{ for any } X \subseteq V.$$

It is called posi-modular *if*

$$b(X) + b(Y) \geq b(X - Y) + b(Y - X) \ \forall X, Y \subseteq V.$$

Note that a symmetric submodular function is also posi-modular.

Notations: For nodes $s, t \in V$ we will say that $X \subseteq V$ is an $\overline{s}t$*-set* to mean that $s \notin X, t \in X$. If we say that a set $X \subseteq V$ is *minimal* to some property then we mean that there is no set $Y \subsetneq X$ with this property. The term *maximal* will be used in the same sense. We will usually use $n = |V|$. If $S \cap X \neq \emptyset$ for two sets S, X then we will also say that S *covers* X. A set containing exactly one point will be called a *singleton* sometimes.

1.1 Preliminaries on hypergraphs

An undirected hypergraph (or shortly hypergraph) $H = (V, \mathcal{E})$ is a pair of a finite set V and a family $\mathcal{E}$ of subsets of V (repetitions are allowed). The set V is called the *node set* of the hypergraph, the family $\mathcal{E}$ is called the *edge set* of the hypergraph. An element of $\mathcal{E}$ will be called a *hyperedge*.

In a hypergraph H, a *path* between nodes s and t is an alternating sequence of distinct nodes and hyperedges $s = v_0, e_1, v_1, e_2, \ldots, e_k, v_k = t$, such that $v_{i-1}, v_i \in e_i$ for all i between 1 and k. H is connected if there is a path between any two distinct nodes. A hyperedge e enters a set X if $e \cap X \neq \emptyset$ and $e \cap (V - X) \neq \emptyset$. For a set X we define $d_H(X) = |\{e \in \mathcal{E} : e \text{ enters } X\}|$ (the *degree* of set X in H). This is a symmetric submodular function.

Definition 3. *Given a hypergraph $H = (V, \mathcal{E})$, a set $S \subseteq V$ and node $x \in V$. Let $\lambda_H(x, S)$ denote the maximum number of edge-disjoint paths starting at x and ending at a vertex of S (we agree that $\lambda_H(x, S) = \infty$ if $x \in S$). We will write $\lambda_H(x, y)$ instead of $\lambda_H(x, \{y\})$, if $x, y \in V$. Subscript H may be omitted if no confusion can arise.*

It is well known that Menger's theorem can be generalized for hypergraphs:

Theorem 1.1. *Let $H = (V, \mathcal{E})$ be a hypergraph, and $s, t \in V$ distinct nodes. Then*

$$\lambda_H(x, y) = \min\{d_H(X) : X \subseteq V \text{ is an } \overline{st}\text{-set}\}.$$

Definition 4. *For any nonnegative real number r define the binary relation $\sim_r$ as follows*

$$x \sim_r y \iff \lambda_H(x, y) \geq r.$$

It is easy to see that this is an equivalence relation. The equivalence classes are called *r-connectivity components*.

1.2 Preliminaries on directed hypergraphs

We will use the notion of directed hypergraphs (or *dypergraphs*, for short) as it is introduced in [5].

A directed hypergraph $H = (V, \mathcal{A})$ is a pair of a finite set V and a family $\mathcal{A}$ of subsets of V (repetitions are allowed) where each member a of $\mathcal{A}$ contains a designated *head* node, denoted by $h(a)$. The rest of the elements of a are called the *tails* and we introduce the notation $T(a) = a - h(a)$ for the tail set of a (we will assume that $|a| \geq 2$ for each $a \in \mathcal{A}$). The set V is called the *node set* of the dypergraph, the family $\mathcal{A}$ is called the *hyperarc set* (or sometimes shortly the *arc set*) of the dypergraph. An element of $\mathcal{A}$ will be called a *hyperarc*.

In a dypergraph H, a *path* between nodes s and t is an alternating sequence of distinct nodes and hyperarcs $s = v_0, a_1, v_1, a_2, \ldots, a_k, v_k = t$, such that $v_{i-1} \in T(a_i)$ and $v_i = h(a_i)$ for all i between 1 and k. A hyperarc a enters a set X if $h(a) \in X$ and $T(a) \cap (V - X) \neq \emptyset$. A hyperarc leaves a set if it enters the complement of this set. For a set X we define $\varrho_H(X) = |\{a \in \mathcal{A} : a \text{ enters } X\}|$ (the *in-degree* of X) and $\delta_H(X) = \varrho_H(V - X)$ (the *out-degree* of X). It is easy to check that functions ϱ and δ are submodular functions.

