Theoretical Computer Science 410 (2009) 2592–2618

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Linear connectivity problems in directed hypergraphs[☆]

Mayur Thakur[a], Rahul Tripathi[b,*]

[a] Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA

[b] Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620, USA

ARTICLE INFO

Article history:
Received 1 August 2007
Received in revised form 5 November 2008
Accepted 27 February 2009
Communicated by G. Ausiello

Keywords:
Directed hypergraphs
Directed hyperpaths
Cyclomatic number of hypergraphs
Complexity classes

ABSTRACT

We introduce a notion of hyperconnection (formally called *L-hyperpath*) between vertices in a directed hypergraph and relate this notion to existing notions of hyperpaths in directed hypergraphs. We show that some interesting questions in problem domains such as distributed secret sharing and routing in packet filtered networks are basically questions about the existence of *L*-hyperpaths in directed hypergraphs. We study the computational complexity of problems related to *L*-hyperpaths and the *L*-cyclomatic number of directed hypergraphs (the minimum number of hyperedges that need to be deleted to make a directed hypergraph free of *L*-hypercycles). We prove that the *L*-hyperpath existence problem, the *L*-cyclomatic number problem, the minimum *L*-cyclomatic set problem, and the minimal *L*-cyclomatic set problem are each complete for the complexity class NP, Σ_2^p, Π_2^p, and DP, respectively.

Published by Elsevier B.V.

1. Introduction

A directed hypergraph is a generalization of a directed graph in which each directed hyperedge is allowed to have multiple source (tail) vertices and multiple destination (head) vertices. A (simple) directed edge is a hyperedge with exactly one tail vertex and exactly one head vertex. Directed hypergraphs have been used to model a wide variety of problems in propositional logic [6,22], relational databases [2,36,11], urban transportation planning [10,23], chemical reaction mechanisms [35,40], Petri nets [5,26], operations research [10], and probabilistic parsing [18]. They have been introduced under different names such as “And-Or graphs” and “FD-graphs”.

Ausiello, D’Atri, and Saccá [2] introduced the notion of a directed hypergraph, though they called this notion an “FD-graph”, where FD stands for “functional dependency”. They used this notion to represent FDs among attributes in relational databases and presented efficient algorithms for the manipulation of FDs (e.g., the closure of FDs and the minimal coverings of FDs). Gallo et al. [10] formalized the basic notions related to directed hypergraphs such as connectivity, paths, and cuts, and showed applications of directed hypergraphs to problems such as functional dependency in relational databases and route planning in urban transport system design. More recently, a unified view of deterministic and probabilistic parsing has been obtained by showing them to be equivalent to traversals in directed hypergraphs [18]. Directed hypergraphs have also been used in the representation and analysis of chemical reaction mechanisms, in particular in the theory of intermediate reactions [35,40].

☆ The authors were supported in part by grants NSF-INT-9815095 and NSF-CCF-0426761 when they were affiliated with the University of Rochester, where the preliminary part of this work was done. Rahul Tripathi was also supported in part by the New Researcher Grant of the University of South Florida. A preliminary version of this paper appeared in the *Proceedings of the 24th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS ’04)*.

* Corresponding author. Tel.: +1 813 326 8302; fax: +1 813 974 5456.
E-mail addresses: mayurthakur@google.com (M. Thakur), tripathi@cse.usf.edu (R. Tripathi).

0304-3975/$ – see front matter. Published by Elsevier B.V.
doi:10.1016/j.tcs.2009.02.038

The notion of *connection* in a directed graph can be recursively defined as follows. Each vertex is connected to itself and a vertex x is connected to another vertex y, if there exists a vertex z such that there is a connection from vertex x to vertex z and there is an edge from vertex z to vertex y. But there does not seem to be one common intuitive notion for *hyperconnection* (i.e., connection in directed hypergraphs). In fact, different notions of hyperpaths and hypercycles in directed hypergraphs have been defined in the literature [7,8,10,24,40] based on varying intuitive notions of hyperconnection in problem domains. For example, if we are trying to model "logical implications" among hypotheses in logic using directed hyperedges, then the notion of a hyperconnection from s to t would correspond to a derivation of t using hypothesis s [10]. On the other hand, if we are using directed hyperedges to model "intermediate reactants to product" relationship in chemical reaction mechanisms, the notion of a hyperconnection from s to t may correspond to a method of producing t from s such that there are no intermediates [40].

In this paper, first we review these notions of hyperpaths and hypercycles in directed hypergraphs. We describe our notion of hyperconnection, and define L-hyperpaths and L-hypercycles in directed hypergraphs. Basically, an L-hyperpath is an alternating sequence of distinct vertices and distinct hyperedges (just as a simple path in a directed graph is an alternating sequence of distinct vertices and distinct edges) such that if a vertex v is the immediate predecessor of a hyperedge e in this sequence, then v is a tail vertex of e, and if w is a tail vertex of e, then either w is the source vertex of the L-hyperpath or w occurs in the head of a hyperedge that precedes e in this sequence. The "L" ("linear") denotes the fact that the vertices and hyperedges in an L-hyperpath can be (meaningfully) put in a sequence. An L-hypercycle is defined analogously as an alternating sequence of vertices and hyperedges except that we require that this sequence must begin and end at the same vertex. (Precise definitions and examples of L-hyperpaths and L-hypercycles appear in Section 3.)

We identify domains such as distributed secret sharing and packet filtered networks where the notion of an L-hyperpath can be used to capture key problems. We study the computational complexity of basic L-hyperpath problems in directed hypergraphs. In particular, we study the existence of L-hyperpaths of small cost, size, and rank (see Definition 4.2) between two given vertices.

The cyclomatic number of a graph is the minimum number of edges that need to be removed to make the graph acyclic. Intuitively speaking, the cyclomatic number of a graph measures the degree of cyclicity of a graph. We can define the cyclomatic number of a directed hypergraph analogously to that of a graph. That is, the L-cyclomatic number of a directed hypergraph is the minimum number of hyperedges that need to be removed so that the resulting hypergraph does not contain any L-hypercycle. We study the computational complexity of the following decision problems related to the L-cyclomatic number of directed hypergraphs: computing the L-cyclomatic number of a given directed hypergraph, computing whether a given set of hyperedges forms a minimum L-cyclomatic set, and computing whether a given set of hyperedges forms a minimal L-cyclomatic set. We prove that these problems are each complete for a well-known complexity class (see Definition 5.2 for precise definitions of these problems): CYCLOMATIC-NUMBER is Σ_2^p-complete, MIN-CYCLOMATIC-SET is Π_2^p-complete, and MINIMAL-CYCLOMATIC-SET is DP-complete.

It is interesting to compare the complexity of connectivity problems on directed graphs and directed hypergraphs. In particular, consider the complexity of the following *path existence* problems:

- Is there a (directed) path between two given vertices of a given directed graph? This problem is known to be NL-complete [16,15].
- Is there a B-hyperpath (see Definition 2.3) between two given vertices of a given directed hypergraph? This problem is known to be P-complete [10].
- Is there an L-hyperpath between two given vertices of a given directed hypergraph? In this paper, we show that this problem is NP-complete.

Compare also the complexity of the following *cyclomatic number* problems:

- Is there a set of k edges, for a given k, whose deletion makes a given directed graph free of cycles? This problem is called the *feedback arc set* problem, which is known to be NP-complete [17].
- Is there a set of k hyperedges, for a given k, whose deletion makes a given directed hypergraph free of L-hypercycles? In this paper, we show that this problem is Σ_2^p-complete.

Since tail and head sets of a directed hyperedge can contain an arbitrary number of vertices, the number of hyperedges in a directed hypergraph can be exponential in the number of vertices. Thus, in some cases it might not be feasible to enumerate all the hyperedges that exist in a directed hypergraph even though it might be possible to describe all of them succinctly. For example, the set of hyperedges might be represented more succinctly using a circuit that takes as input two sets (X and Y) of vertices. The set of directed hyperedges represented by this circuit is all pairs (X, Y) of subsets of vertices such that the circuit outputs 1 on input (X, Y). Galperin and Wigderson [13] and Papadimitriou and Yannakakis [28] showed that for many graph properties the complexity of checking the property "jumps" considerably (for example, from being NP-complete to being NEXP-complete) when the graphs are represented succinctly. We show that the complexity of the L-hyperpath existence problem remains NP-complete when the input is succinctly represented. On the other hand, we show that, for k-directed hypergraphs (hypergraphs in which the number of head and tail vertices in any hyperedge is bounded by a constant k), the L-hyperpath existence problem is NEXP-complete when the hypergraphs are succinctly represented.

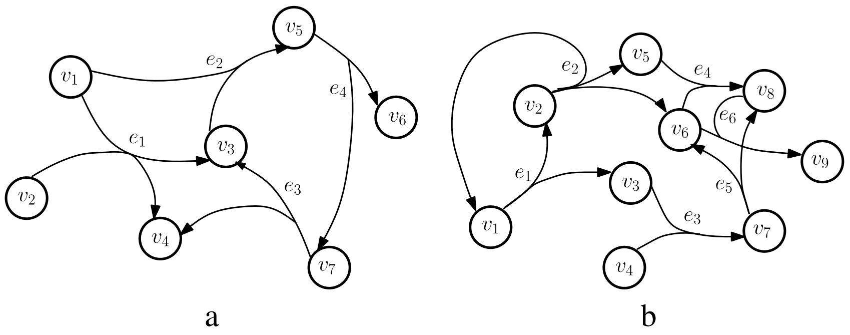


Fig. 1. (a) A directed hypergraph $\mathcal{H}_1$ with 7 vertices and 4 hyperedges. (b) A directed hypergraph $\mathcal{H}_2$ with 9 vertices and 6 hyperedges. In both the figures, arrows on a hyperedge point to the vertices in the head of the hyperedge.

This paper is organized as follows. Section 2 discusses the different notions of directed hyperpaths that exist in the literature. In Section 3, we formally define the notion of L-hyperpaths and introduce other concepts that will be used in the rest of the paper. We also describe two real world domains that can be modeled using directed hypergraphs and show that some interesting questions in these domains are essentially questions about the existence of L-hyperpaths in directed hypergraphs. In Section 4, we study the complexity of finding small (with respect to various metrics) L-hyperpaths in directed hypergraphs. In Section 5, we characterize the complexity of problems related to the L-cyclomatic number of directed hypergraphs. Finally in Section 6, we look at the complexity of succinct versions of the L-hyperpath existence problem.

2. Notions of hyperpaths in directed hypergraphs

In this section, we formally define directed hypergraphs and compare the different notions of directed hyperpaths that have appeared in the literature. For the purpose of comparison with directed hypergraphs, we first define undirected hypergraphs and present the notion of simple paths in undirected hypergraphs.

Definition 2.1 (*See [9]*). 1. An *undirected hypergraph* $\mathcal{H}$ is a pair (V, E), where V is a finite set of vertices and $E \subseteq 2^V$ is a finite set of hyperedges such that, for every $e_i \in E$, $e_i \neq \emptyset$.

2. A *simple path* P_{st} from vertex s to vertex t in $\mathcal{H}$ is a sequence $(v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1})$ consisting of distinct vertices v_i, where $1 \leq i \leq k+1$, and distinct hyperedges e_j, where $1 \leq j \leq k$, such that $s = v_1$, $t = v_{k+1}$, and for all $1 \leq i \leq k$, $v_i, v_{i+1} \in e_i$. If only the constraint $s \neq t$ is not required and instead $s = t$ is enforced in this definition, then P_{st} is a *simple cycle*.

Just as undirected hypergraphs generalize undirected graphs, in the same way directed hypergraphs generalize directed graphs. Depending on the type of hyperedges, we may define the subclasses B-hypergraphs and F-hypergraphs of directed hypergraphs.

Definition 2.2 (*[10]*). 1. A *directed hypergraph* $\mathcal{H}$ is a pair (V, E), where V is a finite set and $E \subseteq 2^V \times 2^V$ such that, for every $e = (T(e), H(e)) \in E$, $T(e) \neq \emptyset$, $H(e) \neq \emptyset$, and $T(e) \cap H(e) = \emptyset$. For every integer $k \geq 1$, a *k-directed hypergraph* $\mathcal{H} = (V, E)$ is a directed hypergraph in which, for every $e \in E$, $|T(e)| \leq k$ and $|H(e)| \leq k$.

2. A *B-hyperedge* (*F-hyperedge*) is a hyperedge $e = (T(e), H(e))$ such that $|H(e)| = 1$ (respectively, $|T(e)| = 1$).
3. A directed *B-hypergraph* (*F-hypergraph*) is a directed hypergraph $\mathcal{H}$ such that each hyperedge in $\mathcal{H}$ is a B-hyperedge (respectively, F-hyperedge).

For a directed hypergraph $\mathcal{H} = (V, E)$, we refer to V also as $V(\mathcal{H})$ and E also as $E(\mathcal{H})$. Let $e = (T(e), H(e))$ be a hyperedge in some directed hypergraph $\mathcal{H}$. Then, $T(e)$ is known as the *tail* of e and $H(e)$ is known as the *head* of e. We say that e is *incident* on each vertex in $T(e) \cup H(e)$. For each $v \in T(e)$, e is an *outgoing* hyperedge from v and for each $v \in H(e)$, e is an *incoming* hyperedge to v. If $|T(e)| = |H(e)| = 1$, then e is also referred to as a simple directed edge. The size of representing a directed hypergraph $\mathcal{H}$ is taken to be $|V(\mathcal{H})| + \sum_{e \in E(\mathcal{H})} (|T(e)| + |H(e)|)$ unless another representation scheme is explicitly mentioned as in, for example, Section 6.

Given a directed hypergraph $\mathcal{H} = (V, E)$, we say that $\mathcal{H}' = (V', E')$ is a *subhypergraph* of $\mathcal{H}$ if $V' \subseteq V$, $E' \subseteq E$, and $E' \subseteq 2^{V'} \times 2^{V'}$. The symmetric image $\overline{\mathcal{H}}$ of $\mathcal{H}$ is a directed hypergraph defined as follows: $V(\overline{\mathcal{H}}) = V(\mathcal{H})$ and $E(\overline{\mathcal{H}}) = \{(H, T) \mid (T, H) \in E(\mathcal{H})\}$.

Fig. 1 (part a) shows a directed hypergraph $\mathcal{H}_1$ with vertices $\{v_1, v_2, \ldots, v_7\}$ and hyperedges $\{e_1, e_2, e_3, e_4\}$, where $e_1 = (\{v_1, v_2\}, \{v_3, v_4\})$, $e_2 = (\{v_1, v_3\}, \{v_5\})$ $e_3 = (\{v_7\}, \{v_3, v_4\})$, and $e_4 = (\{v_5\}, \{v_6, v_7\})$. Note that e_2 is a B-hyperedge, e_3 and e_4 are F-hyperedges, and e_1 is neither a B- nor an F-hyperedge in $\mathcal{H}_1$. Similarly, Fig. 1 (part b) shows a directed hypergraph $\mathcal{H}_2$ with vertices $\{v_1, v_2, \ldots, v_9\}$ and hyperedges $\{e_1, e_2, \ldots, e_6\}$.

Gallo et al. [10] defined two basic types of paths in hypergraphs: simple paths and hyperpaths. They defined three different types of hyperpaths: B-hyperpaths, F-hyperpaths, and BF-hyperpaths.

Definition 2.3 (*[10]*). Let $\mathcal{H} = (V, E)$ be a directed hypergraph and let $s, t \in V(\mathcal{H})$.

1. A *simple path* P_{st} from s to t in $\mathcal{H}$ is a sequence $(v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1})$ consisting of vertices v_i, where $1 \le i \le k+1$, and distinct hyperedges e_j, where $1 \le j \le k$, such that $s = v_1$, $t = v_{k+1}$, and for every $1 \le i \le k$, $v_i \in T(e_i)$ and $v_{i+1} \in H(e_i)$. If, in addition, $t \in T(e_1)$ then P_{st} is a *simple cycle*. The simple path P_{st} is called *elementary* if all the vertices v_i, where $1 \le i \le k+1$, are distinct. A simple path is *cycle-free* if it does not contain any subpath that is a simple cycle. (Here, a *subpath* of P_{st} is $P_{v_i v_j} = (v_i, e_i, v_{i+1}, \ldots, v_j, e_j, v_{j+1})$, where $1 \le i \le j \le k$).
2. A *B-hyperpath*[1] Π_{st} from s to t in $\mathcal{H}$ is defined in terms of the notion of *B-connection* in directed hypergraphs. A B-connection to a vertex s is recursively defined as follows: (i) a vertex s is B-connected to itself, and (ii) if there is a hyperedge e such that all the vertices in $T(e)$ are B-connected to s, then every vertex in $H(e)$ is B-connected to s. A B-hyperpath[2] from s to t in $\mathcal{H}$ is a minimal (with respect to deletion of vertices and hyperedges) subhypergraph of $\mathcal{H}$ such that t is B-connected to s. A B-hyperpath is acyclic or cycle-free if it does not include any simple cycle within itself.
3. An *F-hyperpath* Π_{st} from s to t in $\mathcal{H}$ is a subhypergraph of $\mathcal{H}$ such that $\overline{\Pi_{st}}$ is a B-hyperpath from t to s in $\overline{\mathcal{H}}$.
4. A *BF-hyperpath* from s to t in $\mathcal{H}$ is subhypergraph that is both a B-hyperpath and an F-hyperpath from s to t.

In Fig. 1 (part a), $\Pi_{v_1 v_4} = (v_1, e_2, v_5, e_4, v_7, e_3, v_4)$ is a simple path and $\Pi_{v_1 v_3} = (v_1, e_2, v_5, e_4, v_7, e_3, v_3)$ is a simple cycle in $\mathcal{H}_1$ (since $v_3 \in T(e_2) = \{v_1, v_3\}$). In Fig. 1 (part b), directed hypergraph $\mathcal{G}$ with $V(\mathcal{G}) = \{v_1, v_2, v_3, v_5, v_6, v_8, v_9\}$ and $E(\mathcal{G}) = \{e_1, e_2, e_4, e_6\}$ is a B-hyperpath from v_1 to v_9 in $\mathcal{H}_2$, directed hypergraph $\mathcal{J}$ with $V(\mathcal{J}) = \{v_3, v_4, v_6, v_7, v_8, v_9\}$ and $E(\mathcal{J}) = \{e_3, e_5, e_6\}$ is an F-hyperpath from v_3 to v_9 in $\mathcal{H}_2$, and directed hypergraph $\mathcal{K}$ with $V(\mathcal{K}) = \{v_6, v_7, v_8, v_9\}$ and $E(\mathcal{K}) = \{e_5, e_6\}$ is a BF-hyperpath from v_7 to v_9 in $\mathcal{H}_2$. Note that there is no B-hyperpath from v_1 to v_7 in $\mathcal{H}_2$ (since v_7 is not B-connected to v_1 in $\mathcal{H}_2$) and that there is no F-hyperpath from v_1 to v_8 in $\mathcal{H}_2$ (since v_1 is not B-connected to v_8 in $\overline{\mathcal{H}_2}$).

The definition of B-hyperpath in Definition 2.3 (part 2) is consistent with the notion of *folded hyperpath* given in the papers [7,8]. We can alternatively define a B-hyperpath in terms of a sequence of hyperedges used to prove the B-connection of vertices in the hyperpath. This definition of B-hyperpath appeared in [4].[3]

Definition 2.4 (*Alternative Definition of B-Hyperpath [4]*). Let $\mathcal{H} = (V, E)$ be a directed hypergraph and let $s, t \in V(\mathcal{H})$. A *B-hyperpath* from s to t in $\mathcal{H}$ is a minimal directed subhypergraph (with respect to deletion of vertices and hyperedges) $\mathcal{H}'$ of $\mathcal{H}$ with the property that the hyperedges of $\mathcal{H}'$ can be ordered in a sequence $(e_1, e_2, \ldots, e_k)$ such that, for every $e_i \in E(\mathcal{H}')$, it holds that $T(e_i) \subseteq \{s\} \cup H(e_1) \cup \cdots \cup H(e_{i-1})$ and $t \in H(e_k)$.

Even though the notions of B- and F-hyperpaths help to capture problems in several different problem domains (see, e.g., [10, 12]), there are other problem domains for which these definitions do not seem to be the right one. We present two such problem domains in Section 3.

