Finding Hypernetworks in Directed Hypergraphs

Daniele Pretolani

DISMI, University of Modena and Reggio Emilia
Via Amendola 2, 42122 Reggio Emilia (Italy)
Tel.: (+39) 0522 522229, e-mail: daniele.pretolani@unimore.it

Abstract

The term "hypernetwork" (more precisely, s-hypernetwork and (s,d)-hypernetwork) has been recently adopted to denote some logical structures contained in a directed hypergraph. A hypernetwork identifies the core of a hypergraph model, obtained by filtering off redundant components. Therefore, finding hypernetworks has a notable relevance both from a theoretical and from a computational point of view.

In this paper we provide a simple and fast algorithm for finding s-hypernetworks, which substantially improves on a method previously proposed in the literature. We also point out two linearly solvable particular cases.

Finding an (s,d)-hypernetwork is known to be a hard problem, and only one polynomially solvable class has been found so far. Here we point out that this particular case is solvable in linear time.

Keywords: graph theory, directed hypergraphs, hyperpaths

1. Introduction

In the last two decades, hyperpaths in directed hypergraphs have often been used to devise formal models and solution methods in many application areas; see [1, 2, 5, 7, 8, 9] to mention a few. Only recently however, Volpentesta [10] investigated some hypergraph structures, referred to as *hypernetworks*, defined in terms of *sets* of hyperpaths. In particular, the *s-hypernetwork* is the union of all the hyperpaths with origin s, while the *(s,d)-hypernetwork* is the union of all the hyperpaths with origin s and destination d.

Hypernetworks were proposed (see [10] and references therein) in the *Virtual Enterprise* context, where they provide a formal model for a network of

(potentially) cooperating economic actors, capable of responding to specific business opportunities. In general, a hypernetwork represents the relevant part of a hypergraph model, that contains all the feasible solutions while excluding redundant components. On the computational side, working on the hypernetwork rather than the whole hypergraph can lead to substantial savings in execution time. This happens in particular when shortest hyperpaths must be repeatedly computed, possibly a huge number of times [5, 6]. For example, the preprocessing phase described in [5, Section 4] actually finds an (s, d)-hypernetwork.

As shown in [10], the s-hypernetwork H_s contained in a given hypergraph H can be found in polynomial time. However, the algorithm proposed in [10] requires a rather involved analysis, and has a high computational complexity. Here we present a simple and fast method, whose complexity is at most quadratic in the size of H. We also point out that H_s can be computed in linear time for two well known particular classes, namely, directed graphs and acyclic hypergraphs.

Opposed to finding H_s, finding an (s, d) hypernetwork is an NP-hard problem, even in the case of directed graphs [10]. In [10] the problem is shown to be polynomially solvable for the class of *stratified* hypergraphs, but no complexity bound is provided. Here we point out that stratified hypergraphs actually coincide with acyclic hypergraphs, and we show that this case can be solved in linear time.

On the theoretical side, we point out the relation between hypernetworks and other known concepts, namely, *flow hypergraphs* and *dominators*. These relations were not observed explicitly in [10], even if similar results were given; note that our treatment is different from, and substantially simpler than, the one in [10].

The paper is organized as follows. In the next section we introduce our notation and recall some known results. In Section 3 we deal with finding s-hypernetworks, while in Section 4 we deal with (s, d) hypernetworks. Conclusions, and suggestions for further research, are reported in Section 5.

2. Definitions and main properties

A directed hypergraph is a pair $H = (V, E)$, where V is the set of nodes and E is the set of hyperarcs; a hyperarc is a pair $e = (T(e), h(e))$ where $\emptyset \neq T(e) \subset V$ and $h(e) \in V \setminus T(e)$. Clearly, the hypergraphs considered here contain directed graphs as a subclass.

In the literature the term hypergraph is often referred to the broader class introduced by Gallo *et al.* [1], who define a hyperarc as an ordered pair (X, Y) where X and Y are two disjoint subsets of nodes. The hypergraphs addressed in our paper, where $|Y| = 1$ for each hyperarc, are called *B-graphs* in [1]; the symmetric case where $|X| = 1$ is referred to as *F-graphs*. Observe that we obtain an F-graph from a B-graph (and vice-versa) by reversing the direction of hyperarcs, i.e., by replacing each pair (X, Y) by (Y, X).