Definition 5. *Given a dypergraph $H = (V, \mathcal{A})$ and nodes $x, y \in V$ the maximum number of arc-disjoint paths from x to y in H will be denoted by $\lambda_H(x, y)$. For a set $S \subseteq V$ and node $x \in V - S$, $\lambda_H(S, x)$ denotes the maximum number of arc-disjoint paths starting at a vertex of S and ending at x ($\lambda_H(x, S)$ is defined analogously). Subscript H may be omitted if no confusion can arise.*

Again it is well known that Menger's theorem can be generalized for dypergraphs:

Theorem 1.2. *Let $H = (V, \mathcal{A})$ be a directed hypergraph, and $s, t \in V$ distinct nodes. The maximum number of arc-disjoint directed paths from s to t is*

$$\min\{\varrho_H(X) : X \subseteq V \text{ is an } \overline{st}\text{-set}\}.$$

2 Source location in undirected hypergraphs

In this section we introduce a generalization of Problem 1 for (undirected) hypergraphs. Then we generalize the problem further and formulate an abstract source location problem. Of course we can not solve this problem in general (it contains an NP-complete problem), but we observe that it can be solved with an adaptation of the greedy algorithm introduced in [1] in the special case when the demand and the weight functions are *compatible*. This contains the case when either of these functions is uniform (constant, in other words), but in the subsequent subsection we give another algorithm for uniform demand functions that gives a better running time. In every case we also specialize our results for the hypergraphic source location problem, too.

A straightforward generalization of Problem 1 introduced above is the following:

Problem 3 (Source Location in Hypergraphs)**.** *Given an undirected hypergraph $H = (V, \mathcal{E})$, a weight function $w : V \to \mathbb{R}_\oplus$ and a demand function $r : V \to \mathbb{R}_\oplus$ on the nodes. Find a nonempty subset of nodes S such that for every node $v \in V - S$ there are at least $r(v)$ edge-disjoint paths each starting at v and ending in a node of S and $w(S) := \sum_{s \in S} w(s)$ is minimum.*

By Theorem 1.1 we can reformulate the problem in the following way: find a subset of nodes S such that $d_H(X) \geq \max\{r(v) : v \in X\}$ for every $\emptyset \neq X \subseteq V - S$ and $w(S)$ is minimum. However, since Problem 1 is a special case of this problem, this is an NP-complete problem. In what follows we will concentrate on the two special cases of this problem that were solvable for graphs; namely the constant weight and the constant demand case. We will show that the algorithms given in [1] can be modified appropriately to work in this more general setting, too. What is more, we will show that they work in an even more abstract environment, too. To this end let us introduce the following, abstract form of Problem 3.

Problem 4 (Abstract Source Location)**.** *Given a function $d : 2^V \to \mathbb{R}_\oplus$ that is posimodular and submodular, a weight function $w : V \to \mathbb{R}_\oplus$ and a demand function $r : V \to \mathbb{R}_\oplus$ on the nodes. Find a subset of nodes S such that $d(X) \geq \max\{r(v) : v \in X\}$ for every $X \subseteq V - S$ and $w(S)$ is minimum.*

As a remark we mention that the nonnegativity of the function d is not really a restriction: an arbitrary symmetric submodular function d' attains its minimum at the empty set, so defining $d(X) = d'(X) - d'(\emptyset)$ gives a nonnegative symmetric submodular function and we can modify the demand function similarly to be able to reduce the more general problem without the nonnegativity constraints to our problem. On the other hand, the nonnegativity of the weight function is a natural requirement, since any superset of a valid source set is again a valid source set. Also, we could have required the nonemptyness of a source set: that would not have changed the problem significantly.

In the following sections we will show that the special case of this abstract problem when the demand and the weight functions are compatible can be solved using appropriate adaptations of the algorithms given in [1].

Let us introduce some terminology. A set $X \subseteq V$ is called *deficient* if $d(X) < \max\{r(v) : v \in X\}$ (if $X = \emptyset$ then $\max\{r(v) : v \in X\} = -\infty$, so the empty set will never be deficient). It is obvious that a set S is a valid source set if and only if it meets every deficient set, which is equivalent to requiring that S has to meet every *minimal deficient set*.

2.1 Compatible demand- and weight function

In this section we will give an algorithm to solve Problem 4 in the special case when the demand- and weight functions are compatible. This is an adaptation of the greedy algorithm given in [1]. Let us define first what we mean by these functions being compatible.

Definition 6. *Two functions $r, w : V \to \mathbb{R}$ will be called* compatible *if there is an ordering $v_1, v_2, \ldots, v_n$ of V such that $r(v_1) \leq r(v_2) \leq \cdots \leq r(v_n)$ and $w(v_1) \geq w(v_2) \geq \cdots \geq w(v_n)$.*

Observe that compatibility of two functions can be easily checked in $O(n \log n)$ time and it is indeed a symmetric relation. Now we can give the greedy algorithm to solve Problem 4: in this algorithm we only assume the *posi-modularity* of function d but we leave one question open that will only be answered when d is also submodular!