3. *L*-hyperpaths

3.1. Definition and relationship with other notions of hyperpaths

The notions of hyperpaths (B-, F-, and BF-hyperpaths) defined by Gallo et al. [10] differ from the notion of (directed) paths in directed graphs in that, roughly speaking, the hyperpaths are not required to be "linear". By that we mean that while a path in a directed graph is an alternating sequence of vertices and edges, a (B-, F-, or BF-) hyperpath may not have this form. Although the definition of a simple path (Definition 2.3, part 1) requires linearity, that definition is too loose to capture the structural richness of directed hypergraphs. For instance, to determine all vertices v connected to a vertex u via a simple path in a directed hypergraph $\mathcal{H}$, it suffices to perform a variant of the breadth first search algorithm for directed graphs wherein the neighbors of a vertex w of $\mathcal{H}$ are considered to be exactly those vertices that belong to the heads of the outgoing hyperedges from w (see [10]). Similarly, the linearity requirement in the alternative definition of B-hyperpaths (Definition 2.4) is too loose. Indeed, consider two directed hypergraphs $\mathcal{H}_1$ and $\mathcal{H}_2$ on vertices $\{s, v_1, v_2, \ldots, v_n, t\}$. $\mathcal{H}_1$ has $n+1$ hyperedges $e_1, e_2, \ldots, e_{n+1}$, where, for $1 \le i \le n$, $e_i = (\{s\}, \{v_i\})$, and $e_{n+1} = (\{v_1, v_2, \ldots, v_n\}, \{t\})$. $\mathcal{H}_2$ has $n+1$ hyperedges $e_1, e_2, \ldots, e_{n+1}$, where, $e_1 = (\{s\}, \{v_1\})$, for each $2 \le i \le n$, $e_i = (\{v_{i-1}\}, \{v_i\})$, and $e_{n+1} = (\{v_1, v_2, \ldots, v_n\}, \{t\})$. Consider the sequence $(e_1, e_2, \ldots, e_n, e_{n+1})$. Clearly, this sequence defines a B-hyperpath from s to t in both $\mathcal{H}_1$ and $\mathcal{H}_2$. Thus, in both $\mathcal{H}_1$ and $\mathcal{H}_2$, s and t are connected by a B-hyperpath containing $n+1$ hyperedges and $n+2$ vertices.

[1] Gallo et al. [10] gave a topological characterization of B-hyperpaths as follows: A B-hyperpath Π_{st} from s to t in $\mathcal{H}$ is a minimal (with respect to deletion of vertices and hyperedges) subhypergraph $\mathcal{H}' = (V', E')$ of $\mathcal{H}$ such that $s, t \in V'$ and, for each $x \in V' - \{s\}$, there is a simple cycle-free path from s to x in $\mathcal{H}'$. However, Nielsen and Pretolani [24] later pointed out that this topological characterization of B-hyperpaths is not consistent with the commonly interpreted definition of B-hyperpath given in terms of the notion of B-connection. They also pointed out that, for acyclic (i.e., cycle-free) B-hyperpaths, the topological characterization given by Gallo et al. [10] is consistent with the notion of B-connection.

[2] Ausiello et al. [7] and Ausiello, Italiano, and Nanni [8] considered a generalization of B-hyperpath wherein a B-hyperpath is defined as a hyperpath $\Pi_{S,t}$ from any given set S of vertices to some vertex t in the hypergraph. They defined a hypercycle $\Pi_{S,t}$ as a nonempty (consisting of at least one hyperedge) B-hyperpath where $t \in S$.

[3] Actually, Ausiello, Franciosa, and Frigioni [4] gave this alternative definition for a generalization of B-hyperpath, considered also in the papers [7,8], in which a B-hyperpath is defined from a set S of vertices to some vertex t.

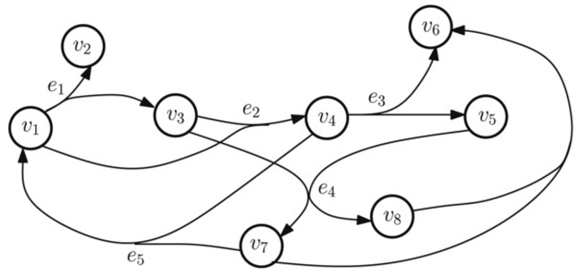


Fig. 2. A directed hypergraph $\mathcal{H}$ with an L-hypercycle. The arrows on a hyperedge in the figure point to the vertices in the head of the hyperedge.

However, the nature of connections in these two hyperpaths is different. The hyperpath in $\mathcal{H}_1$ is "bushy" whereas that in $\mathcal{H}_2$ is "long".

In this section, we first introduce a notion of hyperconnection called an L-hyperpath. We then relate L-hyperpaths to previously studied notions of directed hyperpaths.

Definition 3.1. Let $\mathcal{H} = (V, E)$ be a directed hypergraph and let $s, t \in V(\mathcal{H})$.

1. An *L-hyperpath* Π_{st} from s to t in $\mathcal{H}$ is a sequence $(v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1})$ consisting of distinct vertices v_i, where $1 \leq i \leq k+1$, and distinct hyperedges e_j, where $1 \leq j \leq k$, such that $s = v_1$, $t = v_{k+1}$, and for every $1 \leq i \leq k$, it holds that $v_i \in T(e_i)$, $v_{i+1} \in H(e_i)$, and $T(e_i) \subseteq \{s\} \cup H(e_1) \cup \cdots \cup H(e_{i-1})$.
2. An *L-hypercycle* Π in $\mathcal{H}$ is a sequence $(v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1})$ consisting of distinct vertices v_i, where $1 \leq i \leq k$, and distinct hyperedges e_j, where $1 \leq j \leq k$, such that $v_1 = v_{k+1}$, and for every $1 \leq i \leq k$, it holds that $v_i \in T(e_i)$, $v_{i+1} \in H(e_i)$, and $T(e_i) \subseteq \{v_1\} \cup H(e_1) \cup \cdots \cup H(e_{i-1})$.

For any L-hyperpath $\Pi = (v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1})$ from vertex v_1 to vertex v_{k+1}, let $\mathcal{H}_\Pi$ be defined as the subhypergraph of $\mathcal{H}$ such that $V(\mathcal{H}_\Pi) = \{v_1\} \cup H(e_1) \cup H(e_2) \cup \cdots \cup H(e_k)$ and $E(\mathcal{H}_\Pi) = \{e_1, e_2, \ldots, e_k\}$. We say that $\mathcal{H}_\Pi$ is the *hypergraph representation* of Π. In Fig. 2, $\Pi_1 = (v_1, e_1, v_3, e_2, v_4, e_3, v_5, e_4, v_7)$ is an L-hyperpath and $\Pi_2 = (v_1, e_1, v_3, e_2, v_4, e_3, v_5, e_4, v_7, e_5, v_1)$ is an L-hypercycle in $\mathcal{H}$. Note that there is no L-hyperpath from v_4 to v_1 in $\mathcal{H}$ and that the hypergraph representation of Π_1 is $\mathcal{H}' = (V(\mathcal{H}), \{e_1, e_2, e_3, e_4\})$.

Let $e = (T(e), H(e))$ be a hyperedge in an L-hyperpath Π_{st} from vertex s to vertex t in a directed hypergraph $\mathcal{G}$. Then at least one vertex from $T(e)$ and at least one vertex from $H(e)$ must be in Π_{st}. Also each element of $T(e)$ must be present in $\{s\} \cup (\cup_{e' \prec e} H(e'))$, where the second union is over all hyperedges e' that precede e in Π_{st}. There might be elements of $T(e) \cup H(e)$ that are not present in Π_{st}. However, Theorem 3.2 states that if $\mathcal{G}$ contains only B-hyperedges, then each element of $T(e) \cup H(e)$ is in Π_{st}.

Theorem 3.2. *Let $\mathcal{G}$ be a directed B-hypergraph. Let Π be an L-hyperpath in $\mathcal{G}$ and let $e = (T(e), H(e))$ be a directed hyperedge in Π. Then each $v \in T(e) \cup H(e)$ is present in Π.*

Since the proof of this result is straightforward, we omit the proof.

The main difference between our notion of hyperpath and those defined previously is that we impose an ordering on the vertices and hyperedges in an L-hyperpath. Changing the order of the vertices and hyperedges of an L-hyperpath might result in a different L-hyperpath or it might result in a sequence that is not a legal L-hyperpath. It is thus natural to ask whether we can (nontrivially) permute the vertices and hyperedges so that the resulting sequence is also a legal L-hyperpath. Theorem 3.4 shows that given a set of vertices and a set of hyperedges of a directed B-hypergraph, there is at most one L-hyperpath that can be formed by linearly arranging all the vertices and hyperedges.

Before we state and prove Theorem 3.4, we show a useful property of L-hyperpaths over a fixed set of vertices and a fixed set of hyperedges in a directed B-hypergraph. In particular, Theorem 3.3 shows that if Π and Π' are distinct L-hyperpaths over the same set of vertices V and the same set of hyperedges E in a directed B-hypergraph, then for any hyperedge $e \in E$, there is a unique vertex $v \in V$ that immediately precedes e in both Π and Π'.

Theorem 3.3. *Let Π and Π' be L-hyperpaths over the same set of vertices V and the same set of hyperedges E in a directed B-hypergraph. Then, for any hyperedge $e \in E$, the vertex immediately preceding e in Π is the same as the vertex immediately preceding e in Π'.*

Proof. Let $\Pi = (v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1})$ be an L-hyperpath from vertex v_1 to vertex v_{k+1}. We will show that, for each $i = 1, 2, \ldots, k$, the vertex immediately preceding e_i in Π' is v_i. We will prove this by induction. Consider the base case: $i = 1$. The hyperedge $e_1 = (T(e_1), H(e_1))$ is the first hyperedge in Π. By the definition of an L-hyperpath, $T(e_1) = \{v_1\}$. In Π', the vertex preceding e_1 must be an element of $T(e_1)$. Thus, the vertex immediately preceding e_1 in Π' is v_1. The induction hypothesis is: For each $i = 1, 2, \ldots, \ell$, where $1 \leq \ell < k$, the vertex immediately preceding e_i (in Π') is v_i. Consider $i = \ell+1$. Since Π is an L-hyperpath, the tail of $e_{\ell+1}$, i.e., $T(e_{\ell+1})$, is a subset of $\{v_1\} \cup (\bigcup_{j=1}^{\ell} H(e_j))$. Also, since all the hyperedges e_j are B-hyperedges, it follows that for each $1 \leq j \leq \ell$, $H(e_j) = \{v_{j+1}\}$. Hence, $T(e_{\ell+1})$ is a subset of $\{v_1, v_2, \ldots, v_{\ell+1}\}$. Thus, the vertex immediately preceding $e_{\ell+1}$ in Π' must be one of $v_1, v_2, \ldots, v_{\ell+1}$. However, by the induction hypothesis, each of $v_1, v_2, \ldots, v_\ell$ occurs (in Π') immediately before hyperedges $e_1, e_2, \ldots, e_\ell$, respectively. Since each vertex can immediately precede at most one hyperedge in an L-hyperpath, the vertex immediately preceding $e_{\ell+1}$ in Π' must be $v_{\ell+1}$. This completes the proof of Theorem 3.3. ■ (Theorem 3.3)

It is crucial for the statement of Theorem 3.3 that the underlying directed hypergraph is a B-hypergraph. Consider, for instance, a directed hypergraph $\mathcal{H}$ consisting of vertices v_1, v_2, v_3, and v_4, and hyperedges $e_1 = (\{v_1\}, \{v_2, v_3\})$, $e_2 = (\{v_2, v_3\}, \{v_4\})$, and $e_3 = (\{v_4\}, \{v_2, v_3\})$. The hyperedges e_1 and e_3 are F-hyperedges in $\mathcal{H}$. Let $\Pi = (v_1, e_1, v_2, e_2, v_4, e_3, v_3)$ and $\Pi' = (v_1, e_1, v_3, e_2, v_4, e_3, v_2)$. Clearly, both Π and Π' are legal L-hyperpaths over the same set of vertices $\{v_1, v_2, v_3, v_4\}$ and the same set of hyperedges $\{e_1, e_2, e_3\}$. However, the vertex immediately preceding e_2 in Π is v_2, whereas the vertex immediately preceding e_2 in Π' is v_3. Thus, we see that the statement of Theorem 3.3 does not hold if the underlying directed hypergraph is not a B-hypergraph.

We are now ready to show that in directed B-hypergraphs, any set of vertices and any set of hyperedges can form at most one L-hyperpath.

Theorem 3.4. *Let $\mathcal{H} = (V, E)$ be a directed B-hypergraph. Let $U \subseteq V$ and let $F \subseteq E$. Then there is at most one L-hyperpath in $\mathcal{H}$ with the set of vertices exactly U and with the set of hyperedges exactly F.*

Proof. Let $\Pi = (v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1})$ be an L-hyperpath from vertex v_1 to vertex v_{k+1} in $\mathcal{H}$. Let Π' be an L-hyperpath with the vertex set $\{v_1, v_2, \ldots, v_{k+1}\}$ and the hyperedge set $\{e_1, e_2, \ldots, e_k\}$. We will show that the order of hyperedges in Π' is $e_1, e_2, \ldots, e_k$.

Assume that the order of hyperedges in Π' is not $e_1, e_2, \ldots, e_k$. Let e_i and e_j be such that e_i immediately precedes e_j in Π' and $j \neq i + 1$. However, since v_j immediately precedes e_j in Π, it follows from Theorem 3.3 that v_j also immediately precedes e_j in Π'. Thus, we have that v_j immediately follows e_i in Π'. On the other hand, v_{i+1} immediately follows e_i in Π. Since $j \neq i + 1$, we have that the head of e_i must contain both v_{i+1} and v_j, which is a contradiction since e_i is a B-hyperedge. Thus, the order of hyperedges in Π' is $e_1, e_2, \ldots, e_k$. The result now follows from Theorem 3.3. ■ (Theorem 3.4)

Theorem 3.4 gives a polynomial-time greedy algorithm to decide whether a given set of vertices and a given set of B-hyperedges form an L-hyperpath. In the algorithm presented next for this decision problem, V and E denote the set of vertices and B-hyperedges, respectively.

Algorithm

Input: A set of vertices V and a set of B-hyperedges E.

Output: "Yes" if there is an L-hyperpath containing all of V and E, and "No" otherwise.

```
1.  While there is a hyperedge $e \in E$ such that $e$ has exactly one tail vertex and $e$ was not
    considered previously do
        ▷ Since $e$ is a $B$-hyperedge, it has exactly one head vertex.
2.      Set $v$ to be the unique head vertex of $e$.
3.      Mark $e$ and the head and tail vertices of $e$.
4.      Repeat the following until all the hyperedges are marked:
            (a) Select a hyperedge $e$ such that $v \in T(e)$, all vertices of $T(e)$ are marked, and
            the head vertex of $e$ is unmarked. If no such hyperedge exists, move to the next
            iteration of the while loop.
            (b) Update $v$ to be the unique head vertex of $e$.
            (c) Mark $e$ and the head vertex of $e$.
5.      End Repeat
6.      If all the vertices are marked, output "Yes".
7.  End While
8.  Output "No".
```

Proof of correctness of the algorithm. Suppose that there is no L-hyperpath containing all of V and E. Then, clearly the algorithm outputs "No". If there is an L-hyperpath $\Pi = (v_1, e_1, v_2, \ldots, v_k, e_k, v_{k+1})$ consisting of $V = \{v_1, v_2, \ldots, v_{k+1}\}$ and $E = \{e_1, e_2, \ldots, e_k\}$, then e_1 is chosen as the starting hyperedge in one of the iterations of the while loop (step 1). Suppose that the choice of hyperedges considered up to certain repetitions of step 4 (the repeat loop) is consistent with the sequence $e_1, e_2, \ldots, e_r$, where $1 \leq r < k$. (Meaning that the algorithm selects hyperedges in the order $e_1, e_2, \ldots, e_r$.) We claim that the choice of the next hyperedge in step 4(a) is uniquely determined (that is why a greedy strategy suffices). To see this, suppose that in the next iteration of step 4(a), there are two choices e_{r+1} and e_s, where $s \neq r + 1$, of hyperedges available. Since the hyperedges in E are B-hyperedges, the set of marked vertices at the start of this iteration is $\{v_1, v_2, \ldots, v_{r+1}\}$. Thus, since e_s is a possible choice of a hyperedge, $T(e_s) \subseteq \{v_1, v_2, \ldots, v_{r+1}\}$. On the other hand, since e_s occurs later than the hyperedges $e_1, e_2, \ldots, e_{r+1}$ in Π, $T(e_s)$ must contain a vertex w that occurs later than the vertices $v_1, v_2, \ldots, v_{r+1}$ in Π. That is, there must be a vertex $w \in T(e_s)$ such that $w \notin \{v_1, v_2, \ldots, v_{r+1}\}$. This gives a contradiction. ■

Let us return to our previous example in which we showed that the statement of Theorem 3.3 depends crucially on whether the underlying directed hypergraph is a B-hypergraph. In that example, we defined distinct and legal L-hyperpaths $\Pi = (v_1, e_1, v_2, e_2, v_4, e_3, v_3)$ and $\Pi' = (v_1, e_1, v_3, e_2, v_4, e_3, v_2)$ over the same set of vertices $\{v_1, v_2, v_3, v_4\}$ and the same set of hyperedges $\{e_1, e_2, e_3\}$. Thus, we see that the statement of Theorem 3.4 fails to hold if the underlying directed hypergraph is not a B-hypergraph.

L-hyperpaths inherit the linearity property from simple paths and the B-connection property from B-hyperpaths. It is thus easy to see that if there exists an L-hyperpath from u to v in a directed hypergraph $\mathcal{H}$, then there also exist a simple path from u to v and a B-hyperpath from u to v in $\mathcal{H}$. The converse is false. That is, the presence of a simple path and a B-hyperpath does not imply the presence of an L-hyperpath. As a simple example, consider the B-hypergraph with vertices $\{1, 2, 3, 4\}$ and hyperedges $(\{1\}, \{2\})$, $(\{1\}, \{3\})$, and $(\{2, 3\}, \{4\})$. There exists a simple path from 1 to 4. There also exists a B-hyperpath from 1 to 4, but there exists no L-hyperpath from 1 to 4. It is natural to ask if we can get a sufficient condition for the existence of an L-hyperpath in terms of the existence of a B-hyperpath and/or a simple path. Theorem 3.5 establishes a relationship among the notions of L-hyperpaths, B-connection, and simple paths in directed B-hypergraphs.

Theorem 3.5. *Let $\mathcal{H}$ be a directed B-hypergraph, $\mathcal{G}$ be a subhypergraph of $\mathcal{H}$, and $s, t \in V(\mathcal{G})$ be distinct vertices. Then, the following holds: $\mathcal{G}$ is the hypergraph representation of an L-hyperpath Π_{st} from s to t if and only if $\mathcal{G}$ is a minimal (w.r.t. deletion of vertices and hyperedges) subhypergraph of $\mathcal{H}$ such that t is B-connected to s in $\mathcal{G}$ and there is a simple elementary cycle-free path from s to t that consists of all the hyperedges of $\mathcal{G}$.*

Proof. For the proof of the left-to-right implication, let $\Pi_{st} = (v_1, e_1, \ldots, v_k, e_k, v_{k+1})$ be an L-hyperpath from s to t in $\mathcal{H}$, where $v_1 = s$ and $v_{k+1} = t$. Let $\mathcal{G}$ be the hypergraph representation of Π_{st}. Clearly, Π_{st} is a simple elementary path from s to t that consists of all the hyperedges of $\mathcal{G}$ and $\mathcal{G}$ is a minimal directed subhypergraph of $\mathcal{H}$ such that t is B-connected to s in $\mathcal{G}$. We now show that Π_{st} is cycle-free. Assume to the contrary that there is a subpath $\Pi' = (v_i, e_i, \ldots, v_j, e_j, v_{j+1})$ of Π_{st} such that $v_{j+1} \in T(e_i)$ and $j \geq i$. Then, by the definition of an L-hyperpath, $v_{j+1} \in T(e_i) \subseteq \{v_1\} \cup H(e_1) \cup \cdots \cup H(e_{i-1})$. Since $\mathcal{H}$ is a directed B-hypergraph, it holds that for every hyperedge $e \in E(\mathcal{H})$, we have $|H(e)| = 1$. In particular, for every $1 \leq k \leq i-1$, we have $H(e_k) = \{v_{k+1}\}$. Thus, $v_{j+1} \in \{v_1, \ldots, v_i\}$. This contradicts our assumption that Π_{st} is an L-hyperpath.