A hypergraph $H' = (V', E')$ is a *sub-hypergraph of* (or is *contained in*) H, denoted as $H' \subseteq H$, if $V' \subseteq V$ and $E' \subseteq E$; we write $H' \subset H$ if the inclusion is strict, i.e., $H' \neq H$. For each node $u \in V$ we define the *forward star* $FS(u) = \{e \in E : \ u \in T(e)\}$ and the *backward star* $BS(u) = \{e \in E : h(e) = u\}$. We let $n = |V|$ and $m = |E|$, while the *size* of hypergraph H is denoted by $S(H) = \sum_{e \in E}(|T(e)| + 1)$. In the following we take $S(H)$ as the size of the input problem, assuming $n+m = O(S(H))$; in particular, we have $S(H) \geq \max\{n, m\}$) if there are no *isolated* nodes, i.e., $FS(u) \cup BS(u) \neq \emptyset$ for each $u \in V$.

A *path* P_{st} in a hypergraph H is a sequence of nodes and hyperarcs:

$$P_{st} = (s = v_1, e_1, v_2, e_2, ..., e_q, v_{q+1} = t)$$

where, for $i = 1, \ldots, q$, $v_i \in T(e_i)$ and $v_{i+1} = h(e_i)$. Node t is *connected* to node s if a path P_{st} exists in H. A *cycle* is a path P_{st}, where $t \in T(e_1)$. A hypergraph is *acyclic* if contains no cycles. Gallo *et al.* ([1], Section 7.2) show[1] that some well known properties of acyclic directed graphs hold true for acyclic hypergraphs.

Property 1. *Given a hypergraph $H = (V, E)$:*

i. *H is acyclic if and only if there exists at least one* topological order *$V = \{v_1, v_2, \ldots, v_n\}$ of the nodes such that, for each $e \in E$*

$$v_i \in T(e) \ \wedge \ h(e) = v_j \quad \Rightarrow \quad i < j. \tag{1}$$

ii. *it takes linear time $O(S(H))$ to check wether H is acyclic and, in that case, to obtain a topological order.*

The concepts of *hyperconnection* and *hyperpath* are captured by the following definitions.

[1]Actually their results are stated for F-graphs, but the translation to B-graphs is straightforward, due to the simmetry between the two classes.

Definition 1. (hyperconnection) *A node $u \in V$ is hyperconnected to s in H if either $u = s$ or there exists a hyperarc $e \in BS(u)$ such that each node $v \in T(e)$ is hyperconnected to s.*

Definition 2. (hyperpath) *A hyperpath from an origin node s to a destination node t (i.e., an s-t hyperpath) is a minimal hypergraph $\pi = (V_\pi, E_\pi)$ where t is hyperconnected to s.*

In Definition 2 minimality of π means that it does not exist any $\pi' \subset \pi$ where t is hyperconnected to s. Clearly, t is hyperconnected to s in H if and only if H contains a hyperpath from s to t. Note that a hyperpath in a directed graph is a simple (i.e., non-looping) directed path. The following properties of hyperpaths are well known, see e.g. [6].

Property 2. *Let $\pi = (V_\pi, E_\pi)$ be a hyperpath from s to t:*

i. *π is acyclic;*
ii. *for each $u \in V_\pi \setminus \{t\}$ there is at least one $e \in E_\pi$ such that $u \in T(e)$, thus there is at least one u-t path in π;*
iii. *for each $u \in V_\pi \setminus \{s\}$ there is exactly one $e \in E_\pi$ such that $u = h(e)$;*
iv. *for each $u \in V_\pi \setminus \{t\}$, π contains exactly one hyperpath from s to u.*

Given a hypergraph $H = (V, E)$ and a distinguished *source node* $s \in V$, we denote by Π_s the set of hyperpaths with origin node s in H, and by Π_{sd} the set of hyperpaths from s to $d \neq s$ in H.