Algorithm GREEDY
begin
INPUT A *posi-modular* function $d : 2^V \to \mathbb{R}_\oplus$, a demand function $r : V \to \mathbb{R}_\oplus$ and a weight function $w : V \to \mathbb{R}_\oplus$ on the finite set V. Assume that r and w are compatible functions.
OUTPUT A minimum weight source set S.
1.1. Let $S = V$. Order V s.t. $r(v_1) \leq r(v_2) \leq \cdots \leq r(v_n)$ and $w(v_1) \geq w(v_2) \geq \cdots \geq w(v_n)$.
1.2. For $i = 1$ to n do
1.3. (*) If there is no deficient set X with $S \cap X = \{v_i\}$ then $S := S - v_i$.
1.4. Output S.
end

One thing remains to make clear: how to implement Step (*); to be able to do this we will also assume the submodularity of the function d. Let us speak about this later and first check the correctness of the algorithm. Let us denote the current set S in the ith iteration *before* executing step (*) by S_i (so $S_1 = V$) and let the output of the algorithm be $S_{n+1} = \{v_{i_1}, v_{i_2}, \ldots, v_{i_t}\}$ for some $t \geq 0$. A simple inductive argument shows that S_i is a valid source set for any i between 1 and $n+1$. We only need to show that S_{n+1} has minimum weight among the source sets. If, in the ith step of the for loop, $S_i - v_i$ is not a valid source set then there must be a deficient set X_i with $S_i \cap X_i = \{v_i\}$. We can assume that this X_i is minimal deficient: there is a minimal deficient set in X_i but it must contain v_i otherwise S_i was not a valid source set. In particular $X_i \subseteq \{v_1, v_2, \ldots, v_i\}$ which also implies that $\max\{r(v) : v \in X_i\} = r(v_i)$. We will show that these sets $X_{i_1}, X_{i_2}, \ldots, X_{i_t}$ are pairwise disjoint minimal deficient sets. Assume that this is not the case and suppose $i_j < i_k$ are such that $X_{i_j} \cap X_{i_k} \neq \emptyset$. It is clear that $v_{i_k} \notin X_{i_j}$ because $X_{i_j} \subseteq \{v_1, v_2, \ldots, v_{i_j}\}$. But $v_{i_j} \in X_{i_k}$ can not hold either since in the i_kth iteration X_{i_k} is only covered by v_{i_k} from among the nodes of S_{i_k} which also contains v_{i_j}. But then we have a contradiction from

$$r(v_{i_j}) + r(v_{i_k}) > d(X_{i_j}) + d(X_{i_k}) \geq d(X_{i_j} - X_{i_k}) + d(X_{i_k} - X_{i_j}) \geq r(v_{i_j}) + r(v_{i_k}).$$

(The first inequality follows from the deficiency of sets X_{i_j} and X_{i_k} and the last is because they are minimal deficient sets.) From these observations the optimality of the algorithm follows: we have given an optimal covering of the disjoint minimal deficient sets $X_{i_1}, X_{i_2}, \ldots, X_{i_t}$.

2.2 Implementation of Step (*)

The only thing to show is the subroutine that implements Step (*). We note that a subroutine checking whether a given set S is a valid source set or not would also suffice but the one we gave is easier to implement in our case (and also gives a better running time). It is an open question already mentioned in [13] whether this can be done for a posi-modular set function d: so we also assume that d is submodular. Then we define the following set function $d' : 2^{V-S} \to \mathbb{R}$ with $d'(X) := d(X + v_i)$ for any $X \subseteq V - S$. This is a submodular function and $S - v_i$ is a valid source set if and only if $\min\{d'(X) : X \subseteq V - S\} \geq r(v_i)$. So any algorithm for submodular function minimization will solve this problem. This yields the following result:

Theorem 2.1. *Algorithm GREEDY solves the Abstract Source Location Problem with compatible demand and weight function in $O(n\,\mathrm{SFM}(n,\gamma))$ time, where $\mathrm{SFM}(n,\gamma)$ denotes the time needed to solve submodular function minimization on an n element ground set if γ denotes the time of a call to the function-evaluation oracle.*

We mention that the dual solution (disjoint deficient sets) can be obtained, since algorithms for submodular function minimization can produce these, too. We also point out, that deciding whether a submodular function takes a value less than a given bound B can be formulated as Problem 4 with uniform weight function (following the ideas of Narayanan [11], for example), so any algorithm solving Problem 4 with uniform weight function will need essentially at least $\mathrm{SFM}(n,\gamma)$ time.