We now give the proof of the right-to-left implication. Let $\mathcal{G}$ be a minimal subhypergraph of $\mathcal{H}$ such that t is B-connected to s in $\mathcal{G}$. Let $\Pi = (v_1, e_1, \ldots, v_k, e_k, v_{k+1})$ be a simple elementary cycle-free path such that $v_1 = s$, $v_{k+1} = t$, and $\{e_1, e_2, \ldots, e_k\} = E(\mathcal{G})$. We first show in Claim 3.6 that Π is an L-hyperpath from s to t.

Claim 3.6. *Π is an L-hyperpath from s to t.*

Proof of Claim 3.6. From Definitions 2.3 (part 1) and 3.1, it suffices to show that for every $1 \leq i \leq k$, $T(e_i) \subseteq \{s\} \cup H(e_1) \cup \cdots \cup H(e_{i-1})$. Assume to the contrary that j is the least index for which $T(e_j) \nsubseteq \{s\} \cup H(e_1) \cup \cdots \cup H(e_{j-1})$, where $1 \leq j \leq k$. Then there is a vertex $w \in T(e_j)$ such that $w \notin \{s\} \cup H(e_1) \cup \cdots \cup H(e_{j-1})$. Since $\mathcal{G}$ is a minimal subhypergraph such that t is B-connected to s in $\mathcal{G}$, as in Definition 2.4, the hyperedges of $\mathcal{G}$ can be ordered in a sequence $(e_{i_1}, e_{i_2}, \ldots, e_{i_k})$ such that for every $e_{i_\ell} \in E(\mathcal{G})$, $T(e_{i_\ell}) \subseteq \{s\} \cup H(e_{i_1}) \cup \cdots \cup H(e_{i_{\ell-1}})$ and $t \in H(e_{i_k})$. In particular, e_j occurs in this sequence. Therefore, there exists a minimum index r, where $1 \leq r \leq k$, such that $w \in H(e_r)$. Since $\mathcal{H}$ is a directed B-hypergraph, $w \in H(e_r) = \{v_{r+1}\}$. Note that $e_r \notin \{e_1, e_2, \ldots, e_{j-1}\}$, since otherwise we get a contradiction with the assumption that $w \notin \{s\} \cup H(e_1) \cup \cdots \cup H(e_{j-1})$. Thus, $e_r \in \{e_j, e_{j+1}, \ldots, e_k\}$. Therefore, there is simple subpath $\Pi' = (v_j, e_j, v_{j+1}, e_{j+1}, \ldots, v_r, e_r, w)$ of Π in $\mathcal{G}$. Since $w \in T(e_j)$, Π' is a simple cycle. This gives a contradiction because Π is cycle-free. Hence, Claim 3.6 is proved. ■ (Claim 3.6)

Since Π is an L-hyperpath from s to t that consists of all the hyperedges of $\mathcal{G}$ and since $\mathcal{G}$ is a minimal (w.r.t. deletion of vertices and hyperedges) subhypergraph of $\mathcal{H}$ such that t is B-connected to s in $\mathcal{G}$, $\mathcal{G}$ must be the hypergraph representation $\mathcal{H}_\Pi$ of Π. This completes the proof of Theorem 3.5. ■ (Theorem 3.5)

Finally, we would like to point out a subtlety in the definition of L-hyperpaths. Consider a directed hypergraph with vertices $\{1, 2, 3, 4\}$ and hyperedges $e_1 = (\{1\}, \{2, 3\})$ and $e_2 = (\{2, 3\}, \{4\})$. According to our definition, this hypergraph contains two *distinct* L-hyperpaths between vertices 1 and 4, namely $(1, e_1, 2, e_2, 4)$ and $(1, e_1, 3, e_2, 4)$. However, note that these two distinct hyperpaths start and end at the same vertex and go through the same sequence of hyperedges. We could have defined L-hyperpaths to consider these two hyperpaths equivalent. It would be interesting to explore the relationship between our definition and this related but different definition. We leave this exploration as a direction for future research.

3.2. Further generalizations of directed hypergraphs and L-hyperpaths

Directed hypergraphs are generalizations of directed graphs and L-hyperpaths are generalizations of directed paths. We note that further generalizations of directed hypergraphs and L-hyperpaths are interesting both from theoretical and practical perspectives. In this subsection, we mention these generalizations both as motivation for our work and as areas for future work. In both these generalizations, a hyperpath is an alternating sequence of vertices and hyperedges (as defined earlier). However, in these generalizations, we will have additional requirements on which vertex–hyperedge sequences are legal.

3.2.1. *Asymmetric directed hypergraphs*

In our definition of L-hyperpaths, we allow a hyperedge $e = (T(e), H(e))$ to follow a vertex v if $v \in T(e)$ and each vertex in $T(e)$ occurs either as a source vertex of the L-hyperpath or in the heads of hyperedges preceding e in the L-hyperpath. Thus, e can legally follow *any* vertex $v \in T(e)$ as long as $T(e) \subseteq \{s\} \cup (\bigcup_{e_i \prec e} H(e_i))$, where s is the source vertex of the L-hyperpath and $e_i \prec e$ denotes that e_i precedes e in the sequence given by the L-hyperpath. One might, however, require a hyperedge $e = (T(e), H(e))$ to follow only a specific vertex $v \in T(e)$. For example: This generalization is needed in Section 3.3 when considering the Routing problem in which the forwarding policies need not be symmetric. Let us formalize this notion.

Definition 3.7. A *directed hypergraph $\mathcal{H}$ with asymmetric hyperedges* (or simply, a *directed asymmetric hypergraph*) is a pair (V, E), where V is a finite set of vertices, and $E \subseteq V \times 2^V \times 2^V$ is a finite set of hyperedges such that for every hyperedge $e = (v, T(e), H(e)) \in E$, it holds that $v \in T(e)$, $H(e) \neq \emptyset$, and $T(e) \cap H(e) = \emptyset$.

Note that for any hyperedge $e = (v, T(e), H(e))$, v represents the only vertex that can precede e. We now define the notion of an L-hyperpath in a directed asymmetric hypergraph.

Definition 3.8. An *L-hyperpath Π_{st}* from vertex s to vertex t in a directed asymmetric hypergraph $\mathcal{H} = (V, E)$ is a sequence $(v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1})$ consisting of distinct vertices v_i (for $1 \le i \le k+1$) and distinct hyperedges e_j (for $1 \le j \le k$) such that:

1. $v_1 = s$,
2. $v_{k+1} = t$,
3. for each $1 \le i \le k$, if $e_i = (w, T(e_i), H(e_i))$, then $v_i = w \in T(e_i)$ and $v_{i+1} \in H(e_i)$,
4. for each $1 \le i \le k$, if $e_i = (w, T(e_i), H(e_i))$, then $T(e_i) \subseteq \{s\} \cup H(e_1) \cup H(e_2) \cup \cdots \cup H(e_{i-1})$.

Suppose we are given a directed asymmetric hypergraph $\mathcal{H}$ and we want to answer L-hyperpath reachability questions on $\mathcal{H}$. We might want to convert $\mathcal{H}$ into a directed (symmetric) graph $\mathcal{H}'$ such that vertices/hyperedges in $\mathcal{H}'$ correspond in some natural manner to vertices/hyperedges in $\mathcal{H}$. Here is such a transformation from $\mathcal{H}$ to $\mathcal{H}'$. Initially, there are no vertices or hyperedges in $\mathcal{H}'$; we will add these step-by-step. First, add all vertices of $\mathcal{H}$. Now, for each hyperedge $e = (v, T, H)$, add a new vertex u_e and add a (symmetric) hyperedge $d_e = (\{v\}, \{u_e\})$. Now, we will add a hyperedge e' corresponding to e. Let $T' = (T - \{v\}) \cup \{u_e\}$. ($T'$ is formed by replacing v with u_e in T.) Add the (symmetric) hyperedge $e' = (T', H)$. The following claim has an easy proof.

Claim 3.9. *$\Pi = (v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1})$ is an L-hyperpath from vertex v_1 to vertex v_{k+1} in the directed asymmetric hypergraph $\mathcal{H}$ if and only if $\Pi' = (v_1, d_{e_1}, u_{e_1}, e'_1, v_2, d_{e_2}, u_{e_2}, e'_2, \ldots, v_k, d_{e_k}, u_{e_k}, e'_k, v_{k+1})$ is an L-hyperpath from vertex v_1 to vertex v_{k+1} in the directed (symmetric) hypergraph $\mathcal{H}'$.*

3.2.2. *Directed permutation hypergraphs*

A further generalization of directed hypergraphs is one in which a hyperedge with tail vertices T and head vertices H is allowed to be taken only if the set of vertices in T have occurred in the path in a *specific order*. We formalize this notion as a *directed permutation hypergraph*. In the definition below, we use the notation $\mathcal{P}(X)$ to denote the set $\{Y \mid Y$ is an ordered subset of elements from $X\}$.

Definition 3.10. A *directed permutation hypergraph $\mathcal{H}$* is a pair (V, E), where V is a finite set of vertices and $E \subseteq \mathcal{P}(V) \times 2^V$ is a finite set of hyperedges such that, for every $e = (T(e), H(e)) \in E$, it holds that $T(e) \neq \emptyset$, $H(e) \neq \emptyset$, and $T(e) \cap H(e) = \emptyset$.

We can now define the notion of L-hyperpaths for directed permutation hypergraphs. We will need the following notation first: Let P_1 and P_2 be ordered sets. We say that P_1 and P_2 are *consistent* if the order (in P_1 and P_2) of the elements in the set $P_1 \cap P_2$ are the same. Thus, (1, 2, 3) is consistent with (4, 1, 5, 3) because the order of the common elements (namely, 1 and 3) is the same in both the ordered sets.

Definition 3.11. An *L-hyperpath Π_{st}* from vertex s to vertex t in a directed permutation hypergraph $\mathcal{H} = (V, E)$ is a sequence $(v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_{k+1})$ consisting of distinct vertices v_i (for $1 \le i \le k+1$) and distinct hyperedges e_j (for $1 \le j \le k$) such that:

1. $v_1 = s$,
2. $v_{k+1} = t$,
3. for each $1 \le i \le k$, if $e_i = (T(e_i), H(e_i))$, then $v_i \in T(e_i)$ and $v_{i+1} \in H(e_i)$,
4. for each $1 \le i \le k$, if $e_i = (T(e_i), H(e_i))$, then $T(e_i) \subseteq \{s\} \cup H(e_1) \cup H(e_2) \cup \cdots \cup H(e_{i-1})$,
5. for each $1 \le i \le k$, if $e_i = (T(e_i), H(e_i))$, then $T(e_i)$ and $(v_1, v_2, \ldots, v_{k+1})$ are consistent.

Let $\mathcal{H}$ be a directed permutation hypergraph. Just as in the directed asymmetric hypergraph case, we might want to convert $\mathcal{H}$ into a directed (symmetric) hypergraph $\mathcal{H}'$ such that vertices/hyperedges in $\mathcal{H}'$ correspond in some natural manner to vertices/hyperedges in $\mathcal{H}$. We leave this as an open problem.

3.3. Motivation

The study of L-hyperpaths is interesting from a theoretical point of view because they are a natural extension of directed paths in directed graphs, and also a restriction of simple paths and of B-hyperpaths in directed hypergraphs. The study of the cyclomatic number of hypergraphs is of fundamental significance (e.g., [9,1]). Therefore, it is worth studying the complexity of computing the L-cyclomatic number of directed hypergraphs. On the practical side, we show that some interesting questions in problem domains such as distributed secret sharing and routing in packet filtered networks can be modeled using the notion of L-hyperpaths. The linearity constraint of L-hyperpaths turns out to be crucial in correctly modeling problems in these domains.

1. Distributed secret sharing: In traditional secret sharing,[4] a secret key $\mathcal{S}$ is shared among n parties with shares $\mathcal{S}_1$, $\mathcal{S}_2$, ... $\mathcal{S}_n$, respectively, such that certain specified subsets (say, $X_1, X_2, \ldots, X_p$) of the parties can cooperate to compute $\mathcal{S}$ and certain other specified subsets (say, $Y_1, Y_2, \ldots, Y_r$) of the parties cannot determine $\mathcal{S}$. As a concrete example, consider the (k, n)-threshold scheme of Shamir [30], for any $k \leq n$. In this scheme, a secret key $\mathcal{S}$ (assumed to be an integer without loss of generality) is divided into n shares $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ such that any k of these shares allows one to easily reconstruct $\mathcal{S}$ but no $k - 1$ of them reveal anything about $\mathcal{S}$. The basic idea is to build a random $k - 1$ degree univariate polynomial $p(x) = a_0 + a_1x + \cdots + a_{k-1}x^{k-1}$ over a finite field $\mathbb{F}$ such that $a_0 = \mathcal{S}$ and the coefficients $a_1, a_2, \ldots, a_{k-1}$ are chosen uniformly at random from $\mathbb{F}$. Define $\mathcal{S}_i = p(i)$, for each $1 \leq i \leq n$. Given any such k shares, it is easy to compute the coefficients of $p(x)$ using interpolation, and hence to determine $\mathcal{S}$, which equals a_0. On the other hand, if less than k shares are known, then it can be shown that all possible choices of a_0 are equally likely. In other words, knowledge of fewer than k shares does not suffice to compute $\mathcal{S}$. Therefore, this scheme is a (k, n)-threshold scheme.

Now consider the secret sharing problem in a distributed setting: a directed network of n vertices through which a packet containing certain share of a secret key $\mathcal{S}$ is to be transmitted from vertex s to vertex t so that at the end of the packet transmission, t has enough shares to recover $\mathcal{S}$. The secret key $\mathcal{S}$ is assumed to be shared among the n vertices using the (k, n)-threshold scheme described above. Each vertex i knows its share $\mathcal{S}_i$ but has no information about any other shares of $\mathcal{S}$ before the packet transmission begins. When a vertex gets the packet containing certain shares of $\mathcal{S}$, it attaches its own share to the packet and forwards the packet along to a neighboring vertex. The packet transmission ends when the packet reaches the destination vertex t. Upon receiving the packet, the vertex t knows the shares contained in the packet. So, it could use them and its own share to retrieve the secret key $\mathcal{S}$. Note that if the packet goes through at most k vertices including s and t, then any adversary who intercepts packet transmission learns nothing about $\mathcal{S}$, the secret key. We say that a path from s to t is (k, n)-*feasible* if it is a simple path and if it goes through at most k vertices of the network. If a (k, n)-feasible path goes through exactly k vertices of the network (counting s and t), then we say that the path is (k, n)-*optimal*. Clearly, if there is a (k, n)-optimal path from s to t, then s can successfully transmit the packet to t, and t has enough information to recover the secret key. Furthermore, in the case of a packet transmission along a (k, n)-feasible path, the node t would know that no adversary has any information about the secret key.

Notice that if a vertex v follows a vertex u in a (k, n)-feasible path, then v learns u's share of the secret key $\mathcal{S}$. In the situation considered above, we were assuming that all the n vertices trust each other. In a more realistic setting the vertices do not fully trust each other. In particular, we might have a vertex u that is willing to reveal its share of $\mathcal{S}$ only if some other node v has already revealed its share of $\mathcal{S}$. (This will ensure that v does not know u's share since v cannot appear again in any simple path.) In this scenario, a (k, n)-feasible path from s to t is a simple path of at most k vertices with constraints of the form "v must occur before u". We show next that the question of the existence of a (k, n)-feasible path from s to t is exactly the question of the existence of an L-hyperpath of at most $2k$ vertices in a directed B-hypergraph.

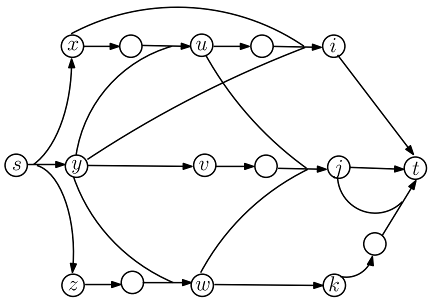
To see this equivalence, we give the following reduction: Let $G = (V, E)$ be the original directed network. Let s and t be the source and destination vertices, respectively. For each vertex u, let P_u be the set of vertices that must occur before u in any (k, n)-feasible path. Consider the directed asymmetric B-hypergraph $\mathcal{H} = (V, E')$, where for each edge $(u, v) \in E$, the set E' contains the hyperedge $(u, P_v \cup \{u\}, \{v\})$. Note that this hyperedge can be taken from u to v if and only if the constraint for v is satisfied (i.e., all vertices in P_v have been visited before). It is easy to show that there is a (k, n)-feasible path from s to t in G if and only if there is an L-hyperpath containing at most k vertices from s to t in $\mathcal{H}$. It follows from Claim 3.9 that the existence of a (k, n)-feasible path in a network is equivalent to the existence of an L-hyperpath with at most $2k$ vertices (counting the end vertices) in a directed (symmetric) hypergraph.

2. Routing in a packet filtered network: This problem is similar to the Distributed Secret Sharing problem but it differs in motivation. Packet filtering [20,21,32] is the mechanism of forwarding or routing packets based on the *type* of the packet. This mechanism is central to access control in firewalls, differentiated quality of service, policy-based routing, traffic billing, etc. Roughly speaking, a packet filter classifies a packet according to the content of the packet. The *information* that the filter uses for classification and the *action* that it takes depend on the nature of the filter. For example, an access control filter would use the source and destination address to filter packets and it could either forward the packet or drop it.

[4] Much work has been done in secret sharing schemes and reviewing all this work is beyond the scope of this paper. However, we refer the interested reader to an excellent survey of secret sharing schemes [31]. To the best of our knowledge, the application mentioned here has not been discussed in earlier papers.

Vertex	Routing History	Target Set
s	$\{\}$	$\{x, y, z\}$
x	$\{y\}$	$\{u\}$
y	$\{\}$	$\{v\}$
z	$\{y\}$	$\{w\}$
u	$\{x, y\}$	$\{i\}$
v	$\{u, w\}$	$\{j\}$
w	$\{\}$	$\{k\}$
i	$\{\}$	$\{t\}$
j	$\{\}$	$\{t\}$
k	$\{j\}$	$\{t\}$

a



b

Fig. 3. (a) shows the filtering policies of a network N, and (b) shows the directed (symmetric) hypergraph $\mathcal{H}$ corresponding to N. The labeled vertices of $\mathcal{H}$ corresponds to the vertices of N and the unlabeled vertices of $\mathcal{H}$ are vertices added to make $\mathcal{H}$ a symmetric directed hypergraph. Note that a packet containing shares of a secret key can be routed from vertex s to vertex t in N because there is an L-hyperpath from s to t in $\mathcal{H}$.

Consider a network consisting of packet filters whose classification mechanism is based both on the intended destination and on the route a packet takes to reach the filter. Thus, we can think of the "class" of a packet as being its *routing history*. Why would we be interested in such a classification? One intuitive motivation stems from the fact that we can look upon the path history of a packet as encoding partial routing table information of the preceding vertices. Thus, a filter that routes packets based on path history of the packet may be able to route packets better. A packet filter upon getting a packet, looks at the path history of the packet and the intended destination of the packet, and decides which of its neighbors to forward the packet to.

We can model the network and the filtering policies of its vertices by a directed asymmetric hypergraph $\mathcal{H}$. The vertex set of $\mathcal{H}$ contains all the vertices of the network. A typical filtering policy of a vertex k of the network is as follows: *If the routing history of the packet contains vertices $i_1, i_2, \ldots, i_r$, then forward the packet to all vertices $j_1, j_2, \ldots, j_s$, otherwise drop the packet.* We can represent this filtering policy by a directed asymmetric hyperedge $(k, \{k\} \cup \{i_1, i_2, \ldots, i_r\}, \{j_1, j_2, \ldots, j_s\})$. It is easy to show that a packet can be routed from a vertex s to a vertex t in the network if and only if there is an L-hyperpath from s to t in the directed asymmetric hypergraph $\mathcal{H}$. It follows from Claim 3.9 that the question of whether a packet can be routed from s to t in a network is also equivalent to the question of whether there is an L-hyperpath between two given vertices in a directed (symmetric) hypergraph.

Fig. 3(a) shows the filtering policies of a packet filtered network and Fig. 3(b) shows the corresponding directed (symmetric) hypergraph used to model this problem.

4. Computational problems related to directed hyperpaths

In this section, we study the computational complexity of several computational problems related to directed hyperpaths (hyperpaths of the kind previously studied and L-hyperpaths). Several of these computational problems are optimization problems, though we consider here decision versions of these problems. We prove that the decision versions of these optimization problems related to L-hyperpaths are NP-complete. First, though, we describe common metrics that have been defined previously and extend them to L-hyperpaths.