Definition 3. (hypernetworks) *Given $H = (V, E)$ and $s \in V$, the s-hypernetwork $H_s = (V_s, E_s)$ is the union of all the hyperpaths in Π_s, i.e.,*

$$V_s = \bigcup_{\pi \in \Pi_s} V_\pi, \qquad E_s = \bigcup_{\pi \in \Pi_s} E_\pi;$$

for each $d \in V$, $d \neq s$, the (s,d)-hypernetwork $H_{sd} = (V_{sd}, E_{sd})$ is the union of all the hyperpaths in Π_{sd}, i.e.,

$$V_{sd} = \bigcup_{\pi \in \Pi_{sd}} V_\pi, \qquad E_{sd} = \bigcup_{\pi \in \Pi_{sd}} E_\pi.$$

Clearly, V_s is the set of nodes hyperconnected to s in H; from now on we restrict ourselves to (s,d)-hypernetworks where $d \in V_s$, i.e., Π_{sd} is not empty. Note that we have $E_s \subseteq E^{(s)}$, where

$$E^{(s)} = \{e \in E : \; T(e) \subset V_s \; \wedge \; h(e) \in V_s\}$$

is the set of hyperarcs with nodes in V_s; thus we have $H_s \subseteq H^{(s)}$, where $H^{(s)} = (V_s, E^{(s)})$, and this inclusion can be strict, as shown in Example 1.

Example 1. In the hypergraph H in Figure 1 we have $V_s = \{s, a, b, c, d\}$; nodes and hyperarcs in dotted lines do not belong to $H^{(s)}$. Hyperarc 6 belongs to $H^{(s)}$, but not to H_s. Indeed, any hyperpath from s to d must contain hyperarc 5, by Property 2.iii; thus a hyperpath containing 6 would contain a cycle $(a, 5, d, 6, a)$, contradicting Property 2.i.

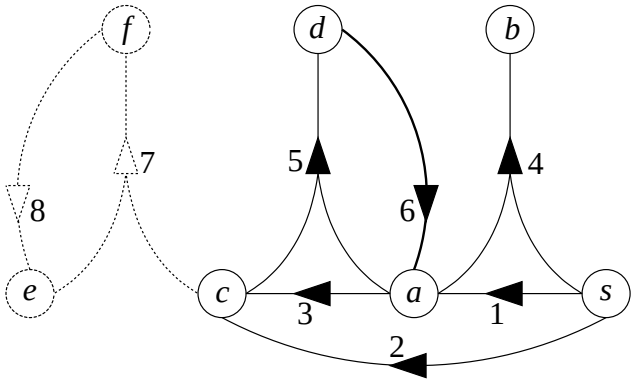

Figure 1: A hypergraph H, and the hypergraphs $H^{(s)}$ and H_s in H.

Hypernetworks can be related to the concepts of *flow hypergraph* and *dominator*. Flow hypergraphs, introduced by Guedes et al. [4], extend the well known concept of *flowgraph* to the broader class of hypergraphs introduced by Gallo *et al.* [1]. We say that $H = (V, E)$ is a flow hypergraph with source $s \in V$ if each node in V is hyperconnected to s in H; this can be denoted by $H = (V, E, s)$. It is easy to see that hypernetworks (both H_s and H_{sd}) as well as the hypergraph $H^{(s)} = (V_s, E^{(s)})$ are flow hypergraphs. Moreover, the concept of dominator in a flowgraph can be extended to flow hypergraphs.

Definition 4. *Given two distinguished nodes u and v in a flow hypergraph $H = (V, E, s)$, u is a dominator of v if u belongs to every hyperpath from s to v in H.*

Note that a reflexive version of the dominance relation is referred to as "s-indispensability" in [10]; however, the connection with dominance is not pointed out there.

3. Finding the *s*-hypernetwork

As discussed above, to identify H_s we can proceed as follows: first build the hypergraph $H^{(s)} = (V_s, E^{(s)})$ and then remove from $E^{(s)}$ all those hyperarcs that do not belong to any hyperpath in Π_s. The set V_s can be found in $O(S(H))$ time, applying procedure *B-Visit* [1]; given V_s, $E^{(s)}$ can be found in time $O(S(H))$. Thus building $H^{(s)}$ takes linear time in the size of H.

In the following we show how to derive the hypernetwork H_s from $H^{(s)}$, first for the general case and then for two easier classes. We exploit the following quite intuitive fact, following from Property 2.iv: a hyperarc e in $E^{(s)}$ belongs to H_s if and only if it belongs to a hyperpath from s to $h(e)$ in $H^{(s)}$. Hyperarcs that do not fulfill this condition are removed, and the remaining ones define H_s.