The following corollary is a direct extension of Theorems 1 and 2 of [1] and it improves the result (Theorem 24) in [13] by a factor of n:

Corollary 2.2. *Problem 4 with a uniform weight function (and arbitrary demands) can be solved in $O(n\,\mathrm{SFM}(n,\gamma))$ time.*

Another corollary of Theorem 2.1 is that Problem 4 with a uniform demand function (and arbitrary weights) can be solved in $O(n\,\mathrm{SFM}(n,\gamma))$ time. However, in Section 2.3 we will improve this running time for this special case.

We can specialize our algorithm to the hypergraphic source location problem: in this case Step (*) can be regarded as a minimum $(S - v_i, v_i)$ cut problem in our hypergraph $H = (V, \mathcal{E})$ that can be translated to a maximum-flow problem in a graph that has $O(n + |\mathcal{E}|)$ nodes and $O(||\mathcal{E}||)$ edges, where $||\mathcal{E}||$ denotes the sum of the cardinalities of the hyperedges. This yields the following result:

Theorem 2.3. *Algorithm GREEDY solves the Source Location Problem in Hypergraphs with compatible demand and weight function in $O(nM(n + |\mathcal{E}|, ||\mathcal{E}||))$ time, where $M(n', m')$ denotes the time needed for a maximum-flow computation in a graph with n' nodes and m' edges.*

2.3 Improving the running time for uniform demands

In this section we will give an algorithm to solve Problem 4 in the special case when the demand function is constant, that is $r(v) = k\ \forall v \in V$ with some $k \in \mathbb{R}_{\oplus}$. This is an adaptation of the algorithm solving a special case of this problem that was given in [1]. Let us make one simple observation: minimal deficient sets are disjoint in this case. This is proved by the following argument. Suppose X and Y are two such sets with $X \cap Y \neq \emptyset$. Since they can not contain each other, $\emptyset \neq X - Y \subsetneq X$ and $\emptyset \neq Y - X \subsetneq Y$ which gives a contradiction together with

$$k + k > d(X) + d(Y) \geq d(X - Y) + d(Y - X) \geq k + k.$$

(The first inequality follows from deficiency and the last from minimality.)

In [12] Queyranne introduces the notion of a *pendent pair* and gives an algorithm to find a pendent pair. The algorithm computes an ordering of the nodes (that will be called Max-Adj ordering here) and finds the pendent pair using this ordering. This Max-Adj ordering is nothing else but the appropriate generalization of the ordering used in the algorithm by Nagamochi and Ibaraki to find a minimum cut in a graph [10]. Let us give the definitions and results.

Definition 7. *If $h : 2^V \to \mathbb{R}$ is a symmetric submodular function then an ordering $(v_1, v_2, \ldots, v_n)$ is a* Max-Adj *ordering for this function if it satisfies the following:*

$$h(\{v_i\}) - h(\{v_1, \ldots, v_i\}) \geq h(\{v_j\}) - h(\{v_1, \ldots, v_{i-1}, v_j\})\ \forall 2 \leq i \leq j \leq n.$$

We note that the first element v_1 of such an ordering can be specified arbitrarily.

Definition 8. *If $h : 2^V \to \mathbb{R}$ is a symmetric submodular function then an ordered pair of nodes (s,t) is called a* pendent pair *if*

$$h(\{t\}) = \min\{h(X) : X \subseteq V, s \notin X, t \in X\}.$$

Lemma 2.4 ([12])**.** *If $(v_1, v_2, \ldots, v_n)$ is a* Max-Adj *ordering for the function h then (v_{n-1}, v_n) is a pendent pair. Such an ordering can be calculated with $O(n^2)$ calls to the h-value oracle and $O(n^2)$ other operations.*

After these preliminaries we can present the algorithm that solves Problem 4 when the demand function is uniform. Let us first give a sketch of this algorithm.

We want to calculate a minimum weight source set S and we start with $S = \emptyset$. The main idea is the following: the minimal deficient sets form a subpartition of V. In each step of the algorithm we find a suitable pair of nodes that is not separated by a (yet uncovered) minimal deficient set. Then we contract this pair of nodes, and repeat this until one minimal deficient set becomes a singleton. Then we notice this and include an element of this set in S. But what shall we do with the minimal deficient sets that are already covered? We should somehow increase the value of function d on sets that contain such a set: this is best described using an auxiliary graph and its cut function.

After a certain number of such contractions every node in the resulting set is the image of a contracted set in the original ground set: for every such contracted set we remember the "cheapest" element in it (i.e. the one with the smallest weight: this is done with the ancestor function in the algorithm below) so this way when we contract a minimal deficient set into a singleton we can pick the cheapest element from it to include in our set S. The correct formulation of the algorithm is described by the succeeding pseudocode.

Algorithm CONSTANT_DEMAND
begin

INPUT A posi-modular and submodular function $d : 2^V \to \mathbb{R}_\oplus$, a demand $k \in \mathbb{R}_\oplus$ and a weight function $w : V \to \mathbb{R}_\oplus$ on the finite set V.
OUTPUT A minimum weight source set S.