4.1. Metrics for directed hyperpaths

Many applications of graphs require one to associate a cost (or weight) on the edges of the graph. The cost of a path in a graph is then defined to be the sum of the cost of edges in the path. In contrast, since the structure of a hyperpath is more complicated, a number of measures on hyperpaths in a directed hypergraph are defined and studied in the literature [19,3,14,23,29,7,8]. We observe that the measures defined for previously studied notions of directed hyperpaths are applicable also for L-hyperpaths if the directed hypergraph representation of an L-hyperpath is considered in the definition. This indicates that the notion of L-hyperpaths is robust and it suggests that L-hyperpaths may be used to model a variety of problems that require these measures on hyperpaths.

In Definition 4.1, we define the notion of a weighted directed hypergraph and in Definition 4.2, we describe measures on directed hyperpaths that have been previously studied. Note that in Definition 4.2, we look upon an L-hyperpath $\Pi = (v_1, e_1, \ldots, v_k, e_k, v_{k+1})$ in terms of its directed hypergraph representation $\Pi(V, E)$, where $V = \{v_1\} \cup H(e_1) \cup H(e_2) \cup \cdots \cup H(e_k)$ and $E = \{e_1, e_2, \ldots, e_k\}$. (Here for notational convenience, we use $\Pi(V, E)$, instead of our conventional notation $\mathcal{H}_\Pi$, to denote the directed hypergraph representation of Π.) For $X \in \{B, L\}$, we say that Π is an acyclic X-hyperpath if Π does not include any simple cycle within itself.

Definition 4.1 (*[7,8]*). A weight function on a directed hypergraph $\mathcal{H} = (V, E)$ maps each $e \in E$ to a nonnegative integer.

Definition 4.2 (*See Also [7,8]*). Let $X \in \{B, L\}$. Let $\mathcal{H}(V, E)$ be a directed hypergraph, W be a weight function on $\mathcal{H}$, and s and t be distinct vertices of $\mathcal{H}$.

1. The *number of hyperedges* $n(\Pi_{st})$ of an X-hyperpath Π_{st} from s to t is the cardinality of the set of hyperedges in Π_{st}. That is,
$$n(\Pi_{st}) = |E(\Pi_{st})|.$$
2. The *cost* $c(\Pi_{st})$ of an X-hyperpath Π_{st} from s to t is the sum of the weights of the hyperedges in Π_{st}. That is,
$$c(\Pi_{st}) = \sum_{e \in E(\Pi_{st})} W(e).$$
3. The *size* $s(\Pi_{st})$ of an X-hyperpath Π_{st} from s to t is defined as follows:
$$s(\Pi_{st}) = \sum_{e \in E(\Pi_{st})} (|T(e)| + |H(e)|).$$
4. The *rank* $r(\Pi_{st})$ of an acyclic X-hyperpath Π_{st} from s to t is inductively defined as follows:
   (a) If there is no hyperedge in Π_{st}, then $r(\Pi_{st}) = 0$.
   (b) If e is a hyperedge in Π_{st} such that $t \in H(e)$, $T(e) = \{w_1, \ldots, w_k\}$, and for every $1 \le i \le k$, Π_{sw_i} is a subhypergraph of Π_{st}, then
$$r(\Pi_{st}) = W(e) + \max_{w_i \in T(e)} r(\Pi_{sw_i}).$$

For an F-hyperpath Π_{st} from s to t, any of the aforementioned measures of Π_{st} is defined to be the same measure of the B-hyperpath $\overline{\Pi_{st}}$ from t to s.

4.2. Computational problems

For any $X \in \{B, F, L\}$ and for any measure function μ_X on X-hyperpaths of $\mathcal{H}$, we define the following decision problems related to directed hyperpaths:

- X-HYPERPATH $= \{\langle \mathcal{H}, s, t\rangle \mid \mathcal{H}$ is a directed hypergraph that contains an X-hyperpath Π_{st} from vertex s to vertex $t\}$.
- μ_X-OPT-HYPERPATH $= \{\langle \mathcal{H}, s, t, k\rangle \mid \mathcal{H}$ is a directed weighted hypergraph that contains an X-hyperpath Π_{st} from vertex s to vertex t such that $\mu_X(\Pi_{st}) \le k\}$.
- L-HYPERCYCLE $= \{\langle \mathcal{H}\rangle \mid \mathcal{H}$ is a directed hypergraph that contains an L-hypercycle$\}$.

Theorem 4.3 states some known results related to the different notions of hyperpaths in directed hypergraphs.

Theorem 4.3. *The following results are known:*

1. *Both B-*HYPERPATH *and F-*HYPERPATH *are* P*-complete* [10].
2. μ_B-OPT-HYPERPATH *(μ_F-*OPT-HYPERPATH*) is* NP*-complete when μ_B (respectively, μ_F) is one of the following measure functions:* (a) *number of hyperedges $n(\cdot)$,* (b) *cost $c(\cdot)$, and* (c) *size $s(\cdot)$* [14,7].
3. μ_B-OPT-HYPERPATH *(μ_F-*OPT-HYPERPATH*) is solvable in deterministic polynomial-time when the measure function μ_B (respectively, μ_F) is the rank $r(\cdot)$ function* [7,8].

4.3. The computational complexity of L-HYPERPATH

We now obtain complexity results related to L-hyperpaths. In Theorem 4.4, we show that L-HYPERPATH is NP-hard by giving a polynomial-time many-one reduction from a known NP-complete problem, HAMILTONIAN PATH, to L-HYPERPATH. The problem HAMILTONIAN PATH is the following: Given an instance $\langle G, w_1, w_2\rangle$ where G is a directed graph and $w_1, w_2 \in V(G)$, does G contain a Hamiltonian path from w_1 to w_2 (i.e., a path from w_1 to w_2 that visits each vertex of G exactly once)?

Theorem 4.4. *For every $k \ge 2$,* L-HYPERPATH *is* NP*-complete even when restricted to k-directed B-hypergraphs.*

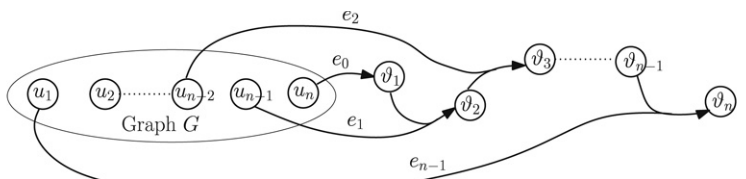


Fig. 4. The directed hypergraph $\mathcal{H}$ constructed from the graph G in the reduction given in Theorem 4.4.

Proof. Clearly, L-HYPERPATH is in NP. So, it only remains to prove that L-HYPERPATH is NP-hard.

Let $\langle G, w_1, w_2\rangle$ be an instance of HAMILTONIAN PATH. Let $n = |V(G)|$ and let $u_1, \ldots, u_n$ be arbitrary labels on the vertices of G such that $u_1 = w_1$ and $u_n = w_2$. We define a polynomial-time function σ as follows: $\sigma(\langle G, u_1, u_n\rangle) = \langle \mathcal{H}, u_1, \vartheta_n\rangle$, where $\vartheta_1, \ldots, \vartheta_n$ are distinct vertices such that $\{\vartheta_1, \ldots, \vartheta_n\} \cap \{u_1, \ldots, u_n\} = \emptyset$, $V(\mathcal{H}) = V(G) \cup \{\vartheta_1, \ldots, \vartheta_n\}$, and $E(\mathcal{H}) = E(G) \cup \{e_0, e_1, \ldots, e_{n-1}\}$, where $e_0 = (\{u_n\}, \{\vartheta_1\})$, and for each $1 \le j \le n-1$, $e_j = (\{u_{n-j}, \vartheta_j\}, \{\vartheta_{j+1}\})$. That is, we construct $\mathcal{H}$ from G by adding n vertices $\vartheta_1, \vartheta_2, \ldots, \vartheta_n$ and adding n hyperedges $e_0, e_1, \ldots, e_{n-1}$. (See Fig. 4 for the construction of $\mathcal{H}$.)

We will now formally prove that σ is a many-one reduction from HAMILTONIAN PATH to L-HYPERPATH. This would immediately imply that L-HYPERPATH is NP-hard.

Assume that $\langle G, u_1, u_n\rangle \in$ HAMILTONIAN PATH. Then, there is a Hamiltonian path $(u_1, e'_{i_1}, u_{i_2}, e'_{i_2}, \ldots, e'_{i_{n-1}}, u_n)$ in G. It is easy to see that $(u_1, e'_{i_1}, u_{i_2}, e'_{i_2}, \ldots, e'_{i_{n-1}}, u_n, e_0, \vartheta_1, e_1, \vartheta_2, e_2, \ldots, \vartheta_{n-1}, e_{n-1}, \vartheta_n)$ is an L-hyperpath in $\mathcal{H}$. This finishes one direction of the proof.

For the other direction of the proof, we assume that $\langle \mathcal{H}, u_1, \vartheta_n\rangle \in$ L-HYPERPATH. Let Π be an L-hyperpath from u_1 to ϑ_n in $\mathcal{H}$. For vertices/hyperedges x and y, let "$x \prec y$ in Π" denote that x precedes y in Π. (Note that $x \prec y$ does *not* necessarily mean that x *immediately precedes* y.) We finish this direction of the proof via the following claims.

Claim 4.5. *Let u_i and ϑ_j be arbitrary vertices that are present in Π. Then $u_i \prec \vartheta_j$ in Π.*

Proof. Say $\vartheta_j \prec u_i$. Then there are vertices ϑ_k and u_ℓ such that u_ℓ is the vertex immediately following ϑ_k in Π. However, this means that there is a hyperedge $e = (T(e), H(e))$ in $\mathcal{H}$ such that $\vartheta_k \in T(e)$ and $u_\ell \in H(e)$. This is a contradiction because no such hyperedge exists in $\mathcal{H}$. This completes the proof of Claim 4.5. ■ (Claim 4.5)

Claim 4.6. *The following statements hold:*

1. *Π contains all vertices u_i of G.*
2. *$\vartheta_1 \prec \vartheta_2 \prec \cdots \prec \vartheta_n$ in Π.*
3. *The vertex immediately preceding ϑ_1 in Π is u_n.*

Proof. Π contains ϑ_n. The only hyperedge into ϑ_n is e_{n-1}. Thus, Π contains e_{n-1}. Since $\mathcal{H}$ contains only B-hyperedges, Theorem 3.2 applies. Thus, each tail vertex of e_{n-1} (that is, ϑ_{n-1} and u_1) is in Π. Furthermore, we can infer that $\vartheta_{n-1} \prec e_{n-1} \prec \vartheta_n$ in Π.

Applying the above argument for ϑ_{n-1}, we obtain that ϑ_{n-2} and u_2 are in Π and $\vartheta_{n-2} \prec \vartheta_{n-1}$ in Π. Continuing this argument, we obtain that u_n is in Π and $u_n \prec \vartheta_1$. Furthermore, since e_0 has exactly one tail vertex, the vertex immediately preceding ϑ_1 in Π is u_n.

Putting this all together, we obtain that each u_i is in Π and that $\vartheta_1 \prec \vartheta_2 \prec \cdots \prec \cdots \vartheta_n$ in Π. This completes the proof of Claim 4.6. ■ (Claim 4.6)

It now follows from the above claims that the first n vertices of Π are from the set $\{u_1, u_2, \ldots, u_n\}$. Furthermore, by definition, the first vertex is u_1. From Claim 4.5, each u_i precedes each ϑ_j. Thus, from Claim 4.6 (part 3) it follows that among u_i's, u_n is the last vertex in Π. Since each u_i is in Π, it follows that the first n vertices are the vertices from $\{u_1, u_2, \ldots, u_n\}$, the first being u_1, and the nth being u_n. Since the only edges among these vertices are the edges from G, it follows that there is a Hamiltonian path from u_1 to u_n in G. This finishes the other direction of the proof. Thus, we get that σ is a polynomial-time many-one reduction from HAMILTONIAN PATH to L-HYPERPATH.

In the above reduction, notice that the construction of $\mathcal{H}$ involves only B-hyperedges, that all B-hyperedges of $\mathcal{H}$ have tails of size at most 2, and that there are B-hyperedges of $\mathcal{H}$ whose tails have size exactly 2. Hence, it follows that for each $k \ge 2$, L-HYPERPATH is NP-complete even when restricted to k-directed B-hypergraphs. This completes the proof of Theorem 4.4. ■ (Theorem 4.4)

It is interesting to ask whether L-HYPERPATH is also NP-complete when restricted to 1-directed hypergraphs. Notice that under this restriction the problem L-HYPERPATH is the same as the computational problem of deciding whether there is a (directed) path between two given vertices of a given directed graph, which is NL-complete [16,15]. Thus, this restriction of L-HYPERPATH cannot be NP-complete unless NL = NP.

The reduction used to show that L-HYPERPATH is NP-complete in Theorem 4.4 can be easily modified to obtain the following corollary:

Corollary 4.7. *For every $k \geq 2$ and for any measure function $\mu_L \in \{$number of hyperedges $n(\cdot)$, cost $c(\cdot)$, size $s(\cdot)$, rank $r(\cdot)\}$ on L-hyperpaths, the problem μ_L-*OPT-HYPERPATH *is* NP-*complete even when restricted to k-directed B-hypergraphs.*

The proof of Theorem 4.4 does not seem to easily give an NP-hardness result for L-HYPERCYCLE since the input graph G may contain a directed cycle in itself. However, we state in Theorem 4.8 below that L-HYPERCYCLE is NP-complete. We refer the reader to Lemma 5.17 (proved later in this paper), which can be used to give a proof of Theorem 4.8.

Theorem 4.8. *For every $k \geq 2$,* L-HYPERCYCLE *is* NP-*complete even when restricted to k-directed B-hypergraphs.*

Finally, we mention that the problems L-HYPERPATH and μ_L-OPT-HYPERPATH can be easily shown to be in P when restricted to directed F-hypergraphs. Thus, the intrinsic hardness of these problems is mainly due to the presence of B-hyperedges.

5. The *L*-cyclomatic number of a directed hypergraph

The cyclomatic number of a hypergraph is the minimum number of hyperedges that need to be deleted so that the resulting hypergraph has no hypercycle. For a connected graph $G = (V, E)$, the cyclomatic number is given by $|E| - |V| + 1$. The cyclomatic number of any undirected hypergraph is also given by an efficiently computable expression (see [1]). On the other hand, it is NP-complete to decide given a *directed graph* G and an integer k whether the cyclomatic number of G is at most k [17].

For directed hypergraphs, the notion of cyclomatic number can be defined as follows.

Definition 5.1. Given a directed hypergraph $\mathcal{H} = (V, E)$, the *L-cyclomatic number* of $\mathcal{H}$ is the following: $\min\{k \in \mathbb{N} \mid (\exists B \subseteq E)[|B| = k$ and there are no L-hypercycles in $(V, E - B)]\}$. A set $B \subseteq E$ is an L-cyclomatic set of $\mathcal{H}$ if B is a minimum cardinality set such that there are no L-hypercycles in $(V, E - B)$.

Acharya [1] studied the connection between the cyclomatic number and the planarity of an undirected hypergraph. In this section, we study the computational complexity of several decision problems related to the L-cyclomatic number of a directed hypergraph.

5.1. *Computational problems*

Definition 5.2. We define the following computational problems:

1. CYCLOMATIC-SET $= \{\langle \mathcal{H}, B\rangle \mid \mathcal{H} = (V, E)$ is a directed hypergraph and $B \subseteq E$ such that $\mathcal{H}' = (V, E - B)$ has no L-hypercycle$\}$.
2. CYCLOMATIC-NUMBER $= \{\langle \mathcal{H}, k\rangle \mid \mathcal{H} = (V, E)$ is a directed hypergraph such that there exists a set $B \subseteq E$, $|B| \leq k$, and $\langle \mathcal{H}, B\rangle \in$ CYCLOMATIC-SET$\}$.
3. MIN-CYCLOMATIC-SET $= \{\langle \mathcal{H}, B\rangle \mid \langle \mathcal{H}, B\rangle \in$ CYCLOMATIC-SET and for each B' such that $|B'| < |B|$, $\langle \mathcal{H}, B'\rangle \notin$ CYCLOMATIC-SET$\}$.
4. MINIMAL-CYCLOMATIC-SET $= \{\langle \mathcal{H}, B\rangle \mid \langle \mathcal{H}, B\rangle \in$ CYCLOMATIC-SET and for each B' such that $B' \subsetneq B$, $\langle \mathcal{H}, B'\rangle \notin$ CYCLOMATIC-SET$\}$.

Clearly CYCLOMATIC-SET is coNP-complete, since for any directed hypergraph $\mathcal{H}$, $\langle \mathcal{H}\rangle \in$ L-HYPERCYCLE $\iff \langle \mathcal{H}, \emptyset\rangle \notin$ CYCLOMATIC-SET. In Section 5.5 we prove that CYCLOMATIC-NUMBER is Σ_2^p-complete and in Section 5.6 we prove that MIN-CYCLOMATIC-SET is Π_2^p-complete. To prove these results, we give polynomial-time many-one reductions from ∃∀CNF-UNSAT, a problem known to be Σ_2^p-complete [33,39], to the relevant Σ_2^p problems. Next, we prove in Section 5.7 that the problem MINIMAL-CYCLOMATIC-SET is complete for the complexity class DP, the second level of the boolean hierarchy over NP. For the proof of this result, we give a polynomial-time many-one reduction from SAT-UNSAT, a problem known to be DP-complete [27], to MINIMAL-CYCLOMATIC-SET.

We now define the problems ∃∀CNF-UNSAT and SAT-UNSAT.

5.2. *The problems* ∃∀CNF-UNSAT *and* SAT-UNSAT

Notations: Let {T, F} denote the truth-value set, where T stands for *true* and F stands for *false*. For any set X of boolean variables, a truth-value assignment function α of X is a mapping from X to {T, F}. Let X and Y be two disjoint sets of boolean variables and let $\phi(X, Y)$ be a boolean CNF formula on $X \cup Y$. For each truth-value assignment α of X, let $\phi(\alpha, Y)$ denote the simplified CNF formula obtained from $\phi(X, Y)$ by substituting each $x \in X$ by $\alpha(x)$. Additionally, for each truth-value assignment β of Y, let $\phi(\alpha, \beta)$ denote the truth-value obtained by substituting $\beta(y)$ for each $y \in Y$ in $\phi(\alpha, Y)$. We call the literals x and $\overline{x}$, where the variable $x \in X$, X-literals. Similarly, we call y and $\overline{y}$, where the variable $y \in Y$, Y-literals.

We next define the computational problem ∃∀CNF-UNSAT. Theorem 5.4 states that this problem is complete for Σ_2^p, the second level of the polynomial hierarchy. In Section 5.5, we will give a polynomial-time many-one reduction ∃∀CNF-UNSAT to CYCLOMATIC-NUMBER to prove the Σ_2^p-hardness of the latter problem. We will also give a polynomial-time many-one reduction from ∃∀CNF-UNSAT to the complement of MIN-CYCLOMATIC-SET in Section 5.6 to prove that MIN-CYCLOMATIC-SET is Π_2^p-hard.

Definition 5.3. An instance of the problem ∃∀CNF-UNSAT is given by $\langle X, Y, \phi\rangle$, where X and Y are disjoint sets of variables and ϕ is a boolean CNF formula involving variables of X and Y. The instance $\langle X, Y, \phi\rangle$ is in ∃∀CNF-UNSAT if and only if there exists a truth-value assignment α of X such that for all truth-value assignments β of Y, it holds that $\phi(\alpha, \beta) = \mathrm{F}$.

Theorem 5.4 (*[33,39]*). ∃∀CNF-UNSAT *is* Σ_2^p*-complete.*

The computational problem SAT-UNSAT is defined below. Theorem 5.6 states that this problem is complete for DP, the second level of the boolean hierarchy. We will give a polynomial-time many-one reduction from SAT-UNSAT to MINIMAL-CYCLOMATIC-SET in Section 5.7 to prove that the latter problem is DP-hard.

Definition 5.5 (*[27]*). An instance of the problem SAT-UNSAT is given by $\langle\phi_1, \phi_2\rangle$, where $\phi_1(X)$ and $\phi_2(Y)$ are boolean CNF formulas involving variables from disjoint sets X and Y, respectively. The membership of any instance $\langle\phi_1, \phi_2\rangle$ in SAT-UNSAT is given by:

$$\langle\phi_1, \phi_2\rangle \in \text{SAT-UNSAT} \iff (\phi_1 \in \text{SAT and } \phi_2 \notin \text{SAT}).$$

Theorem 5.6 (*[27]*). SAT-UNSAT *is* DP*-complete.*

We next describe an interesting construction of a directed hypergraph from any boolean CNF formula $\phi(X, Y)$ over disjoint sets X and Y of variables. This construction will be used in the proofs of computational hardness results of CYCLOMATIC-NUMBER, MIN-CYCLOMATIC-SET, and MINIMAL-CYCLOMATIC-SET.