3.1. The general case

Besides finding hyperarcs to be removed from $H^{(s)}$, our algorithm generates an $n \times n$ $\{0,1\}$ matrix D (the analogue of the "s-indispensability matrix" in [10]) where $D_{ij} = 1$ if node v_i is a dominator of node v_j in $H^{(s)}$, and $D_{ij} = 0$ otherwise. Here we assume an arbitary but fixed ordering of the nodes in $H^{(s)}$, i.e., $V_s = \{s = v_1, v_2, \ldots, v_n\}$.

To improve efficiency, all the hyperarcs in the backward star of a node are processed at the same time. Thus the algorithm works in *stages*, processing a node $u \in V_s \setminus \{s\}$ at each stage; recall that $BS(u)$ is not empty. For each u, a hypergraph $H^u = (V^u, E^u)$ is obtained from $H^{(s)}$ as follows: for each hyperarc $e \in BS(u)$ a new node u^e is introduced, and e is replaced by a new hyperarc $e^u = (T(e), u^e)$. Formally,

- $V^u = V_s \cup \{u^e : e \in BS(u)\}$
- $E^u = \left(E^{(s)} \setminus BS(u)\right) \cup \{e^u = (T(e), u^e) : e \in BS(u)\}$.

Note that, in H^u, node u has an empty backward star, and each node u^e has an empty forward star; moreover, $S(H^u) = S(H^{(s)})$.

Example 1 (continued) The hypergraph $H^{(s)}$ shown in Figure 1 appears on the left in Figure 2; on the right, the hypergraph H^u for $u = a$ is shown. The two hyperarcs 1 and 6 in $BS(a)$ are replaced by the new hyperarcs 1^a and 6^a, respectively, introducing two new nodes $a^1 = h(1^a)$ and $a^6 = h(6^a)$. The nodes hyperconnected to s in H^u (shown in thick lines) are c and a^1.

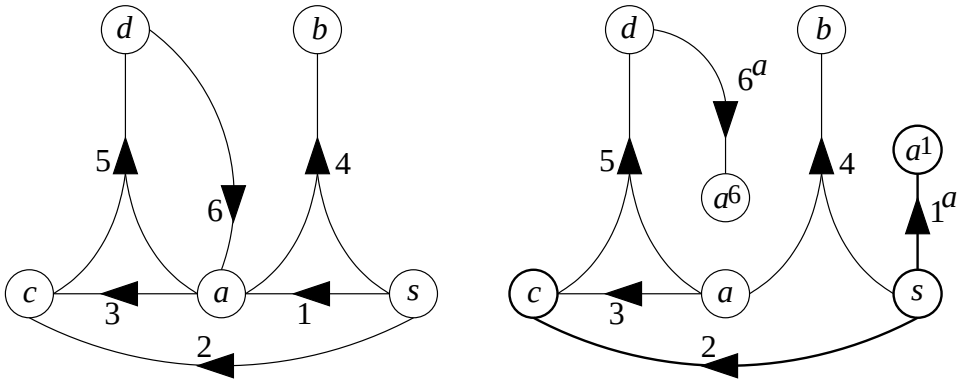

Figure 2: Hypergraph $H^{(s)}$ (left) and hypergraph H^u for $u = a$ (right).

Let us denote by V_s^u the set of nodes hyperconnected to s in H^u; clearly, $u \notin V_s^u$. The set V_s^u provides all the necessary information to identify the hyperarcs in $BS(u)$ that must be removed and the nodes dominated by u.

Lemma 1. *Given a hyperarc $e \in E^{(s)}$ with $h(e) = u$, $e \in E_s$ if and only if node $u^e \in V_s^u$.*

Proof. (if) Let π be a hyperpath from s to u^e in H^u; clearly, π contains hyperarc e^u, and does not contain node u, thus by replacing e^u by e we obtain a hyperpath from s to u in $H^{(s)}$.
(only if) Let $\pi \subseteq H^{(s)}$ be an s-u hyperpath containing e: by replacing e by e^u we obtain a hyperpath from s to u^e in H^u. □

Lemma 2. *For each node $v \neq u$ in V_s, node u is a dominator of v in $H^{(s)}$ if and only if $v \notin V_s^u$.*

Proof. (if) Let $\pi \subseteq H^{(s)}$ be an s-v hyperpath: if π contains node u then π is not contained in H^u, otherwise $\pi \subset H^u$; thus $v \notin V_s^u$ implies that each s-v hyperpath in $H^{(s)}$ contains node u.
(only if) Suppose there is an s-v hyperpath π in H^u; clearly, π does not contain u; by Property 2.ii, π does not contain any node u^e, thus $\pi \subseteq H^{(s)}$. □