1.1. Let $S = \emptyset$, $V' = V + s$ with a new node s and $G = (V', E)$ an auxiliary graph: in the beginning E is empty but later we add edges in E (one endpoint of these will always be s). Define the symmetric submodular function $h : 2^{V'} \to \mathbb{R}$ by $h(X) = d(X) + kd_G(X)$ for any $X \subseteq V$ and $h(X) = h(V' - X)$ if $s \in X \subseteq V'$.
1.2. For each $v \in V$
1.3. $\quad$ ancestor$(v) := v$.
1.4. $\quad$ If $d(\{v\}) < k$ then $S := S + v$ and add an edge between s and v to G (so function h changes here!).
1.5. End for.
1.6. While $|V'| > 2$ and $\exists v \in V' - s$ with $(s, v) \notin E$
1.7. $\quad$ Construct the Max-Adj ordering of V' for function h starting with s and take the last two elements in this ordering (a pendent pair) $(v_{n'-1}, v_{n'})$.

1.8. Contract $v_{n'-1}$ and $v_{n'}$ into a node v' and let the ancestor of v' be the cheaper of the ancestor of $v_{n'-1}$ and that of $v_{n'}$ (so V' gets smaller: function h and graph G follow these changes the obvious way).
1.9. If $h(\{v'\}) < k$ then add an edge between s and v' to G and $S := S + \mathrm{ancestor}(v')$ (function h changes, too!).
1.10. End while.
1.11. Output S.
end

Let us prove the correctness of the above algorithm. In the argumentation below we will think of the set V as the original ground set, while set V' always means the current ground set after some contractions (note that s is never contracted with any other node). Also, for a set $X \subseteq V$ we will use the notation X' for the image of this set after the contractions made so far: this only makes sense if there is no set $Y \subseteq V$ with $X \cap Y \neq \emptyset$ and $(V - X) \cap Y \neq \emptyset$ that was contracted.

Lemma 2.5. *In Step 1.7 of the above algorithm*

(i) if $X \subseteq V$ is a minimal deficient set that is not covered yet (i.e. $S \cap X = \emptyset$ with the current S) then the image X' of X makes sense, and

(ii) there is no (yet uncovered) minimal deficient set $X \subseteq V$ for which $|X' \cap \{v_{n'-1}, v_{n'}\}| = 1$.

Proof. Initially (i) is true. At any stage, if (i) is true, then the statement of (ii) makes sense: assume it is not true and there is a minimal deficient set $X \subseteq V$ for which X' separates $v_{n'-1}$ and $v_{n'}$. But then $d(X') = h(X')$ since X is not covered yet by S and

$$k > d(X) = d(X') = h(X') \geq h(v_{n'}) \geq k,$$

gives a contradiction (the second inequality follows from the properties of the MAX-ADJ ordering; for the last inequality observe that we always assure in the algorithm that $h(v) \geq k$ for singletons $v \in V' - s$). Finally, if (ii) is true then (i) will be true in the next iteration, after contracting $v_{n'-1}$ and $v_{n'}$, too. □

Lemma 2.6. *Algorithm CONSTANT_DEMAND solves Problem 4 with uniform demand function in $O(n^3\gamma + n^3)$ time (where γ denotes the time of a call to the d-value oracle).*

Proof. The correctness of the algorithm follows from Lemma 2.5 and the definition of the ancestor function. It is straightforward to check that Step 1.7 dominates the algorithm which gives the running time indicated above. □

If we specialize our algorithm to hypergraphs, that is to Problem 3, then we can substitute Step 1.7 of our algorithm with a similar subroutine described in [9]. This yields the following running time:

Lemma 2.7. *Algorithm CONSTANT_DEMAND solves Problem 3 with uniform demand function in $O(n^2 \log(n) + n||\mathcal{E}||)$ time ($||\mathcal{E}||$ denotes the sum of the cardinalities of the hyperedges).*

3 Source location in directed hypergraphs

We consider the following problem in dypergraphs.

Problem 5. *Given a directed hypergraph $H = (V, \mathcal{A})$ and nonnegative integers k, l, find a subset S of nodes such that for every node $v \in V - S$*

$$\lambda_H(S, v) \geq k \text{ and } \lambda_H(v, S) \geq l$$

and $|S|$ is minimum.

We will call this problem the (k, l) *source location problem in dypergraphs*. We will show that this problem can be solved in polynomial time if $l \leq 1$ and k is arbitrary, but it is NP-complete if l is arbitrary. This asymmetry can be attributed to the asymmetric definition of directed hypergraphs. We have to mention that one can prove similar results for a capacitated version of Problem 5, too, but we do not include this here.