5.3. A directed hypergraph construction from any boolean CNF *formula* $\phi(X, Y)$

Let $X = \{x_1, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ be disjoint sets of variables. Let $\langle X, Y, \phi\rangle$ be a boolean CNF formula involving variables of X and Y. For the construction described in this section, we will assume that each variable (of ϕ) appears in any clause (of ϕ) at most once and that each variable (of ϕ) appears at least once over all the clauses. We do not loose any generality of our construction with these assumptions since every boolean CNF formula $\phi(X, Y)$ can be easily transformed into one that satisfies these assumptions. Let the clauses of ϕ be $\phi_1, \phi_2, \ldots \phi_s$, and for $i \in \{1, \ldots, s\}$, let p_i denote the number of occurrences of variables of Y in ϕ_i. Also, for each $1 \le i \le s$ and $1 \le j \le p_i$, we use $y_{v(i,j)}$ to denote the jth variable of ϕ_i that belongs to Y. (Thus, if $\phi_i = (\overline{x_1} \vee y_3 \vee \overline{y_7} \vee x_4 \vee \overline{y_8})$, then $y_{v(i,1)} = y_3, y_{v(i,2)} = y_7$, and $y_{v(i,3)} = y_8$). For each $i \le n$, let n_i denote the number of occurrences of y_i (i.e., as y_i or $\overline{y_i}$) in ϕ. We now describe the construction of a directed hypergraph $\mathcal{H}$ from any boolean CNF formula $\phi(X, Y)$ of the kind assumed above.

The construction of $\mathcal{H}$ uses four kinds of gadgets. These are selectors, k-choosers for $k \ge 0$, k-dividers for $k \ge 1$, and switches.

Definition 5.7. The gadgets *selectors*, *choosers*, *dividers*, and *switches* are defined as follows:

1. A *selector* is a directed hypergraph as shown in Fig. 5(a).
2. For every $k \ge 0$, a k*-chooser* is a directed hypergraph as shown in Fig. 5(b).
3. For every $k \ge 1$, a k*-divider* is a directed hypergraph as shown in Fig. 5(c).
4. A *switch* is a directed hypergraph as shown in Fig. 5(d).

In each of the gadgets defined above, there are distinguished vertices and hyperedges that we label as follows. A selector has distinguished vertices labeled A, B, C, D, E, F, G, J and distinguished hyperedges $(\{B\}, \{C\})$, which is labeled 1, and $(\{C\}, \{B\})$, which is labeled 2 (see Fig. 5(a)). A k-chooser, where $k \ge 0$, has distinguished vertices labeled A, B and distinguished hyperedges labeled 1, 2, ..., k (see Fig. 5(b)). Note that a 0-chooser has only distinguished vertices labeled A and B but no hyperedge. A k-divider, where $k \ge 1$, has distinguished vertices labeled A, B, F, F' and distinguished hyperedges labeled $0, 0', 1, 1', \ldots, k, k', (k+1), (k+1)'$ (see Fig. 5(c)). A switch has distinguished vertices labeled $A, B, C, C', D, D', F, F', G, G', K, K'$ and distinguished hyperedges $(\{A\}, \{F\})$, which is labeled 1, $(\{A\}, \{F'\})$, which is labeled $1'$, $(\{F\}, \{G\})$, which is labeled 2, $(\{F'\}, \{G'\})$, which is labeled $2'$, $(\{G\}, \{K\})$, which is labeled 3, $(\{G'\}, \{K'\})$, which is labeled $3'$, $(\{A, K\}, \{B\})$, which is labeled 4, $(\{A, K'\}, \{B\})$, which is labeled $4'$, $(\{C\}, \{F\})$, which is labeled 5, $(\{C'\}, \{F'\})$, which is labeled $5'$, $(\{C, G\}, \{D\})$, which is labeled 6, and $(\{C', G'\}, \{D'\})$, which is labeled $6'$ (see Fig. 5(d)).

Definition 5.8. Let $\mathcal{H} = (V, E)$ be a directed hypergraph. Let u, v, u', v' be vertices of $\mathcal{H}$ and let $e = (\{u\}, \{v\})$ and $e' = (\{u'\}, \{v'\})$ be simple directed edges of $\mathcal{H}$. Let $S = (V_S, E_S)$ be a switch. Then, the directed hypergraph obtained by *placing the switch S between* e *and* e′ is the hypergraph $\mathcal{H}' = (V \cup V_S, (E - \{e, e'\}) \cup E_S \cup \{(\{u\}, \{C\}), (\{D\}, \{v\}), (\{u'\}, \{C'\}), (\{D'\}, \{v'\})\})$, where C, D, C', and D' are the labels of vertices of S (see Fig. 6(a), (b), and (c)).

$\mathcal{H}$ consists of the following gadgets.

1. s choosers $C_1, C_2, \ldots, C_s$, where for $1 \le i \le s$, C_i is a p_i-chooser corresponding to the clause ϕ_i. (Recall that s denotes the number of clauses of ϕ and p_i denotes the number of occurrences of variables of Y in ϕ_i.)
2. n dividers $D_1, D_2, \ldots, D_n$, where for $1 \le i \le n$, D_i is a n_i-divider corresponding to the variable y_i. (Recall that n denotes the number of variables of Y and n_i denotes the number of occurrences of y_i (as y_i or $\overline{y_i}$) in ϕ.)
3. m selectors $L_1, L_2, \ldots, L_m$, where for $1 \le i \le m$, L_i corresponds to the variable x_i. (Recall that m denotes the number of variables of X.)

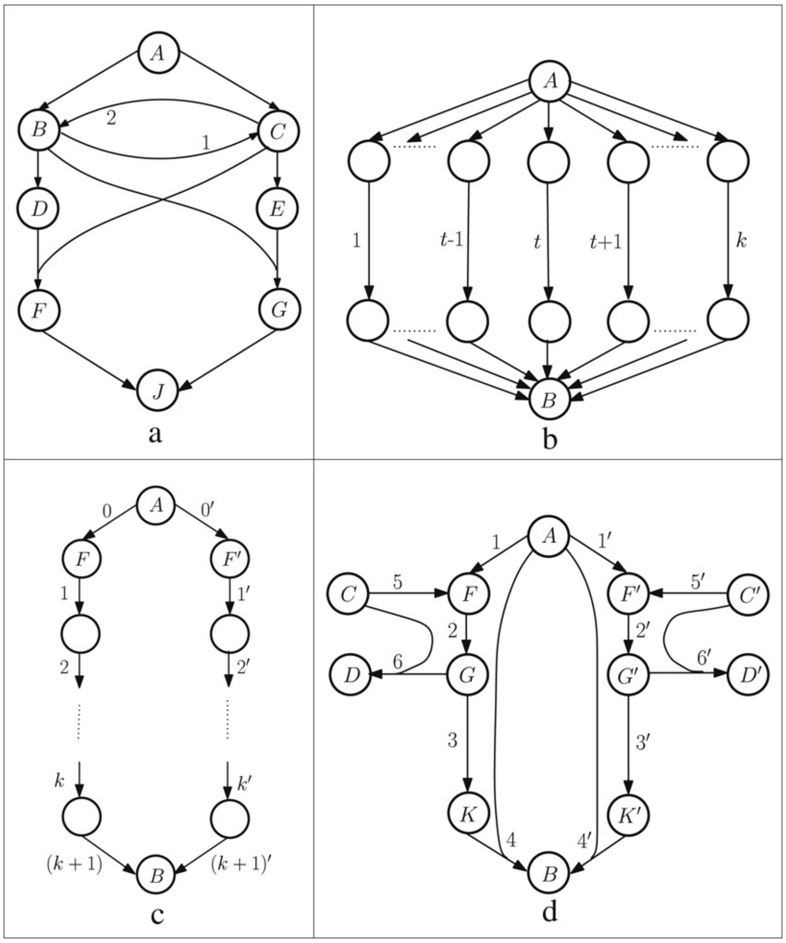


Fig. 5. Gadgets used in the reduction from ∃∀CNF-UNSAT to CYCLOMATIC-NUMBER. (a) A selector. (b) A k-chooser, where $k \geq 0$. (c) A k-divider, where $k \geq 1$. (d) A switch.

4. $\sum_{i=1}^{s} p_i$ switches $S_{1,1}, S_{1,2}, \ldots, S_{1,p_1}, S_{2,1}, \ldots, S_{2,p_2}, \ldots, S_{s,1}, \ldots, S_{s,p_s}$, where for $1 \leq i \leq s$ and $1 \leq j \leq p_i$, $S_{i,j}$ corresponds to the jth variable of ϕ_i that belongs to Y. Note that if for some i, $p_i = 0$, i.e., the clause ϕ_i has no variable of Y, then there is no switch corresponding to the clause ϕ_i.

We use $h(k, j)$ to denote the sum of the number of occurrences of y_j (as y_j or $\overline{y_j}$) in clauses $\phi_1, \ldots, \phi_k$ with $h(0, j) = 0$ for each j. For each switch $S_{i,j}$, where $1 \leq i \leq s$ and $1 \leq j \leq p_i$, let $\mathrm{succ}(S_{i,j})$ be the switch succeeding $S_{i,j}$ in the ordering $S_{1,1}, S_{1,2}, \ldots, S_{1,p_1}, S_{2,1}, \ldots, S_{2,p_2}, \ldots, S_{s,1}, \ldots, S_{s,p_s}$ if $i \neq s$ or $j \neq p_s$, and be undefined if $i = s$ and $j = p_s$. We denote the first and the last switch in this ordering by S_{first} and S_{last}, respectively. For each label x and for each gadget y, we use the shorthand $\text{vertex}_{\mathcal{H}}(x, y)$ ($\text{edge}_{\mathcal{H}}(x, y)$) to denote "the vertex labeled x in gadget y of $\mathcal{H}$" (respectively, "the hyperedge labeled x in gadget y of $\mathcal{H}$").

The vertex set of $\mathcal{H}$ consists exactly of the vertices of the above gadgets. The hyperedge set of $\mathcal{H}$ can be classified into three categories. The first category includes hyperedges of the gadgets of $\mathcal{H}$. We call these hyperedges *gadget hyperedges*. The second category consists of hyperedges needed to connect these gadgets together. We call these hyperedges *connection hyperedges*. The third category consists of hyperedges whose tail sets consist of the vertices of the selectors and the choosers. Since these hyperedges will be used in the proofs to "jump over" the choosers, we refer to these hyperedges as *bypass hyperedges*. The gadget hyperedges have already been defined in the description of the gadgets in Definition 5.7. We now describe the connection hyperedges and the bypass hyperedges used in the construction of $\mathcal{H}$. Fig. 7 shows the directed hypergraph $\mathcal{H}$ when the input instance is $\langle X, Y, \phi(X, Y)\rangle$, where $X = \{x_1, x_2\}$, $Y = \{y_1, y_2, y_3\}$, and $\phi(X, Y) = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee y_1) \wedge (x_2 \vee \overline{y_1} \vee y_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee y_1 \vee \overline{y_2} \vee \overline{y_3})$.

Connection hyperedges

1. **Place a switch between an edge of the chooser C_i corresponding to the occurrence of a variable y_k in ϕ_i and an edge of the divider D_k:** For every $1 \leq i \leq s$ and for every $1 \leq j \leq p_i$, if $y_{\upsilon(i,j)}$ appears as $y_{\upsilon(i,j)}$ in ϕ_i, then place the switch $S_{i,j}$ between the edge j of the p_i-chooser C_i and the edge $h(i, \upsilon(i, j))$ of the divider $D_{\upsilon(i,j)}$. Otherwise, i.e., if $y_{\upsilon(i,j)}$ appears as

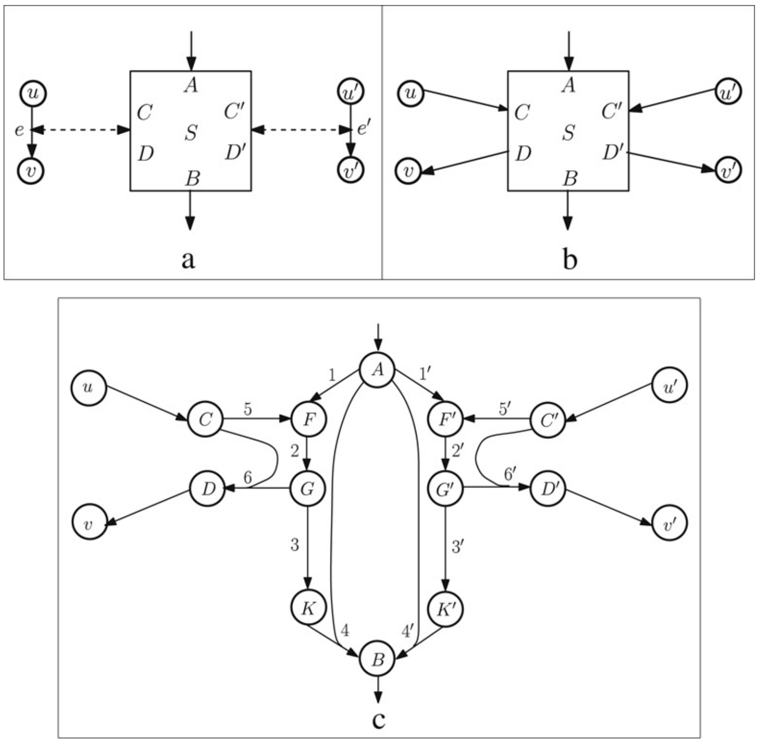


Fig. 6. (a) Schematic representation of placing a switch S (shown as a rectangular box in the figure) between simple directed edges $e = (\{u\}, \{v\})$ and $e' = (\{u'\}, \{v'\})$. (b) The actual placement of a switch S between hyperedges e and e': the hyperedges e and e' are deleted and new hyperedges $(\{u\}, \{C\})$, $(\{D\}, \{v\})$, $(\{u'\}, \{C'\})$ and $(\{D'\}, \{v'\})$ are added. (c) The actual directed subhypergraph obtained after the placement of the switch S between hyperedges e and e'.

$\overline{y_{v(i,j)}}$ in ϕ_i, then place the switch $S_{i,j}$ between the edge j of the p_i-chooser C_i and the edge $h(i, v(i,j))'$ of the divider $D_{v(i,j)}$. These placements are done as explained in Definition 5.8.

2. **Connect the choosers in series:** For every $1 \le i < s$, connect vertex$_{\mathcal{H}}(B, C_i)$ to vertex$_{\mathcal{H}}(A, C_{i+1})$ by a simple directed edge.
3. **Connect the dividers in series:** For every $1 \le j < n$, connect vertex$_{\mathcal{H}}(B, D_j)$ to vertex$_{\mathcal{H}}(A, D_{j+1})$ by a simple directed edge.
4. **Connect the selectors in series:** For every $1 \le k < m$, connect vertex$_{\mathcal{H}}(J, L_k)$ to vertex$_{\mathcal{H}}(A, L_{k+1})$ by a simple directed edge.
5. **Connect the switches in series:** For every $1 \le i \le s$ and $1 \le j \le p_i$, if succ$(S_{i,j})$ is defined then connect vertex$_{\mathcal{H}}(B, S_{i,j})$ to vertex$_{\mathcal{H}}(A, \text{succ}(S_{i,j}))$ by a simple directed edge.
6. **Connect the choosers, dividers, selectors, and switches:** Connect vertex$_{\mathcal{H}}(B, C_s)$ to vertex$_{\mathcal{H}}(A, S_{\text{first}})$, connect vertex$_{\mathcal{H}}(B, S_{\text{last}})$ to vertex$_{\mathcal{H}}(A, L_1)$, connect vertex$_{\mathcal{H}}(J, L_m)$ to vertex$_{\mathcal{H}}(A, D_1)$, and connect vertex$_{\mathcal{H}}(B, D_n)$ to vertex$_{\mathcal{H}}(A, C_1)$ by simple directed edges.

Bypass hyperedges

1. **Connect the selector L_i with the chooser C_j:** For every $1 \le i \le m$ and $1 \le j \le s$, if x_i appears as x_i in ϕ_j, then add the hyperedge $(\{\text{vertex}_{\mathcal{H}}(G, L_i), \text{vertex}_{\mathcal{H}}(A, C_j)\}, \{\text{vertex}_{\mathcal{H}}(B, C_j)\})$. Otherwise, if x_i appears as $\overline{x_i}$ in ϕ_j, then add the hyperedge $(\{\text{vertex}_{\mathcal{H}}(F, L_i), \text{vertex}_{\mathcal{H}}(A, C_j)\}, \{\text{vertex}_{\mathcal{H}}(B, C_j)\})$.

This concludes the construction of $\mathcal{H}$. Notice that $\mathcal{H}$ is a 2-directed B-hypergraph. We will use this construction in Section 5.5 to prove that CYCLOMATIC-NUMBER is Σ_2^p-hard, in Section 5.6 to prove that MIN-CYCLOMATIC-SET is Π_2^p-hard, and in Section 5.7 to prove that MINIMAL-CYCLOMATIC-SET is DP-hard.

5.4. Pivoting operation on a directed hypergraph

Often we will need to impose on a directed hypergraph $\mathcal{H}$ the constraint that every L-hyperpath (or L-hypercycle) in $\mathcal{H}$ must start from a fixed vertex v of $\mathcal{H}$. For this, we define an operation on $\mathcal{H}$ called *pivoting*. The pivoting operation on $\mathcal{H}$ around v basically ensures that the above constraint is satisfied. That is, after the pivoting operation on $\mathcal{H}$ around v, every L-hyperpath (or L-hypercycle) in the new directed hypergraph starts from v; the vertex v is called the *pivot* of the new hypergraph.

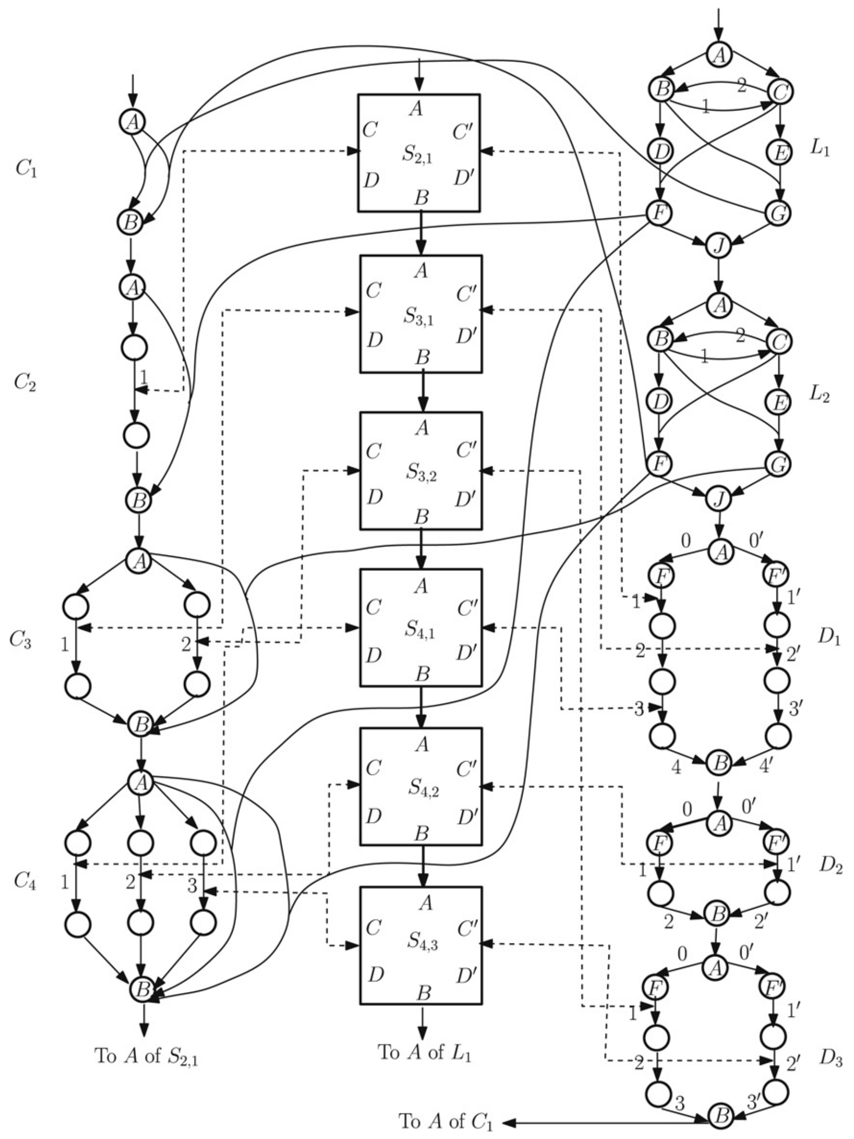


Fig. 7. The directed hypergraph $\mathcal{H}$ obtained from $\langle X, Y, \phi(X, Y)\rangle$ using the construction described in Section 5.3. Here $X = \{x_1, x_2\}$, $Y = \{y_1, y_2, y_3\}$, and $\phi(X, Y) = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee y_1) \wedge (x_2 \vee \overline{y_1} \vee y_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee y_1 \vee \overline{y_2} \vee \overline{y_3})$.