Example 1 (continued) Consider again the hypergraph H^u, for $u = a$, in Figure 2. We have $a^6 = h(6^a) \notin V_s^u$, while $a^1 = h(1^a) \in V_s^u$; indeed,

recall that hyperarc 6 does not belong to H_s, while hyperarc 1 belongs to H_s. Moreover, we have $b, d \notin V_s^u$; indeed, node a dominates nodes b and d in $H^{(s)}$.

Algorithm *FindHs*
input: hypergraph $H^{(s)} = (V_s, E^{(s)})$, $V_s = \{s = v_1, v_2, \ldots, v_n\}$
output: hyperarc label $R \in \{0,1\}^m$, matrix $D \in \{0,1\}^{n \times n}$
1. $D_{11} := 0$; **for all** $j \in \{2, \ldots, n\}$: $D_{1j} := 1$;
2. **for all** $i \in \{2, \ldots, n\}$ **do**
   let $u := v_i$
3. build $H^u = (V^u, E^u)$
4. apply *B-Visit*(H^u, s), obtaining the set V_s^u
5. **for all** $e \in BS(u)$:
   if $u^e \in V_s^u$ **then** $R_e := false$ **else** $R_e := true$;
6. **for all** $j \in \{1, \ldots, n\}$:
   if $v_j \in V_s^u$ **or** $j = i$ **then** $D_{ij} := 0$ **else** $D_{ij} := 1$;
   end-for

Table 1: Algorithm *FindHs*.

Our algorithm, referred to as *FindHs*, is described in Table 1. We use a label R to mark hyperarcs to be removed from $H^{(s)}$, in particular, we set $R_e = true$ if hyperarc e is removed, and $R_e = false$ otherwise. Line 1 writes the first row of D; the main loop starts at line 2, and each stage consists of lines 3-6. Lines 3 and 4 take $O(S(H))$ time, while lines 5 and 6 take $O(m)$ and $O(n)$ time, respectively. All the step of the algorithm require $O(S(H))$ space, but a further $O(n^2)$ space is required for matrix D. Note however that it is not necessary to store D during the algorithm, i.e., each row i of D can be sent to output and discarded at the end of step 6; in this case, the overall space requirement becomes $O(S(H))$.

Theorem 1. *Algorithm* FindHs *finds the hypernetwork H_s and writes the matrix D in $O(n \cdot S(H))$ time, with an $O(S(H))$ space requirement.*

The *forward procedure* proposed in [10] finds H_s with time complexity $O(n^2 m^2 T)$, where $T = \max_{e \in E} |T(e)|$. In general $S(H)$ is $O(mT)$, thus

FindHs reduces complexity by a factor nm at least. Moreover, it is easy to define classes of hypergraphs where $S(H)$ is $O(m)$ while T is $\Theta(n)$; in these cases the improvement increases to a factor n^2m.

Combining Lemma 1 and Lemma 2, and taking into account the structure of a hypergraph H^u, we obtain the following relevant relation.

Property 3. *A hyperarc $e \in E^{(s)}$ belongs to H_s if and only if $h(e)$ does not dominate any node $v \in T(e)$ in the flow hypergraph $H^{(s)}$.*

Property 3 implies that H_s can be determined in linear time $O(S(H))$ if the matrix D is available. Moreover, even if D is not known, $H^{(s)}$ can be determined in linear time if all the relevant information about dominators in $H^{(s)}$ is available in a suitable form; we exploit this fact to devise linear time methods for two particular classes.

3.2. Linearly solvable cases

The hypernetwork H_s can be found in linear time $O(S(H))$ for two non-trivial classes of hypegraphs: acyclic hypergraphs and directed graphs. Note that hypergraphs in these classes can be recognized in linear time. The two classes require quite different algorithmic techniques, and we discuss them separately, pointing out the information about dominators that is relevant to each case.