Because of Theorem 1.2 we can obtain similar reformulations for this problem as for undirected hypergraphs. We define the notion of deficient set appropriately: a nonempty set $X \subseteq V$ is deficient if $\varrho(X) < k$ or $\delta(X) < l$. With this definition a valid source set is a set that meets every deficient set (observe that apart from the uninteresting $k = l = 0$ case a valid source set can never be empty: from now on we will assume that this is not the case). Fortunately we can lead back this problem to the digraph case if $l \leq 1$.

3.1 Solution in the case $l \leq 1$

For a directed hypergraph $H = (V, \mathcal{A})$ consider the following digraph $D_H = (V \cup V_{\mathcal{A}}, A)$: for each hyperarc $a \in \mathcal{A}$ introduce a new node v_a, the new nodes will form the set $V_{\mathcal{A}}$ (disjoint from V). The arc set A contains the following arcs: for each hyperarc $a \in \mathcal{A}$ draw an arc from v_a to the head of the hyperarc a and from each tail of a draw $k' = \max(k, l)$ parallel arcs from this tail to v_a (see Figure 1).

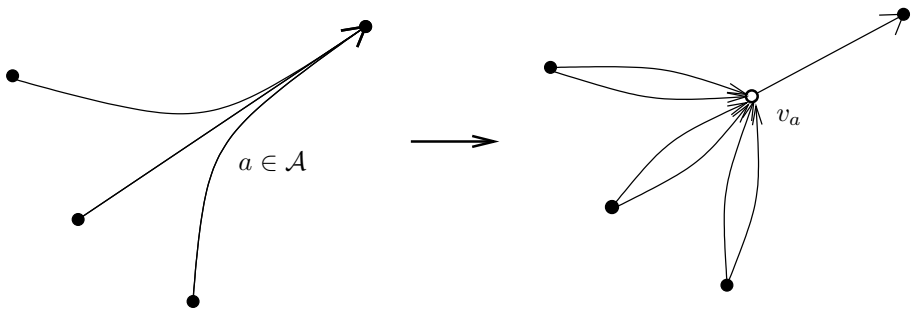

Figure 1: Constructing the digraph from the dypergraph ($k' = 2$)

We will solve the source location in D_H and obtain a solution for H. First observe the following: if $S \subseteq V \cup V_{\mathcal{A}}$ is a source set in D_H then we can easily obtain a source

set which lies already in V and is not bigger than S. Really, if $v_a \in S$ for some $a \in \mathcal{A}$ then do the following: if $l = 0$ then call function SIMPLE_REPLACE(S, a), if $l = 1$ then call function TRICKY_REPLACE(S, a). See the description of these functions below.

Function SIMPLE_REPLACE(S, a)
begin
1.1. if $T(a) \cap S = \emptyset$ then let $S = S - v_a + t$ for some $t \in T(a)$ otherwise let $S = S - v_a$
end

Function TRICKY_REPLACE(S, a)
begin
1.1. if $h(a) \in S$ then SIMPLE_REPLACE(S, v_a)
1.2. else (there is a directed path from $h(a)$ to S, since S is a source-set)
1.3. if there is a directed path from $h(a)$ to $S - v_a$ then
SIMPLE_REPLACE(S, v_a)
1.4. else: the last-but-one element of the directed path from $h(a)$ to S is some $t \in T(a)$, let $S = S - v_a + t$
end

It is an easy exercise to check that S will remain a (k, l)-source set after this operation.

This observation together with the following lemma gives a solution to our source location problem in directed hypergraphs.

Lemma 3.1. *For a nonempty $S \subseteq V$ and nonnegative integers k, l with $l \leq 1$*

$$S \text{ is a } (k,l) \text{ source set for } D_H \iff S \text{ is a } (k,l) \text{ source set for } H.$$

Proof. $\Rightarrow$ If $v \in V - S$ then in D_H there exist k edge-disjoint directed paths from S to v: these correspond to edge-disjoint directed paths in H. Similarly, there are l edge-disjoint directed paths from v to S.

$\Leftarrow$ For nodes in $V - S$ it is the same as before. For $v_a \in V_{\mathcal{A}}$ we have k edge-disjoint paths from S to one of its tails, that can be completed, and similarly we have l edge-disjoint paths to S from $h(a)$ which can be completed. □

3.2 NP-completeness of the case when l is not fixed

We will prove that the (k, l) source location problem in dypergraphs is NP-complete if l is not fixed. To be more precise formulate it as a decision-problem.

Problem 6. *:* DypergraphSourceLocation
Instance*: A dypergraph $H = (V, \mathcal{A})$, nonnegative integers k, l and a positive integer B.*
Question*: Is there a nonempty set $S \subseteq V$ with $\lambda(S, v) \geq k$ and $\lambda(v, S) \geq l$ for all $v \in V - S$ satisfying $|S| \leq B$.*

Theorem 3.2. *The problem* DypergraphSourceLocation *is NP-complete if l is not fixed, even if $k = 0$.*

Proof. The problem is clearly in NP, since given a set S it can be checked in polynomial time whether it satisfies the requirements or not.