To perform the pivoting operation on $\mathcal{H}$ around v, we make only the following change in $E(\mathcal{H})$: Replace all hyperedges e for which $|T(e)| = 1$ and $v \notin T(e)$ by the hyperedge $(T(e) \cup \{v\}, H(e))$. Since all hyperedges e with $|T(e)| = 1$ of the new hypergraph $\mathcal{H}'$ have $T(e) = \{v\}$, every L-hyperpath or L-hypercycle in $\mathcal{H}'$ must start from v.

Notice that pivoting adds v to the tail set of only those hyperedges of $\mathcal{H}$ that have $|T(e)| = 1$ and $v \notin T(e)$. Therefore, if $\mathcal{H}$ is a k-directed B-hypergraph, for some $k \geq 2$, then the directed hypergraph obtained after the pivoting operation on $\mathcal{H}$ is also a k-directed B-hypergraph.

5.5. *The computational complexity of* CYCLOMATIC-NUMBER

Theorem 5.9. *For every $k \geq 2$,* CYCLOMATIC-NUMBER *is Σ_2^p-complete even when restricted to k-directed B-hypergraphs.*

Proof. It is easy to show that CYCLOMATIC-NUMBER is in Σ_2^p. We give a polynomial-time many-one reduction σ from ∃∀CNF-UNSAT to CYCLOMATIC-NUMBER to prove that CYCLOMATIC-NUMBER is Σ_2^p-hard. From these results, it would immediately follow that CYCLOMATIC-NUMBER is Σ_2^p-complete.

On input $\langle X, Y, \phi\rangle$, σ outputs $\langle \mathcal{H}, m\rangle$, where m equals $|X|$ and $\mathcal{H}$ is a directed hypergraph obtained from $\phi(X, Y)$ via the following steps:

- First, use the construction described in Section 5.3 to obtain a directed hypergraph $\mathcal{H}'$ from $\phi(X, Y)$.

- Next, apply the pivoting operation on $\mathcal{H}'$ around $\text{vertex}_{\mathcal{H}'}(A, L_1)$ but retain the original hyperedges $\text{edge}_{\mathcal{H}'}(1, L_i)$ and $\text{edge}_{\mathcal{H}'}(2, L_i)$ of each selector L_i of $\mathcal{H}'$.

In Lemma 5.13, we prove that if $\langle X, Y, \phi\rangle \in$ ∃∀CNF-UNSAT, then the L-cyclomatic number of $\mathcal{H}$ is m, and in Lemma 5.14, we prove that if $\langle X, Y, \phi\rangle \notin$ ∃∀CNF-UNSAT, then the L-cyclomatic number of $\mathcal{H}$ is $m+1$. These two lemmas together would imply that σ is a polynomial-time many-one reduction from ∃∀CNF-UNSAT to CYCLOMATIC-NUMBER.

Before we state and prove Lemmas 5.13 and 5.14, we mention some properties of the selector gadget that we will use in the proofs of these lemmas. As Proposition 5.10 states, there are basically two simple directed edges in each selector L_i (of $\mathcal{H}$) that form an L-hypercycle within L_i.

Proposition 5.10. *The following statements hold:*

1. *In each selector L_i of $\mathcal{H}$, the sequence*

$$(\text{vertex}_{\mathcal{H}}(B, L_i), \text{edge}_{\mathcal{H}}(1, L_i), \text{vertex}_{\mathcal{H}}(C, L_i), \text{edge}_{\mathcal{H}}(2, L_i), \text{vertex}_{\mathcal{H}}(B, L_i))$$

   is a directed L-hypercycle.
2. *Let E' be an L-cyclomatic set of $\mathcal{H}$. Then, for each $1 \le i \le m$, E' contains at least one of* $\text{edge}_{\mathcal{H}}(1, L_i)$ *and* $\text{edge}_{\mathcal{H}}(2, L_i)$.
3. *The L-cyclomatic number of $\mathcal{H}$ is at least m.*

Proof. Part 1 follows from the definition of the selector gadget. Parts 2 and 3 follow from (a) the fact that $\mathcal{H}$ has exactly m selectors $L_1, L_2, \dots, L_m$ and (b) the statement in Part 1 that each selector L_i has one L-hypercycle comprising the vertices and hyperedges of L_i. ■(Proposition 5.10)

In the proofs of Lemmas 5.13 and 5.14 below, for each selector L_i of $\mathcal{H}$, we associate the removal of the hyperedge $\text{edge}_{\mathcal{H}}(2, L_i)$ with the setting of the variable x_i (of X) to T, i.e., true. On the other hand, for each selector L_i of $\mathcal{H}$, we associate the removal of the hyperedge $\text{edge}_{\mathcal{H}}(1, L_i)$ with the setting of the variable x_i (of X) to F, i.e., false. This association is made explicit in Definition 5.11.

Definition 5.11. For each truth-value assignment α of X, define $\text{remove}(\mathcal{H}, \alpha)$ to be the hypergraph $\mathcal{H}'$ obtained from $\mathcal{H}$ by the removal of $\text{CYCL}(\mathcal{H}, \alpha)$, where $\text{CYCL}(\mathcal{H}, \alpha) =$

$$\bigcup_{i=1}^{m} \{\text{edge}_{\mathcal{H}}(z, L_i) \mid (z = 2 \iff \alpha(x_i) = \text{T}) \wedge (z = 1 \iff \alpha(x_i) = \text{F})\}.$$

Lemma 5.13 shows that if there exists a truth-value assignment α of X such that for each truth-value assignment β of Y, $\phi(\alpha, \beta) = \text{F}$, then there is a set of m hyperedges (corresponding to α) the removal of which will make $\mathcal{H}$ free of L-hypercycles. Lemma 5.14 shows that if for each truth-value assignment α of X, there is some truth-value assignment β of Y such that $\phi(\alpha, \beta) = \text{T}$, then any set of hyperedges the removal of which will make $\mathcal{H}$ free of L-hypercycles must contain at least $m+1$ hyperedges. The reason is that any assignment of truth-values to the variables of X corresponds to the removal of exactly one hyperedge (either $\text{edge}_{\mathcal{H}}(2, L_i)$ or $\text{edge}_{\mathcal{H}}(1, L_i)$) from each selector L_i. However, we show in Lemma 5.12 that if there is a β such that $\phi(\alpha, \beta) = \text{T}$, then there is an L-hypercycle in $\mathcal{H}$ even after the removal of these hyperedges from the selectors. Thus, we need to remove at least one more hyperedge to make $\mathcal{H}$ free of L-hypercycles.

Lemma 5.12. *Fix an arbitrary truth-value assignment α of X. Then the following holds:*

$\phi(\alpha, Y) \in \text{SAT}$ *if and only if there is an L-hypercycle in* $\text{remove}(\mathcal{H}, \alpha)$.

The proof of Lemma 5.12 is deferred to Section 5.8. We next show how we use this lemma to prove Lemmas 5.13 and 5.14.

Lemma 5.13. *Let $|X| = m$. If $\langle X, Y, \phi\rangle \in$ ∃∀CNF-UNSAT, then the L-cyclomatic number of $\mathcal{H}$ is m.*

Proof. From Proposition 5.10 (part 3), we know that the L-cyclomatic number of $\mathcal{H}$ is at least m. Thus, we only need to prove that the L-cyclomatic number of $\mathcal{H}$ is at most m. For this, we show that there is a collection of m hyperedges of $\mathcal{H}$ such that the deletion of these hyperedges from $\mathcal{H}$ makes $\mathcal{H}$ free of L-hypercycles. Let $\langle X, Y, \phi\rangle \in$ ∃∀CNF-UNSAT and let α_0 be a truth-value assignment of X such that $\phi(\alpha_0, Y)$ is unsatisfiable (i.e., for each truth-value assignment β of Y, $\phi(\alpha_0, \beta) = \text{F}$). By applying Lemma 5.12 for $\alpha = \alpha_0$, it follows that there is no L-hypercycle in $\text{remove}(\mathcal{H}, \alpha_0)$. Since $\text{remove}(\mathcal{H}, \alpha_0)$ is obtained from $\mathcal{H}$ by removing exactly m hyperedges, the L-cyclomatic number of $\mathcal{H}$ is at most m. This completes the proof of Lemma 5.13. ■ (Lemma 5.13)

Lemma 5.14. *Let $|X| = m$. If $\langle X, Y, \phi\rangle \notin$ ∃∀CNF-UNSAT, then the L-cyclomatic number of $\mathcal{H}$ is $m+1$.*

Proof. From Proposition 5.10 (part 3), we know that the L-cyclomatic number of $\mathcal{H}$ is at least m. We claim that the L-cyclomatic number of $\mathcal{H}$ must be at least $m+1$. To prove this claim, we show that any collection of m hyperedges of $\mathcal{H}$ cannot constitute an L-cyclomatic set of $\mathcal{H}$.

Assume to the contrary that there is an L-cyclomatic set Z of $\mathcal{H}$ such that $|Z| = m$. Then, it follows from Proposition 5.10 (part 2) that Z contains, for each $1 \le i \le m$, one of $\text{edge}_{\mathcal{H}}(1, L_i)$ and $\text{edge}_{\mathcal{H}}(2, L_i)$. Since $|Z| = m$, Z must contain exactly one hyperedge (out of $\text{edge}_{\mathcal{H}}(1, L_i)$ and $\text{edge}_{\mathcal{H}}(2, L_i)$) of each selector L_i because the number of selector gadgets of $\mathcal{H}$ equals m. Define α_0 to be the *unique* truth-value assignment of X such that $Z = \text{CYCL}(\mathcal{H}, \alpha_0)$. By the hypothesis $\langle X, Y, \phi\rangle \notin \exists\forall$CNF-UNSAT, we know that $\phi(\alpha_0, Y)$ is satisfiable. By applying Lemma 5.12 for $\alpha = \alpha_0$, it follows that there is an L-hypercycle in remove$(\mathcal{H}, \alpha_0)$. This contradicts the assumption that Z is an L-cyclomatic set of $\mathcal{H}$. Hence, the claim is proven.

In fact, it can be seen from the proof of Lemma 5.12 that deleting the hyperedge $(\{\text{vertex}_{\mathcal{H}}(B, D_n), \text{vertex}_{\mathcal{H}}(A, L_1)\}, \{\text{vertex}_{\mathcal{H}}(A, C_1)\})$ breaks any L-hypercycle in remove$(\mathcal{H}, \alpha)$. Thus, it follows that the L-cyclomatic number of $\mathcal{H}$ is at most $m + 1$. The lower and upper bounds on the L-cyclomatic number of $\mathcal{H}$ imply that this number is $m + 1$. This completes the proof of Lemma 5.14. ■ (Lemma 5.14)

In the above reduction, observe that the construction of $\mathcal{H}$ involves only B-hyperedges, that all B-hyperedges have tails of size at most 2, and that there are B-hyperedges whose tails have size exactly 2. So, the above reduction also shows that for every $k \geq 2$, CYCLOMATIC-NUMBER is Σ_2^p-complete even when restricted to k-directed B-hypergraphs. This completes the proof of Theorem 5.9. ■ (Theorem 5.9)

The restriction of CYCLOMATIC-NUMBER to 1-directed hypergraphs is the computational problem of deciding whether there is a set of k edges whose deletion makes a given directed graph free of cycles. The latter problem is called the *feedback arc set* problem, which is NP-complete [17]. Thus, this restriction of CYCLOMATIC-NUMBER is NP-complete.

We next study the computational complexity of MIN-CYCLOMATIC-SET.

5.6. *The computational complexity of* MIN-CYCLOMATIC-SET

Theorem 5.15. *For every $k \geq 2$,* MIN-CYCLOMATIC-SET *is Π_2^p-complete even when restricted to k-directed B-hypergraphs.*

Proof. To see that MIN-CYCLOMATIC-SET is in Π_2^p, consider a nondeterministic polynomial-time oracle Turing machine N with access to the oracle CYCLOMATIC-SET that, on input $\langle \mathcal{H}, B\rangle$, where $\mathcal{H} = (V, E)$, asks a query $\langle \mathcal{H}, B\rangle$ to the oracle, and accepts if the oracle answers "no". Otherwise, if the oracle answers "yes", then it guesses a set $B' \subseteq E$ such that $|B'| < |B|$, asks a query $\langle \mathcal{H}, B'\rangle$ to the oracle, and accepts if and only if the oracle answers "yes". It is easy to see that $L(N^{\text{CYCLOMATIC-SET}})$ is the complement of MIN-CYCLOMATIC-SET, and so MIN-CYCLOMATIC-SET is in Π_2^p.

We now give a polynomial-time many-one reduction σ from $\exists\forall$CNF-UNSAT to the complement of MIN-CYCLOMATIC-SET to prove that MIN-CYCLOMATIC-SET is Π_2^p-hard. On input $\langle X, Y, \phi\rangle$, σ outputs $\langle \mathcal{H}, B\rangle$, where B is defined as follows:

$$B = \left(\bigcup_{i=1}^{m}\{\text{edge}_{\mathcal{H}}(1, L_i)\}\right) \cup \{(\{\text{vertex}_{\mathcal{H}}(B, D_n), \text{vertex}_{\mathcal{H}}(A, L_1)\}, \{\text{vertex}_{\mathcal{H}}(A, C_1)\})\},$$

and $\mathcal{H}$ is a directed hypergraph obtained from $\phi(X, Y)$ via the following steps:

- First, use the construction described in Section 5.3 to obtain a directed hypergraph $\mathcal{H}'$ from $\phi(X, Y)$.
- Next, apply the pivoting operation on $\mathcal{H}'$ around vertex$_{\mathcal{H}'}(A, L_1)$ but retain the original hyperedges edge$_{\mathcal{H}'}(1, L_i)$ and edge$_{\mathcal{H}'}(2, L_i)$ of each selector L_i of $\mathcal{H}'$.

We consider the following two cases:

Case 1: **$\langle X, Y, \phi\rangle \in \exists\forall$CNF-UNSAT.** Then by Lemma 5.13, there exists a set $B' \subseteq E$ such that $|B'| = m$ and B' is an L-cyclomatic set of $\mathcal{H}$. Since $m = |B'| < |B| = m + 1$, $\langle \mathcal{H}, B\rangle \notin$ MIN-CYCLOMATIC-SET.

Case 2: **$\langle X, Y, \phi\rangle \notin \exists\forall$CNF-UNSAT.** Then by Lemma 5.14, the L-cyclomatic number of $\mathcal{H}$ is $m + 1$. Note that B is an L-cyclomatic set of $\mathcal{H}$ and that for no B' such that $|B'| < |B| = m + 1$, B' is an L-cyclomatic set of $\mathcal{H}$. Thus, $\langle \mathcal{H}, B\rangle \in$ MIN-CYCLOMATIC-SET.

The observation we made towards the end of the proof of Theorem 5.9 applies also in the proof of this theorem since we use the same construction of $\mathcal{H}$ in both the proofs. Therefore, the above reduction also shows that for every $k \geq 2$, MIN-CYCLOMATIC-SET is Π_2^p-complete even when restricted to k-directed B-hypergraphs. ■ (Theorem 5.15)

The restriction of MIN-CYCLOMATIC-SET to 1-directed hypergraphs is the computational problem of deciding whether deleting a given set B of edges makes a given directed graph free of cycles but deleting any smaller set B' of edges does not make. It is not hard to prove by using the techniques of [17] that this restriction of MIN-CYCLOMATIC-SET is coNP-complete.

5.7. *The computational complexity of* MINIMAL-CYCLOMATIC-SET

Theorems 5.9 and 5.15 prove that CYCLOMATIC-NUMBER and MIN-CYCLOMATIC-SET are complete for Σ_2^p and Π_2^p, respectively. In fact, the proofs of both these results are similar in that in both the proofs we use the same construction, i.e., the constriction described in Section 5.3. In Theorem 5.16, we prove that the problem MINIMAL-CYCLOMATIC-SET is complete for DP, the second level of the boolean hierarchy over NP. The proof of Theorem 5.16 requires the construction of a directed hypergraph, which is built by a suitable modification of the construction described in Section 5.3. To prove that MINIMAL-CYCLOMATIC-SET is DP-hard, we will give a polynomial-time many-one reduction from SAT-UNSAT, a problem defined in Section 5.2, to MINIMAL-CYCLOMATIC-SET.

Theorem 5.16. *For every $k \geq 2$,* MINIMAL-CYCLOMATIC-SET *is* DP*-complete even when restricted to k-directed B-hypergraphs.*

Proof. To prove that MINIMAL-CYCLOMATIC-SET is in DP, it is sufficient to show that MINIMAL-CYCLOMATIC-SET can be expressed as the intersection of an NP language L_1 and a coNP language L_2. It is easy to see that the following L_1, L_2 satisfy these properties: $L_1 = \{\langle \mathcal{H}, B\rangle \mid (\forall e \in B)[\langle \mathcal{H}, B - \{e\}\rangle \notin \text{CYCLOMATIC-SET}]\}$, and $L_2 = \text{CYCLOMATIC-SET}$.

We now give a polynomial-time many-one reduction σ from SAT-UNSAT to MINIMAL-CYCLOMATIC-SET to prove that MINIMAL-CYCLOMATIC-SET is DP-hard. To define σ, we need the following lemma:

Lemma 5.17. *There is a polynomial-time computable function f such that for any input boolean* CNF *formula ϕ, f outputs $\langle \mathcal{H}, u, v, w\rangle$, where $\mathcal{H}$ is a directed hypergraph and u, v, and w are the vertices of $\mathcal{H}$. The output $f(\phi)$ has the following properties:*

1. *$\mathcal{H}$ has no L-hypercycle,*
2. *There is an L-hyperpath Π_{uv} from u to v in $\mathcal{H}$,*
3. *$\phi \in$ SAT if and only if there is an L-hyperpath Π_{uw} from u to w in $\mathcal{H}$.*

The proof of Lemma 5.17 is deferred to Section 5.9. We will use this lemma to construct a directed hypergraph $(\mathcal{G})$ from any pair $(\langle \phi, \psi\rangle)$ of boolean CNF formulas, which is input to σ. We next define σ.

On any input $\langle \phi, \psi\rangle$, where both ϕ and ψ are boolean CNF formulas, σ applies the polynomial-time computable function f on each of ϕ and ψ. Let $f(\phi) = \langle \mathcal{H}(\phi), u(\phi), v(\phi), w(\phi)\rangle$ and $f(\psi) = \langle \mathcal{H}(\psi), u(\psi), v(\psi), w(\psi)\rangle$. The output of σ on $\langle \phi, \psi\rangle$ is $\langle \mathcal{G}, B\rangle$, where $\mathcal{G}$ is a directed hypergraph and B is a set of hyperedges of $\mathcal{G}$. The directed hypergraph $\mathcal{G}$ and the set B are defined below.

We first define a directed hypergraph $\mathcal{G}'$ and then apply a pivoting operation on $\mathcal{G}'$ to obtain $\mathcal{G}$. The vertex set of $\mathcal{G}'$ consists of the vertices of $\mathcal{H}(\phi)$ and $\mathcal{H}(\psi)$. The hyperedge set of $\mathcal{G}'$ consists of the hyperedges of $\mathcal{H}(\phi)$ and $\mathcal{H}(\psi)$ along with new hyperedges that connect a vertex in one hypergraph to a vertex in the other hypergraph. These new hyperedges are called *interconnection hyperedges*. The interconnection hyperedges comprise three simple directed edges: a simple directed edge e_1 that connects $w(\phi)$ to $u(\psi)$, another simple directed edge e_2 that connects $w(\psi)$ to $u(\phi)$, and a final simple directed edge e_3 that connects $v(\psi)$ to $u(\phi)$. The hypergraph $\mathcal{G}$ is obtained by pivoting on $\mathcal{G}'$ around $u(\phi)$. Thus, the interconnection hyperedges e_1, e_2, and e_3 become $(\{u(\phi), w(\phi)\}, \{u(\psi)\})$, $(\{u(\phi), w(\psi)\}, \{u(\phi)\})$, and $(\{u(\phi), v(\psi)\}, \{u(\phi)\})$, respectively, in $\mathcal{G}$. The set B is the set $\{e_3\}$ containing the hyperedge e_3 of $\mathcal{G}$. Fig. 9 is a schematic diagram that shows the construction of $\mathcal{G}$ from $\mathcal{H}(\phi)$ and $\mathcal{H}(\psi)$.