Acyclic hypergraphs. This case turns out to be quite simple, since if H is acyclic then $H_s = H^{(s)}$ (the "only if" relation does not hold here). This follows from Property 3, observing that in an acyclic hypergraph the node $h(e)$ does not dominate any node in $T(e)$. More precisely, $h(e)$ does not belong to any hyperpath from s to $u \in T(e)$: if this happens, by Property 2.ii there is a path from $h(e)$ to u, that together with e gives a cycle. Note that, for acyclic hypergraphs, a quite limited information on the dominance relation suffices to find H_s; in fact, we find H_s without computing the matrix D. We leave as an open question whether it is possible to find D in less than $O(n \cdot S(H))$ time for an acyclic hypergraph.

Directed graphs. This case is definitely more complex. Recall that, in directed graphs, a hyperpath corresponds to a simple directed path, i.e., $H^{(s)}$ is a flowgraph. It is easy to see (and actually follows from Property 3) that an arc $(u, v) \in E^{(s)}$ belongs to a simple path from s to v in $H^{(s)}$ if and only if v is

not a dominator of u. Our method checks this condition efficiently, exploiting some known results about dominance in flowgraphs, described below.

In a flowgraph $G = (N, A, s)$ a node u is an *immediate* dominator of node v if there is no dominator w of v such that u dominates w. Each node $v \neq s$ in G has a *unique* immediate dominator, and this implicitly defines a directed spanning tree $T = (N, A^T)$, rooted at s, where $(u, v) \in A^T$ if u is the immediate dominator of node v. Note that arcs in A^T do not necessarily correspond to arcs in A. The dominators of a node v are the nodes in the unique path from s to v in T. The immediate dominators, and thus the tree T, can be found in linear time $O(|A|)$, although this task is definitely far from trivial, see [3] and references therein.

Now suppose that we computed the tree $T = (V_s, A^T)$ in the flowgraph $H^{(s)} = (V_s, E^{(s)})$. In order to find the arcs to be removed from $E^{(s)}$ we perform a *depth first visit* of T. The visit is performed by the recursive procedure *DFVisit*, shown in Table 2. We maintain a node label l that marks *active* nodes: a node u is active (i.e., $l(u) = true$) if and only if the visit of the subtree rooted at u started but did not yet terminate. It is easy to see that, when *DFVisit*(u) is called and line 1 is executed, the active nodes are exactly those in the path from s to u in T, i.e., u and its dominators. At that point (line 2) we process each arc $(u, v) \in E^{(s)}$: if v is active then we set $R_e := true$, otherwise we set $R_e := false$. Then (line 3) we recursively call *DFVisit*(w) for each successor w of u in T. Initially, all the node labels are set to *false*; the visit of T is started by the call *DFVisit*(s). Clearly, the whole process takes linear time $O(m)$.

---

procedure *DFVisit*(u)
1. $l(u) := true$;
2. **for all** $e = (u, v) \in E^{(s)}$:
   if $l(v) = true$ **then** $R_e := true$ **else** $R_e := false$;
3. **for all** $(u, w) \in A^T$: **call** *DFVisit*(w);
4. $l(u) := false$;
   return

Table 2: Recursive procedure *DFVisit*.

For directed graphs, the matrix D is redundant, since the dominance re-

lation is completely described by T. Observe that, opposed to what happens for acyclic hypergraphs, here we exploit the complete knowledge of the dominance relation to find H_s. We do not know if H_s may be found in linear time, or at least in less than $O(nm)$ time, without computing T.

4. Finding an (s, d)-hypernetwork

The problem of finding an (s, d)-hypernetwork in a hypergraph H is in general NP-hard, but it can be solved in linear time if H is acyclic. We prove this result first, and then we point out the relations with stratified hypergraphs.

4.1. A linear time method for acyclic hypergraphs

Similar to H_s, finding H_{sd} is rather simple, and does not require knowledge of the dominance relation. We exploit a general property of acyclic hypergraphs [8, Property 2.1] that in our context can be restated as follows:

Property 4. *An acyclic hypergraph H such that $BS(u) \neq \emptyset$ for each node $u \neq s$ in H is a flow hypergraph.*

Recall that each (s, d)-hypernetwork $H_{sd} = (V_{sd}, E_{sd})$ is contained in H_s, and that $H_s = H^{(s)}$ if H is acyclic. To obtain H_{sd} it may be necessary to remove some hyperarcs and nodes from $H^{(s)}$. Observe that H_{sd} contains at least one path P_{ud} for each node $u \in V_{sd}$, as follows immediately from Property 2.ii. Therefore, we can safely remove a node $u \in V_s$ if there is no path from u to d in $H^{(s)}$. Let us denote by $V^{(sd)} \subseteq V_s$ the set of remaining nodes, i.e., $u \in V^{(sd)}$ if and only if d is connected to u in $H^{(s)}$; moreover, let

$$E^{(sd)} = \{e \in E^{(s)} : \ T(e) \subset V^{(sd)} \ \wedge \ h(e) \in V^{(sd)}\}$$

be the set of hyperarcs in $E^{(s)}$ with nodes in $V^{(sd)}$.