To prove completeness we will give a reduction from Vertex Cover(see [7], Problem GT1). An instance of the Vertex Cover problem consists of a graph $G = (U, E)$ and a positive integer K and the question is whether one can find a vertex set of cardinality at most K that covers all the edges of G. For such an instance let $V = U \cup U_E$ be the ground set of our dypergraph, where $U_E = \{x_e : e \in E\}$ is disjoint from U. The arc set will consist of 2 parts $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$. $\mathcal{A}_1$ contains for all $e \in E$ a hyperarc with head x_e and tail set $V - x_e$ (*big hyperarcs*). $\mathcal{A}_2$ will only contain digraph arcs: for all $e = (uv) \in E$ include the arcs (x_e, u) and (x_e, v) into $\mathcal{A}_2$. Finally let $B = K$, $k = 0$ and $l = |E|$.

We claim that minimal deficient sets are exactly the sets of form $\{x_e, u, v\}$ for all $e = (u, v) \in E$. Well, these have out-degree $|E| - 1$, proper subsets of them have bigger out-degree, so they are minimal deficient sets.

Take a deficient set $X \subseteq V$. We claim that X contains $\{x_e, u, v\}$ for some $e = (u, v) \in E$. Assume not and let $|X \cap U_E| = t$. Then there are exactly $|E| - t$ big hyperarcs leaving X. But for every $x_e \in X \cap U_E$ one of the arcs (x_e, u) or (x_e, v) must leave X (where $e = (u, v)$), because X does not contain $\{x_e, u, v\}$. But then X has out-degree at least $|E| - t + t = |E|$, so it is not deficient.

The equivalence of the two problems is straightforward: we only need to notice that an optimal solution to the problem DypergraphSourceLocation can easily be made disjoint from U_E. Finally, we can easily modify the construction to show that Problem 6 is NP-complete if k is greater than zero, too. □

4 Tree- and in-branching packing requirements

In this section we consider some other related source-location type problems that are tractable in (di)graphs, but turn out to be NP-complete in hypergraphs (dypergraphs). The first problem for graphs was considered in [4].

Problem 7 (Source Location with Tree-Packing Requirements)**.** *Given a hypergraph $H = (V, \mathcal{E})$ and a positive integer k. Find a nonempty subset S of the nodes such that $|S|$ is minimum and contracting S results in a hypergraph that has k edge-disjoint connected spanning sub-hypergraphs.*

Theorem 4.1 (Fekete, [4])**.** *The source-location with tree-packing requirements is solvable for graphs in polynomial time if $k \leq 2$ but is NP-complete for $k = 3$.* □

Not surprisingly this problem is already NP-complete for $k = 2$ in hypergraphs as can be seen as an easy consequence of the following theorem (of course it is solvable for $k = 1$).

Theorem 4.2 (Frank, Király, Kriesell, [6])**.** *The problem whether a hypergraph can be decomposed into k connected sub-hypergraphs is NP-complete for every integer $k \geq 2$.* □

Using these results and the techniques used in [2] we can show NP-completeness of source-location problems with *in-branching* packing requirements. Let us give the definitions.

Definition 9. *Let $H = (V, \mathcal{A})$ be a dypergraph and $s \in V$ be a vertex. An* out-branching *(*in-branching*) rooted at s is a collection of hyperarcs $\mathcal{A}' \subseteq \mathcal{A}$ with the property that in the sub-dypergraph $H = (V, \mathcal{A}')$ for every $t \in V - s$ there exists a path from s to t (from t to s, respectively) but if we leave out hyperarcs from $\mathcal{A}'$ then this is no longer true.*

Because of the asymmetric definition of dypergraphs we have strange phenomena. For example it can be checked directly that an out-branching always contains exactly $n - 1$ hyperarcs (where $|V| = n$) with the property that every vertex except for s is the head of exactly one hyperarc. However an in-branching can consist of only one hyperarc. Similarly, Edmonds' arborescence-packing theorem extends to out-branchings while it does not extend to in-branchings (see [2] for a counterexample).

Theorem 4.3 (Edmonds' Out-Branching Theorem for Dypergraphs [5])**.** *A dypergraph $H = (V, \mathcal{A})$ has k arc-disjoint out-branchings rooted at $s \in V$ if and only if*

$$\varrho_H(X) \geq k \text{ for all } \emptyset \neq X \subseteq V - s.$$

□

Because of this theorem the source location problem in dypergraphs with out-branching packing requirements is the same as the $(k, 0)$ source location problem defined and solved in section 3. However the following problem is different from the $(0, l)$ source-location problem: it is even harder.