We finish proving that σ is a many-one reduction via the following lemma.

Lemma 5.18. *For any boolean* CNF *formulas ϕ and ψ, the following holds:*

$$\langle \phi, \psi\rangle \in \text{SAT-UNSAT if and only if } \langle \mathcal{G}, B\rangle \in \text{MINIMAL-CYCLOMATIC-SET}.$$

Proof. We consider the following three exhaustive cases. We show that in all these cases, it holds that $\langle \phi, \psi\rangle \in$ SAT-UNSAT if and only if $\langle \mathcal{G}, B\rangle \in$ MINIMAL-CYCLOMATIC-SET.

Case 1: $\boldsymbol{\phi} \in$ SAT and $\boldsymbol{\psi} \notin$ SAT. In this case, it holds that $\langle \phi, \psi\rangle \in$ SAT-UNSAT. By Lemma 5.17 (part 3), there is an L-hyperpath $\Pi_{u(\phi)w(\phi)}$ from $u(\phi)$ to $w(\phi)$ in $\mathcal{H}(\phi)$. Also by Lemma 5.17 (part 2), there is an L-hyperpath $\Pi_{u(\psi),v(\psi)}$ from $u(\psi)$ to $v(\psi)$ in $\mathcal{H}(\psi)$. It follows that there is an L-hypercycle Π in $\mathcal{G}$ that starts from the vertex $u(\phi)$ and proceeds with the following sequence of steps in order:

- take the route of $\Pi_{u(\phi)w(\phi)}$ starting from $u(\phi)$,
- make a transition to the vertex $u(\psi)$ from the end vertex $w(\phi)$ of $\Pi_{u(\phi)w(\phi)}$ using the interconnection hyperedge e_1,
- take the route of $\Pi_{u(\psi)v(\psi)}$ starting from the vertex $u(\psi)$,
- finally make a transition to the vertex $u(\phi)$ from the end vertex $v(\psi)$ of $\Pi_{u(\psi)v(\psi)}$ using the interconnection hyperedge e_3.

Thus, we have $\langle \mathcal{G}, \emptyset\rangle \notin$ CYCLOMATIC-SET. We claim that the directed hypergraph $(V(\mathcal{G}), E(\mathcal{G}) - B)$ has no L-hypercycle. Assuming this claim, it would follow that $\langle \mathcal{G}, B\rangle \in$ CYCLOMATIC-SET, and so we would have $\langle \mathcal{G}, B\rangle \in$ MINIMAL-CYCLOMATIC-SET.

Claim 5.19. $(V(\mathcal{G}), E(\mathcal{G}) - B)$ *has no L-hypercycle.*

Proof of Claim 5.19. Assume to the contrary that $(V(\mathcal{G}), E(\mathcal{G}) - B)$ has some L-hypercycle, say Π. Then it follows by Lemma 5.17 (part 1) that Π must include vertices and hyperedges of both $\mathcal{H}(\phi)$ and $\mathcal{H}(\psi)$. It means that Π must include a hyperedge that leaves $\mathcal{H}(\phi)$ (and enters $\mathcal{H}(\psi)$) and also a hyperedge that leaves $\mathcal{H}(\psi)$ (and enters $\mathcal{H}(\phi)$). Since e_1 is the only hyperedge that leaves $\mathcal{H}(\phi)$, e_1 is present in Π. Similarly, it follows that e_2 is present in Π. The head of e_1 is $\{u(\psi)\}$ and the tail of e_2 is $\{w(\psi)\}$; so, $u(\psi)$ and $w(\psi)$ are also present in Π.

Next we note that $u(\phi)$ is the pivot of $\mathcal{G}$. Therefore, Π must start from $u(\phi)$, leave $\mathcal{H}(\phi)$ using e_1, traverse $u(\psi)$, and finally return back to $u(\phi)$ using e_2 after traversing $w(\psi)$. It is not hard to see that the sequence of vertices and hyperedges from $u(\psi)$ to $w(\psi)$ in Π must correspond to an L-hyperpath from $u(\psi)$ to $w(\psi)$ in $\mathcal{H}(\psi)$. But this implies from Lemma 5.17 (part 3) that $\psi \in$ SAT, a contradiction. This completes the proof of Claim 5.19. ■ (Claim 5.19)

Case 2: $\boldsymbol{\phi} \in$ SAT and $\boldsymbol{\psi} \in$ SAT. In this case, it holds that $\langle \phi, \psi\rangle \notin$ SAT-UNSAT. By Lemma 5.17 (part 3), there is an L-hyperpath $\Pi_{u(\phi)w(\phi)}$ from $u(\phi)$ to $w(\phi)$ in $\mathcal{H}(\phi)$ and an L-hyperpath $\Pi_{u(\psi)w(\psi)}$ from $u(\psi)$ to $w(\psi)$ in $\mathcal{H}(\psi)$. It follows that there is an L-hypercycle Π in $\mathcal{G}$ that starts from the vertex $u(\phi)$ and proceeds with the following sequence of steps in order:

- take the route of $\Pi_{u(\phi)w(\phi)}$ starting from $u(\phi)$,
- make a transition to the vertex $u(\psi)$ from the end vertex $w(\phi)$ of $\Pi_{u(\phi)w(\phi)}$ using the interconnection hyperedge e_1,
- take the route of $\Pi_{u(\psi)w(\psi)}$ starting from the vertex $u(\psi)$,
- finally make a transition to the vertex $u(\phi)$ from the end vertex $w(\psi)$ of $\Pi_{u(\psi)w(\psi)}$ using the interconnection hyperedge e_2.

Notice that Π does not include the hyperedge e_3. Therefore, the directed hypergraph $(V(\mathcal{G}), E(\mathcal{G}) - B)$ contains the L-hypercycle Π, and so $\langle \mathcal{G}, B\rangle \notin$ CYCLOMATIC-SET. Hence, it also follows that $\langle \mathcal{G}, B\rangle \notin$ MINIMAL-CYCLOMATIC-SET.

Case 3: $\boldsymbol{\phi} \notin$ SAT. In this case, it holds that $\langle \phi, \psi\rangle \notin$ SAT-UNSAT. By Lemma 5.17 (part 3), there is no L-hyperpath from $u(\phi)$ to $w(\phi)$ in $\mathcal{H}(\phi)$. We make the following claim:

Claim 5.20. *$\mathcal{G}$ has no L-hypercycle.*

The proof of Claim 5.20 is similar to that of Claim 5.19 and so it is omitted. From this claim, it follows that both $\langle \mathcal{G}, B\rangle$ and $\langle \mathcal{G}, \emptyset\rangle$ are in CYCLOMATIC-SET, and so we also have $\langle \mathcal{G}, B\rangle \notin$ MINIMAL-CYCLOMATIC-SET.

Thus, we have shown that in all cases $\langle \phi, \psi\rangle \in$ SAT-UNSAT if and only if $\langle \mathcal{G}, B\rangle \in$ MINIMAL-CYCLOMATIC-SET. This completes the proof of Lemma 5.18. ■ (Lemma 5.18)

In the above reduction, notice that the construction of $\mathcal{G}$ involves only B-hyperedges, that all B-hyperedges of $\mathcal{G}$ have tails of size at most 2, and that there are B-hyperedges of $\mathcal{G}$ whose tails have size exactly 2. Hence, it follows that for every $k \geq 2$, MINIMAL-CYCLOMATIC-SET is DP-complete even when restricted to k-directed B-hypergraphs. This completes the proof of Theorem 5.16. ■ (Theorem 5.16)

5.8. *Proof of Lemma 5.12*

Proof. Fix an arbitrary truth-value assignment α of X. We will prove that $\phi(\alpha, Y) \in$ SAT if and only if there is an L-hypercycle Π in remove$(\mathcal{H}, \alpha)$.

A process of testing the satisfiability of a boolean CNF formula could be as follows: (1) choosing a truth-value assignment to the variables of the CNF formula, (2) expressing each clause as a disjunction of truth-values (T or F) by evaluating the literals of the clause under the truth-value assignment, and (3) looking for an occurrence of T in each clause. If at the end of step (3), all the clauses have an occurrence of T, then the CNF formula is satisfied; otherwise, it is not satisfied by the chosen truth-value assignment. We will give an analogy between these steps and the steps involved in checking the existence of an L-hypercycle Π in remove$(\mathcal{H}, \alpha)$.

Notice that any L-hypercycle Π in remove$(\mathcal{H}, \alpha)$ must start from $v_0 = \text{vertex}_{\mathcal{H}}(A, L_1)$. (This follows because the only hyperedges e of remove$(\mathcal{H}, \alpha)$ for which $|T(e)| = 1$ are (i) the hyperedges whose tail set $T(e)$ is $\{v_0\}$ and (ii) the distinguished hyperedges of the selectors. However, hyperedges of type (ii) are incapable of forming any L-hypercycle. So, Π must start from a hyperedge of type (i). In other words, Π must start from v_0.) It follows by the construction of $\mathcal{H}$ described in Section 5.3 that Π must consist of an L-hyperpath Π_1 from v_0 to $\text{vertex}_{\mathcal{H}}(J, L_m)$ that traverses every selector L_i.

Recall from Definition 5.11 that for each variable x_i of X, the assignment $\alpha(x_i) = \text{F}$ corresponds to the removal of the hyperedge $\text{edge}_{\mathcal{H}}(1, L_i)$ from $\mathcal{H}$ and the assignment $\alpha(x_i) = \text{T}$ corresponds to the removal of the hyperedge $\text{edge}_{\mathcal{H}}(2, L_i)$ from $\mathcal{H}$. Therefore, it is not hard to see that the L-hyperpath Π_1 in remove$(\mathcal{H}, \alpha)$ satisfies the following property by virtue of the design of the selectors: For every $1 \leq i \leq m$, Π_1 traverses $\text{vertex}_{\mathcal{H}}(F, L_i)$ if and only if $\alpha(x_i) = \text{F}$, and Π_1 traverses $\text{vertex}_{\mathcal{H}}(G, L_i)$ if and only if $\alpha(x_i) = \text{T}$. Thus, we have shown an association between the truth-value assignments of the variables of X and the vertices of the selectors traversed by any L-hyperpath Π_1 from v_0 to $\text{vertex}_{\mathcal{H}}(J, L_m)$ in remove$(\mathcal{H}, \alpha)$.

On reaching $\text{vertex}_{\mathcal{H}}(J, L_m)$, the end of Π_1, Π needs to traverse the divider D_1. In this traversal, Π may also traverse certain switches by taking the hyperedges connecting D_1 to these switches. Observe that if Π enters a switch S from any divider D_k (by taking a simple directed edge whose head is $\text{vertex}_{\mathcal{H}}(C', S)$ and whose tail contains some vertex of D_k), then Π must leave S after traversing the vertices and hyperedges of S in the order shown in Fig. 10. In particular, Π must traverse $\text{vertex}_{\mathcal{H}}(F', S)$ before it leaves S and returns to D_k.

The uniqueness of this traversal order is due to the following reasoning: Assume to the contrary that Π does not use this traversal order. Since Π starts from and ends at v_0 and since there is only one hyperedge, namely $(\{\text{vertex}_{\mathcal{H}}(B, S_{\text{last}}), v_0\}, \{v_0\})$, that enters v_0, this hyperedge must be the last one in Π. Therefore, on entering S from D_k, Π must leave S. The only possible traversal order is the one shown in Fig. 11. This order requires that Π must traverse $\text{vertex}_{\mathcal{H}}(A, S)$ before $\text{vertex}_{\mathcal{H}}(C', S)$ because of the B-hyperedge $\text{edge}_{\mathcal{H}}(4', S)$. However if Π traverses $\text{vertex}_{\mathcal{H}}(A, S)$ before $\text{vertex}_{\mathcal{H}}(C', S)$, then Π must also traverse $\text{vertex}_{\mathcal{H}}(B, S)$ before $\text{vertex}_{\mathcal{H}}(C', S)$. This leads to a contradiction since $\text{vertex}_{\mathcal{H}}(B, S)$ appears later than $\text{vertex}_{\mathcal{H}}(C', S)$ in the traversal order.

Hence, it follows that on reaching $\text{vertex}_{\mathcal{H}}(J, L_m)$, Π must traverse each of the dividers $D_1, D_2, \ldots, D_n$ in that order. For every $1 \leq k \leq n$, Π can traverse D_k by taking either the left wing or the right wing, where the left wing is a simple path from $\text{vertex}_{\mathcal{H}}(A, D_k)$ to $\text{vertex}_{\mathcal{H}}(B, D_k)$ through $\text{vertex}_{\mathcal{H}}(F, D_k)$ and the right wing is a simple path from $\text{vertex}_{\mathcal{H}}(A, D_k)$ to $\text{vertex}_{\mathcal{H}}(B, D_k)$ through $\text{vertex}_{\mathcal{H}}(F', D_k)$. There are exactly 2^n different ways to choose wings for traversing all the dividers and they correspond to the 2^n different truth-value assignments to the variables of Y. This leads to a natural way of associating truth-value assignments to the variables of Y with the choice of wings for traversal of all the dividers: For every

$1 \le k \le n$, if Π traverses D_k by taking its left wing, then we assume that y_k is set to F; otherwise, if Π traverses D_k by taking its right wing, then we assume that y_k is set to T. Thus, we have shown an association between the truth-value assignments of the variables of Y and the choice of wings taken by Π in its traversal of all the dividers.

The left wing or the right wing of any divider D_k may traverse the vertices of certain switches because of connections between the dividers and switches. Consider the case that the left wing of D_k traverses the vertices of some switch $S_{i,j}$. By the construction of $\mathcal{H}$ described in Section 5.3, we know that the left wing of D_k is connected to $S_{i,j}$ if and only if the ith clause $\phi_i(X, Y)$ contains y_k as the jth Y-literal. Therefore, in this case, y_k appears as a literal in ϕ_i. Notice that the left wing of D_k, which corresponds to setting y_k to F, must enter $S_{i,j}$ at $\text{vertex}_{\mathcal{H}}(C', S_{i,j})$ and, as explained earlier, must traverse $\text{vertex}_{\mathcal{H}}(F', S_{i,j})$ before it leaves $S_{i,j}$ and returns to D_k. On the other hand, the right wing of D_k, which corresponds to setting y_k to T, does not traverse any vertex of $S_{i,j}$. Similarly, in the other case in which the right wing of D_k traverses the vertices of some switch $S_{i,j}$ and in which $\bar{y}_k$ appears as the jth Y-literal in $\phi_i(X, Y)$, we can show the following: The left wing of D_k, which corresponds to setting y_k to F, does not traverse any vertex of $S_{i,j}$ whereas the right wing of D_k, which corresponds to setting y_k to T, must traverse $\text{vertex}_{\mathcal{H}}(F', S_{i,j})$. In other words, we have shown an association between the evaluation of the Y-literals occurring in the clauses of $\phi(\alpha, Y)$ and the choice of whether to traverse the vertices of the switches taken by Π in its traversal of all the dividers. Proposition 5.21 summarizes this association.

Proposition 5.21. *For every $1 \le i \le s$, every $1 \le j \le p_i$, and for any truth-value assignment β to the variables Y of $\phi(\alpha, Y)$, the wing of $D_{v(i,j)}$, associated with the truth-value of $y_{v(i,j)}$ under β, traverses the vertices of $S_{i,j}$ (in particular,* $\text{vertex}_{\mathcal{H}}(F', S_{i,j})$*) if and only if the jth Y-literal of $\phi_i(\alpha, Y)$ evaluates to* F *under β.*

After traversing all the dividers, Π needs to traverse the chooser C_1. In this traversal, Π either takes a bypass hyperedge of C_1 or traverses the vertices of some switch. Observe that if Π enters a switch S from any chooser C_i (by taking a simple directed edge whose head is $\text{vertex}_{\mathcal{H}}(C, S)$ and whose tail contains some vertex of C_i), then Π must leave S after traversing the vertices and hyperedges of S in the order shown in Fig. 12. In particular, Π must traverse $\text{vertex}_{\mathcal{H}}(F, S)$ before it leaves S and returns to C_i. The reasoning behind the uniqueness of the traversal order is similar to the one given earlier.

Hence, it follows that on reaching $\text{vertex}_{\mathcal{H}}(B, D_n)$, Π must traverse each of the choosers $C_1, C_2, \ldots, C_s$ in that order. For every $1 \le i \le s$, we know from the previous paragraph that if Π enters a switch S from C_i, then it must traverse $\text{vertex}_{\mathcal{H}}(F, S)$ before it leaves S and returns to C_i. On the other hand, if Π takes a bypass hyperedge on reaching $\text{vertex}_{\mathcal{H}}(A, C_i)$, then it does not enter any switch from C_i. We claim that Π can take a bypass hyperedge of C_i if and only if $\phi_i(\alpha, Y)$ is equivalent to T (i.e., $\phi_i(\alpha, Y)$ evaluates to T for all possible truth-value assignments to the variables of Y).

Claim 5.22. *Π can take a bypass hyperedge of C_i if and only if $\phi_i(\alpha, Y) \equiv$* T.

Proof of Claim 5.22. We give the proof of the right-to-left implication only; the proof of the left-to-right implication is similar. Assume that $\phi_i(\alpha, Y) \equiv$ T. Then one of the two cases (a) and (b) holds: (a) there is a literal x_j in $\phi_i(X, Y)$ and $\alpha(x_j) =$ T and (b) there is a literal $\bar{x}_j$ in $\phi_i(X, Y)$ and $\alpha(x_j) =$ F. Let us assume that case (a) holds. (The proof for case (b) is similar.) Since there is a literal x_j in $\phi_i(X, Y)$, $\mathcal{H}$ as well as $\text{remove}(\mathcal{H}, \alpha)$ include a bypass hyperedge e whose tail set contains $\text{vertex}_{\mathcal{H}}(A, C_i)$ and whose head set is $\{\text{vertex}_{\mathcal{H}}(B, C_i)\}$. Notice that the tail set of e contains also $\text{vertex}_{\mathcal{H}}(G, L_j)$. Since $\alpha(x_j) =$ T, it follows (from the association shown earlier between the truth-value assignments of the variables of X and the vertices of the selectors traversed by Π) that Π must have already traversed $\text{vertex}_{\mathcal{H}}(G, L_j)$. Thus, we infer that Π can take the bypass hyperedge e on reaching $\text{vertex}_{\mathcal{H}}(A, C_i)$. Hence, Claim 5.22 is proved. ■ (Claim 5.22)

The following proposition summarizes the above discussion.

Proposition 5.23. *For every $1 \le i \le s$, the following hold during the traversal of the chooser C_i:*

2. *If $\phi_i(\alpha, Y) \equiv$ T, then Π can take a bypass hyperedge of C_i and so can avoid entering any switch from C_i.*
2. *If $\phi_i(\alpha, Y) \not\equiv$ T, then for some $1 \le j \le p_i$, Π must enter $S_{i,j}$ from C_i and so must traverse* $\text{vertex}_{\mathcal{H}}(F, S_{i,j})$.

After traversing all the choosers, Π needs to traverse the switch S_{first}. Observe that if Π enters a switch S at $\text{vertex}_{\mathcal{H}}(A, S)$, then it cannot traverse the chooser and the divider connected to S since otherwise Π would not be an L-hypercycle. Therefore, on reaching $\text{vertex}_{\mathcal{H}}(A, S)$, Π must take either the simple path from $\text{vertex}_{\mathcal{H}}(A, S)$ to $\text{vertex}_{\mathcal{H}}(B, S)$ through $\text{vertex}_{\mathcal{H}}(F, S)$ or the simple path from $\text{vertex}_{\mathcal{H}}(A, S)$ to $\text{vertex}_{\mathcal{H}}(B, S)$ through $\text{vertex}_{\mathcal{H}}(F', S)$. Hence, it follows that on reaching $\text{vertex}_{\mathcal{H}}(B, C_s)$, Π must traverse each of the switches $S_{i,j}$s, for $1 \le i \le s$ and $1 \le j \le p_i$, in the lexicographic order of their labels (i, j)s. We now discuss how Π can traverse each of these switches.