Theorem 2. *If H is acyclic, for each (s, d)-hypernetwork $H_{sd} = (V_{sd}, E_{sd})$ we have $V_{sd} = V^{(sd)}$ and $E_{sd} = E^{(sd)}$.*

Proof. It follows immediately from Property 2.ii that $V_{sd} \subseteq V^{(sd)}$, and thus $E_{sd} \subseteq E^{(sd)}$. It remains to show that $E^{(sd)} \subseteq E_{sd}$, which implies $V^{(sd)} \subseteq V_{sd}$. First note that if $u \in V^{(sd)}$ then we have $e \in E^{(sd)}$ for each $e \in E^{(s)}$ such that $h(e) = u$, which implies that $H^{(sd)} = (V^{(sd)}, E^{(sd)}, s)$ is a flow hypergraph,

due to Property 4. Consider a generic hyperarc $e_0 \in E^{(sd)}$ with $h(e_0) = u$, and a path

$$P_{ud} = (u = v_1, e_1, v_2, e_2, ..., e_q, v_{q+1} = d)$$

contained in $H^{(sd)}$. Note that it can be $q = 0$, i.e., $u = d$. Obtain $H' \subseteq H^{(sd)}$ from $H^{(sd)}$ as follows: for $i = 1, \ldots, q+1$ remove from $E^{(sd)}$ all the hyperarcs in $BS(v_i)$ except e_{i-1}. From Property 4, H' is a flow hypergraph; moreover, by Property 2.iii, every hyperpath from s to d in H' contains the path P_{ud}, and thus contains e_0. In conclusion, $e_0 \in E_{sd}$. □

The set $V^{(sd)}$ can be easily found in linear time $O(S(H))$ by a backward version of procedure Visit [1]. It follows that an (s, d)-hypernetwork in an acyclic hypergraph can be found in linear time.

4.2. Acyclic and stratified hypergraphs

The definition of stratified hypergraphs ([10, Definition 4.4]) can be restated as follows.

Definition 5. *A hypergraph $H = (V, E)$ is* stratified *if V can be partitioned into a sequence of subsets $N_0, N_1, \ldots, N_k$, with $k > 0$, satisfying:*

1. *let $u \in N_i$ and $v \in N_j$: if $i \geq j$ then there does not exist any hyperarc $e \in BS(v)$ such that $u \in T(e)$;*
2. *for each $u \in N_i$, $i = 1, \ldots, k$, there exists a hyperarc e with $h(e) = u$ and $T(e) \subseteq \cup_{j=0,i-1} N_j$.*

Let us say that a hypergraph is *nonempty* if it contains at least one hyperarc. Note that a stratified hypergraph must be nonempty.

Property 5. *A hypergraph $H = (V, E)$ is stratified if and only if it is a nonempty acyclic hypergraph.*

Proof. (if) Let $V = \{v_1, v_2, \ldots, v_n\}$ be a topological order satisfying (1). Let $N_0 = \{v_i \in V : \ BS(v_i) = \emptyset\}$, i.e., N_0 is the set of nodes with an empty backward star; since H is nonempty, $N_0 \subset V$. We sort the nodes in $V \setminus N_0$ according to the topological order, i.e., we let $V \setminus N_0 = \{v_{o(1)}, v_{o(2)}, \ldots, v_{o(k)}\}$, where $o(i) < o(j)$ for each $1 \leq i < j \leq k$. For each $e = (T(e), v_{o(j)}) \in E$, and for each $v_{o(i)} \in T(e)$, we have $i < j$ by Property 1.i. We define $N_i = \{v_{o(i)}\}$ for each $i = 1, \ldots, k$; it is routine to check that the partition $N_0, N_1, \ldots, N_k$ satisfies conditions 5.1 and 5.2.
(only if) It is easy to see that H is acyclic since, for each $e \in E$, a node $u \in T(e)$ cannot be connected to the node $h(e)$. □

Property 5 shows that our linear time method completes a polynomiality result already given in [10]. Note that this polynomiality result exploits some nontrivial theoretical properties of (s, d)-hypernetworks (namely, Theorem 4.1 in [10]) that hold true also if the hypergraph is not acyclic. Our approach is simpler and more direct, since we derive our result from a known property of acyclic hypergraphs, namely Property 4.