Problem 8 (Source Location with In-Branching Packing Requirements)**.** *Given a dypergraph $H = (V, \mathcal{A})$ and a positive integer l. Find a nonempty subset S of the nodes such that $|S|$ is minimum and contracting S into a node s results in a dypergraph that has l arc-disjoint in-branchings with root s.*

So while we haven't settled the status of the $(0, l)$ source location problem in dypergraphs if l is fixed, we will show that the above problem is NP-hard even for $l = 2$: it is an easy consequence of the theorem below.

Theorem 4.4. *Given a dypergraph $H = (V, \mathcal{A})$ and a vertex $s \in V$. Then it is an NP-complete problem to decide whether H contains two arc-disjoint in-branchings with root s or not.*

Proof. The construction is almost the same as in the proof of Theorem 4.2 for $k = 2$ in [6]. Consider the following NP-complete problem (see [6] for a justification): given a hypergraph $H' = (V', \mathcal{E}')$, decide whether the hyperedges can be coloured with two

colours, say blue and red, such that both the blue hyperedges and the red hyperedges cover V'. Let $V = V + s$ with a new node $s \notin V'$ and create $H = (V, \mathcal{A})$ as follows: for every hyperedge $e \in \mathcal{E}'$ put a hyperarc $a_e = e \cup \{s\}$ with head s in $\mathcal{A}$. It is easy to check that the hyperedges of H' can be coloured the desired way if and only if H contains two arc-disjoint in-branchings with root s. □

Acknowledgements

I am indebted to the ADONET framework which made it possible for me to work under very good conditions at Laboratoire Leibniz, Grenoble, where many of these results were born. I would like to thank Henning Bruhn, Tamás Király and Júlia Pap who were always willing to listen to my ideas and to Mihály Bárász and Johanna Becker for checking a preliminary version of this paper.

References

[1] Kouji Arata, Satoru Iwata, Kazuhisa Makino, and Satoru Fujishige, *Locating sources to meet flow demands in undirected networks*, Algorithm theory—SWAT 2000 (Bergen), Lecture Notes in Comput. Sci., vol. 1851, Springer, Berlin, 2000, pp. 300–313.

[2] Jørgen Bang-Jensen and Stéphan Thomassé, *Highly connected hypergraphs containing no two edge-disjoint spanning connected subhypergraphs*, Discrete Appl. Math. **131** (2003), no. 2, 555–559.

[3] Mihály Bárász, Johanna Becker, and András Frank, *An algorithm for source location in directed graphs*, Oper. Res. Lett. **33** (2005), no. 3, 221–230.

[4] Zsolt Fekete, *Source location with rigidity and tree packing requirements*, Oper. Res. Lett. **34** (2006), no. 6, 607–612.

[5] András Frank, Tamás Király, and Zoltán Király, *On the orientation of graphs and hypergraphs*, Discrete Appl. Math. **131** (2003), no. 2, 385–400.

[6] András Frank, Tamás Király, and Matthias Kriesell, *On decomposing a hypergraph into k connected sub-hypergraphs*, Discrete Appl. Math. **131** (2003), no. 2, 373–383.

[7] Michael R. Garey and David S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, W. H. Freeman & Co., New York, NY, USA, 1979.

[8] Hiro Ito, Kazuhisa Makino, Kouji Arata, Shoji Honami, Yuichiro Itatsu, and Satoru Fujishige, *Source location problem with flow requirements in directed networks*, Optim. Methods Softw. **18** (2003), no. 4, 427–435, The Second Japanese-Sino Optimization Meeting, Part II (Kyoto, 2002).

[9] Regina Klimmek and Frank Wagner, *A simple hypergraph min cut algorithm*, Internal Report B 96-02 Bericht FU Berlin Fachbereich Mathematik und Informatik (1995).

[10] Hiroshi Nagamochi and Toshihide Ibaraki, *Computing edge-connectivity in multigraphs and capacitated graphs*, SIAM J. Discrete Math. **5** (1992), no. 1, 54–66.

[11] H. Narayanan, *A note on the minimization of symmetric and general submodular functions*, Discrete Appl. Math. **131** (2003), no. 2, 513–522.

[12] Maurice Queyranne, *Minimizing symmetric submodular functions*, Math. Programming **82** (1998), no. 1-2, Ser. B, 3–12.

[13] Mariko Sakashita, Kazuhisa Makino, Hiroshi Nagamochi, and Satoru Fujishige, *Minimum transversals in posi-modular systems*, 2006, RIMS preprint, RIMS-1547.

[14] Alexander Schrijver, *Combinatorial optimization. Polyhedra and efficiency.*, Algorithms and Combinatorics, vol. 24, Springer-Verlag, Berlin, 2003.