Remark 1. In the original statement of condition 1. in Definition 5, i.e., condition (4.1) in [10, Definition 4.4], the inequality "$i > j$" appears instead of "$i \geq j$" (probably due to a typographical error). This actually results in a *weaker* condition, that allows e.g. a hyperarc $e \in E$ such that $T(e) \cup \{h(e)\} \subseteq N_i$. Clearly, the weaker condition defines a proper extension of acyclic hypergraphs. Unfortunately, the polynomiality result does not extend to this broader class, unless $P = NP$. To see this, consider a generic hypergraph $H = (V, E)$ and a pair of nodes s and d in V. Define a hypergraph H' adding to H a new node d' and a new arc (d, d'). Choosing $N_0 = V$ and $N_1 = \{d'\}$ we see that H' is stratified according to the weaker condition. Clearly, every s-d hyperpath in H corresponds to a unique s-d' hyperpath in H', and vice-versa. Thus finding the (s, d')-hypernetwork in H' is as hard as finding the (s, d)-hypernetwork in H, i.e., NP-hard.

5. Conclusions

Hypernetworks in directed hypergraphs have been introduced as a formal descriptive tool within a specific application. In our paper we pointed out that finding hypernetworks has a relevant interest, from a computational point of view, in many different contexts.

We provided a fast and simple method for finding the s-hypernetwork, and two non-trivial classes solvable in linear time. For one of this two classes we also devised a linear time method for finding the (s, d)-hypernetwork. On the theoretical side, we pointed out the relations between hypernetworks, flow hypergraphs, and dominators.

Our results may be extended along two different directions. On one side, find classes where hypernetworks can be identified exploiting a partial knowledge of the dominators, as for acyclic hypergraphs. On the other side, find classes where dominators can be found in less than $O(n \cdot S(H))$ time. One possible candidate class are the *reducible* flow hypergraphs [4]. Independently of hypernetworks, we believe that the computation of dominators in flow hypergraphs may be an interesting subject for further research.

Acknowledgements

The author thanks two anonymous referees for constructive comments which helped to improve the first version of this article.

References

[1] G. Gallo, G. Longo, S. Nguyen, and S. Pallottino. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2-3) (1993) 177–201.

[2] G. Gallo and M.G. Scutellà. A note on minimum makespan assembly plans. *European Journal of Operational Research*, 142(2) (2002) 309–320.

[3] L. Georgiadis and R. E. Tarjan. Finding dominators revisited. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms* (SODA '04) 869-878. SIAM, Philadelphia, 2004.

[4] A. L. P. Guedes, L. Markenzon, and L. Faria. Flow hypergraph reducibility. *Discrete Applied Mathematics*, 159 (2011) 1775-1785.

[5] L.R. Nielsen, K.A. Andersen, and D. Pretolani. Bicriterion shortest hyperpaths in random time-dependent networks. *IMA Journal of Management Mathematics*, 14(3) (2003) 271–303.

[6] L.R. Nielsen, K.A. Andersen, and D. Pretolani. Finding the K shortest hyperpaths. *Computers & Operations Research*, 32(6) (2005) 1477–1497.

[7] L.R. Nielsen and A.R. Kristensen. Finding the K best policies in a finite-horizon Markov decision process. *European Journal of Operational Research*, 175(2) (2006) 1164–1179.

[8] D. Pretolani. A directed hypergraph model for random time-dependent shortest paths. *European Journal of Operational Research*, 123(2) (2000) 315–324.

[9] D. Pretolani, L.R. Nielsen, K.A. Andersen, and M. Ehrgott. Time-adaptive and history-adaptive multicriterion routing in stochastic, time-dependent networks. *Operations Research Letters*, 37 (2009) 201–205.

[10] A. P. Volpentesta. Hypernetworks in directed hypergraphs. *European Journal of Operational Research*, 188 (2008) 390–405.