Consider the case $\phi_i(\alpha, Y) \equiv$ T. From Proposition 5.23, we can assume that Π would take the bypass hyperedge of the chooser C_i during its traversal of C_i. Thus, Π would avoid traversing $\text{vertex}_{\mathcal{H}}(F, S_{i,j})$, for every $1 \le j \le p_i$, during its traversal of C_i. It follows that for each $1 \le j \le p_i$, Π can traverse $S_{i,j}$ by taking the simple path from $\text{vertex}_{\mathcal{H}}(A, S_{i,j})$ to $\text{vertex}_{\mathcal{H}}(B, S_{i,j})$ through $\text{vertex}_{\mathcal{H}}(F, S_{i,j})$.

Next, consider the case $\phi_i(\alpha, Y) \not\equiv$ T. From Proposition 5.23, we know that Π must traverse $\text{vertex}_{\mathcal{H}}(F, S_{i,j})$, for some $1 \le j \le p_i$, during its traversal of C_i. Thus, for all $k \ne j$ and $1 \le k \le p_i$, it is easy to see that Π on reaching $\text{vertex}_{\mathcal{H}}(A, S_{i,k})$ can always traverse $S_{i,k}$ by taking the simple path from $\text{vertex}_{\mathcal{H}}(A, S_{i,k})$ to $\text{vertex}_{\mathcal{H}}(B, S_{i,k})$ through $\text{vertex}_{\mathcal{H}}(F, S_{i,k})$. The interesting issue here is whether Π can also traverse $S_{i,j}$ on reaching $\text{vertex}_{\mathcal{H}}(A, S_{i,j})$. We analyze this issue next.

Let β be any truth-value assignment of Y such that the jth Y-literal of $\phi_i(\alpha, Y)$ evaluates to T under β. Then Proposition 5.21 implies that the wing of $D_{v(i,j)}$, associated with the truth-value of $y_{v(i,j)}$ under β, avoids traversing the vertices of $S_{i,j}$ (in particular, $\text{vertex}_{\mathcal{H}}(F', S_{i,j})$). If Π takes that wing in its traversal of $D_{v(i,j)}$, then on entering $S_{i,j}$ from $\text{vertex}_{\mathcal{H}}(A, S_{i,j})$, it can take the simple path from $\text{vertex}_{\mathcal{H}}(A, S_{i,j})$ to $\text{vertex}_{\mathcal{H}}(B, S_{i,j})$ through $\text{vertex}_{\mathcal{H}}(F', S_{i,j})$.

Let us now assume that β is a truth-value assignment of Y such that every Y-literal of $\phi_i(\alpha, Y)$ evaluates to F under β, i.e., $\phi_i(\alpha, \beta) = \text{F}$. Under this assumption, Proposition 5.21 implies that for every $1 \le j \le p_i$, the wing of $D_{v(i,j)}$, associated with the truth-value of $y_{v(i,j)}$ under β, traverses $\text{vertex}_{\mathcal{H}}(F', S_{i,j})$. Recall (from the discussion of the case $\phi_i(\alpha, Y) \not\equiv \text{T}$) that Π must traverse $\text{vertex}_{\mathcal{H}}(F, S_{i,j})$ during its traversal of C_i. Thus, if Π takes this wing in its traversal of $D_{v(i,j)}$, then it would have no way to take a simple path from $\text{vertex}_{\mathcal{H}}(A, S_{i,j})$ to $\text{vertex}_{\mathcal{H}}(B, S_{i,j})$ on entering $S_{i,j}$ from $\text{vertex}_{\mathcal{H}}(A, S_{i,j})$. In other words, Π would not be able to reach $\text{vertex}_{\mathcal{H}}(B, S_{i,j})$, and hence complete the L-hypercycle.

The following proposition summarizes the above discussion.

Proposition 5.24. *For every $1 \le i \le s$ and for any truth-value assignment β of Y, the following holds during the traversal of the switches $S_{i,j}$s from $\text{vertex}_{\mathcal{H}}(A, S_{i,j})$, where $1 \le j \le p_i$: Π can traverse all these switches $S_{i,j}$s if and only if $\phi_i(\alpha, \beta) = \text{T}$.*

We claim that $\phi(\alpha, Y) \in \text{SAT}$ if and only if there is an L-hypercycle Π in $\text{remove}(\mathcal{H}, \alpha)$. Suppose that $\phi(\alpha, Y) \in \text{SAT}$. Then there is a truth-value assignment β of Y such that $\phi(\alpha, \beta) = \text{T}$. The choice of α and β defines a unique L-hyperpath from v_0 to $\text{vertex}_{\mathcal{H}}(B, D_n)$. This follows because of the association shown earlier between the truth-value assignments of the variables (of X and Y) and the choice of simple paths taken by any L-hyperpath Π_1 from v_0 to $\text{vertex}_{\mathcal{H}}(B, D_n)$ in $\text{remove}(\mathcal{H}, \alpha)$. We assume that Π would take that L-hyperpath in its traversal of all the selectors and dividers. On reaching $\text{vertex}_{\mathcal{H}}(B, D_n)$, Π needs to traverse all the choosers. We define Π so that for every $1 \le i \le s$, we have

Case 1: $\boldsymbol{\phi_i(\alpha, Y) \equiv \text{T}}$**.** In this case, Π takes a bypass hyperedge of C_i.
Case 1: $\boldsymbol{\phi_i(\alpha, Y) \not\equiv \text{T}}$**.** In this case, Π enters $S_{i,j}$ from C_i, where $j \in \{1, 2, \ldots, p_i\}$ is such that the jth Y-literal of $\phi_i(\alpha, Y)$ evaluates to T under β.

Proposition 5.23 and the fact that $\phi(\alpha, \beta) = \text{T}$ imply that Π is a well-defined L-hyperpath in both the cases. Thus, Π only needs to traverse all the switches in order to complete the L-hypercycle in $\text{remove}(\mathcal{H}, \alpha)$. We observe that Proposition 5.24 implies that Π can traverse all the switches as well since $\phi(\alpha, \beta) = \text{T}$. Hence, it follows that Π is a legal L-hypercycle in $\text{remove}(\mathcal{H}, \alpha)$.

Next suppose that $\phi(\alpha, Y) \notin \text{SAT}$. As explained earlier, any L-hypercycle Π in $\text{remove}(\mathcal{H}, \alpha)$ must start from v_0 and would need to traverse all the selectors, dividers, choosers, and switches in that order. So assume that Π has already traversed all the selectors, dividers, and choosers. We know that the selection of the wings of the dividers taken by Π corresponds to a fixed truth-value assignment β of Y. Since $\phi(\alpha, Y)$ is unsatisfiable, there is some clause $\phi_i(\alpha, Y)$ of ϕ such that $\phi_i(\alpha, Y)$ evaluates to F under β, i.e., $\phi_i(\alpha, \beta) = \text{F}$. Proposition 5.24 now implies that Π cannot traverse all the switches $S_{i,j}$s, where $1 \le j \le p_i$. In other words, there cannot be any L-hypercycle in $\text{remove}(\mathcal{H}, \alpha)$.

This completes the proof of Lemma 5.12. ■ (Lemma 5.12)

5.9. *Proof of Lemma 5.17*

Proof. Let $\phi(Y)$ be any boolean CNF formula, which is input to f. We construct a directed hypergraph $\mathcal{H}$ from $\phi(Y)$ using the construction described in Section 5.3 with one change: There are no selector gadgets in $\mathcal{H}$ since ϕ does not contain any variable from the set X. Thus, $\mathcal{H}$ consists of a sequence of dividers, followed by a sequence of choosers, and finally a sequence of switches with hyperedges connecting these gadgets as in the construction of Section 5.3. However, $\mathcal{H}$ contains no hyperedges incident on the selectors (e.g., the bypass hyperedges).

It can be proven along the lines of the proof of Lemma 5.12 that $\mathcal{H}$ has no L-hypercycle and that $\phi(Y) \in \text{SAT}$ if and only if there is an L-hyperpath from $\text{vertex}_{\mathcal{H}}(A, D_1)$ to $\text{vertex}_{\mathcal{H}}(B, S_{\text{last}})$. Furthermore, the same proof also shows that there is always an L-hyperpath from $\text{vertex}_{\mathcal{H}}(A, D_1)$ to $\text{vertex}_{\mathcal{H}}(B, C_s)$.

It follows that setting u to $\text{vertex}_{\mathcal{H}}(A, D_1)$, v to $\text{vertex}_{\mathcal{H}}(B, C_s)$, and w to $\text{vertex}_{\mathcal{H}}(B, S_{\text{last}})$ satisfies the statement of the lemma. (Fig. 8 is a schematic diagram that shows the construction of $\mathcal{H}$ and the vertices u, v, and w.)

This completes the proof of Lemma 5.17. ■ (Lemma 5.17)

6. Succinct representations of directed hypergraphs

A directed hypergraph on n vertices may have as many as $\Theta(3^n)$ hyperedges. In contrast, the number of edges in a directed graph is $O(n^2)$. From an implementation perspective, any representation that stores information for individual hyperedges of a directed hypergraph is impractical for directed hypergraphs with a large number of hyperedges. Thus, alternative ways to represent directed hypergraphs must be explored. Several graphs occurring in practice, such as the graphs that model VLSI circuits, have a highly organized structure and can be described in a succinct way by a circuit or a boolean formula. Galperin and Wigderson [13] showed that trivial graph properties, e.g., the existence of a triangle, become NP-complete, and Papadimitriou and Yannakakis [28] showed that graph properties that are ordinarily NP-complete become NEXP-complete when the graph is succinctly described by a circuit. In this section, we investigate the computational complexity of the L-hyperpath existence problem when directed hypergraphs are represented in an exponentially succinct way.

Definition 6.1. We define the following succinct representations:

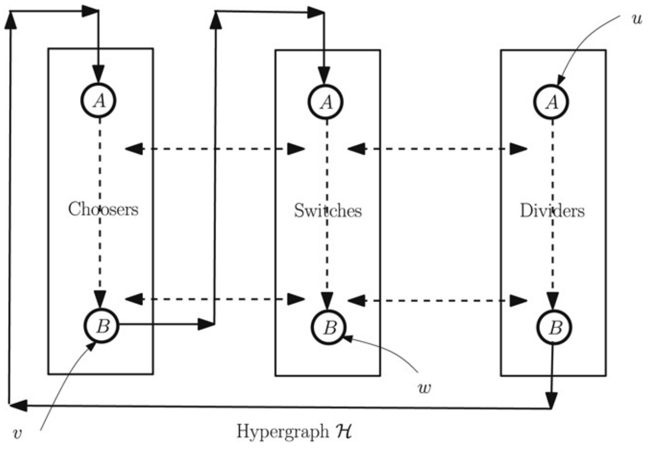


Fig. 8. The figure shows the directed hypergraph $\mathcal{H}$ and the vertices u, v, and w as defined in the proof of Lemma 5.17. Dashed double-arrowed lines show the connections between choosers and switches, and the connections between switches and dividers. Dashed single-arrowed lines show the connections between the choosers, between the dividers, and between the switches.

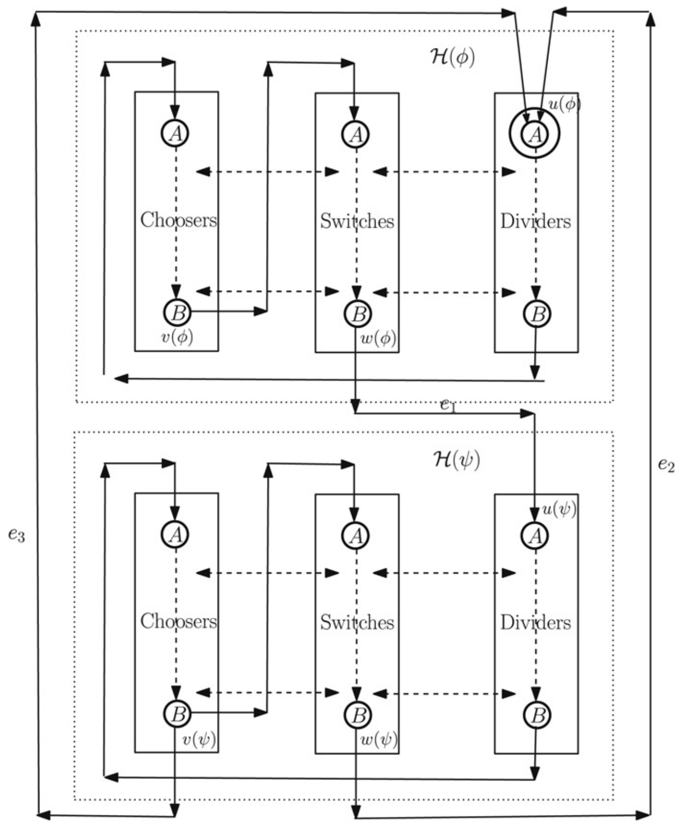


Fig. 9. The directed hypergraph $\mathcal{G}$ obtained from $\langle \phi, \psi \rangle$ using the reduction given in the proof of Theorem 5.16 is shown. The vertex vertex$_{\mathcal{H}(\phi)}(A, D_1)$, highlighted by concentric circles, is the pivot of $\mathcal{G}$. Though the figure shows hyperedges e for which $|T(e)| = 1$ and the pivot is not in $T(e)$, these hyperedges are implicitly assumed to also have the pivot in their tail set.

1. A *succinct representation* of a directed hypergraph $\mathcal{H}(V, E)$, where $V = \{1, 2, \ldots, n\}$, is a boolean circuit $C_{\mathcal{H}}$ with $2n$ input gates and an output gate such that, for every $e \subseteq 2^V \times 2^V$, $e \in E$ if and only if $C_{\mathcal{H}}(x, y)$ outputs 1, where $x = \chi_{T(e)}(1)\chi_{T(e)}(2)\ldots\chi_{T(e)}(n)$ and $y = \chi_{H(e)}(1)\chi_{H(e)}(2)\ldots\chi_{H(e)}(n)$. (Here, for a set S and an element e from the universe, $\chi_S(e) = 1$ if $e \in S$, and $\chi_S(e) = 0$ if $e \notin S$.)
2. For every $k \geq 1$, a *succinct representation* of a k-directed hypergraph $\mathcal{H}(V, E)$, where $V = \{1, \ldots, n\}$, is a boolean circuit $C_{\mathcal{H}}$ with $2k\lceil\log(n + 1)\rceil$ input gates and an output gate, where $0^{\lceil\log(n+1)\rceil}$ is the encoding of a dummy vertex not in $\mathcal{H}$ and for each $1 \leq i \leq n$, $bin(i)$ – the binary representation of integer i in $\lceil\log(n + 1)\rceil$ bits – is the encoding of vertex i in $\mathcal{H}$. Furthermore, for each $e = (\{i_1, \ldots, i_{\ell_1}\}, \{j_1, \ldots, j_{\ell_2}\}) \subseteq 2^V \times 2^V$ such that $i_1 < \cdots < i_{\ell_1}, j_1 < \cdots < j_{\ell_2}$, and

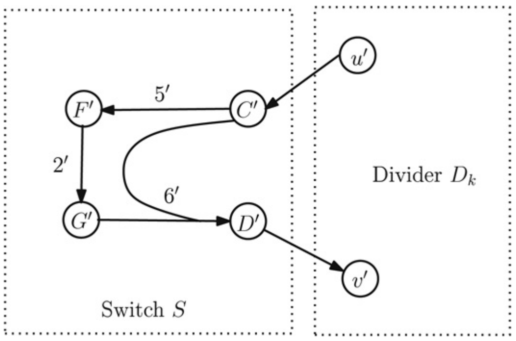


Fig. 10. The sequence of vertices of S traversed by Π on reaching the vertex C' is C', F', G', D'.

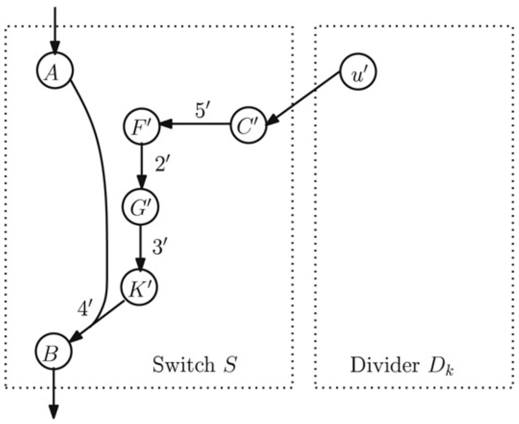


Fig. 11. If an L-hypercycle Π enters a switch S from a divider D_k, then it can take the simple path $(C', 5', F', 2', G', 3', K', 4', B)$ only if it has already traversed the vertex A.

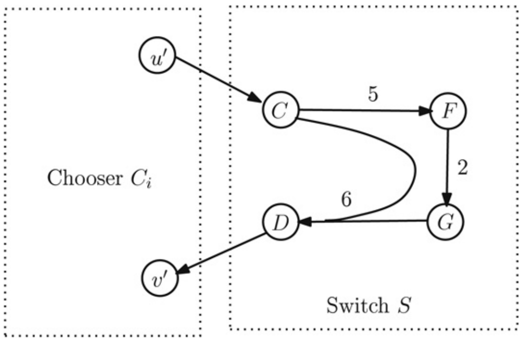


Fig. 12. The sequence of vertices of S traversed by Π on reaching the vertex C is C, F, G, D.

$1 \leq \ell_1, \ell_2 \leq k$, it holds that $e \in E$ if and only if $C_{\mathcal{H}}(x, y)$ outputs 1, where $x = 0^{(k-\ell_1)\lceil\log(n+1)\rceil} bin(i_{\ell_1}) \ldots bin(i_2)bin(i_1)$ and $y = 0^{(k-\ell_2)\lceil\log(n+1)\rceil} bin(j_{\ell_2}) \ldots bin(j_2)bin(j_1)$.

Since a directed graph is a 1-directed hypergraph, its succinct representation is given by Definition 6.1 (part 2) for $k = 1$.

Definition 6.2. We define the following computational problems:

1. SUCCINCT-LHYPERPATH $= \{\langle C, u, v\rangle \mid C$ is a circuit succinctly representing a directed hypergraph $\mathcal{H}_C$ and $\langle \mathcal{H}_C, u, v\rangle \in$ L-HYPERPATH$\}$.
2. k-SUCCINCT-LHYPERPATH $= \{\langle C, u, v\rangle \mid C$ is a circuit succinctly representing a k-directed hypergraph $\mathcal{H}_C$ and $\langle \mathcal{H}_C, u, v\rangle \in$ L-HYPERPATH$\}$.

Wagner [37,38] (see also [28]) showed that even for simple subclasses of graphs – directed trees, directed acyclic graphs, directed forests, and undirected forests – the reachability problem for each class with succinct input representation is PSPACE-complete. Tantau [34] showed that the reachability problem for succinctly represented tournaments (in fact, for strongly connected tournaments) is Π_2^p-complete. Using the proof of Theorem 4.4, it can be easily shown that SUCCINCT-LHYPERPATH is NP-complete and for every $k \geq 2$, k-SUCCINCT-LHYPERPATH is NEXP-complete.

Theorem 6.3. SUCCINCT-LHYPERPATH *is* NP-*complete.*

Proof. We first show that there is an NP machine that accepts SUCCINCT-LHYPERPATH. On input $\langle C, u, v\rangle$, M does the following:

1. M first guesses ℓ, an integer in the range $0 \dots n-2$, and then guesses ℓ vertices $w_1, w_2, \dots, w_\ell$ and $\ell+1$ hyperedges $e_1 = (x_1, y_1), e_2 = (x_2, y_2), \dots, e_{\ell+1} = (x_{\ell+1}, y_{\ell+1})$. Here n is the number of vertices in the directed hypergraph represented by C and for every $1 \le i \le \ell+1, x_i = \chi_{T(e_i)}(1)\chi_{T(e_i)}(2)\dots\chi_{T(e_i)}(n)$ and $y_i = \chi_{H(e_i)}(1)\chi_{H(e_i)}(2)\dots\chi_{H(e_i)}(n)$.
2. M then verifies the constraints, stated in Definition 3.1, for the existence of an L-hyperpath $\Pi_{uv} = (u, e_1, w_1, e_2, w_2, \dots, e_\ell, w_\ell, e_{\ell+1}, v)$ from u to v in $\mathcal{H}$. (Thus, for instance, to check whether $e_1, \dots, e_{\ell+1}$ are distinct hyperedges, M verifies in time polynomial in n whether the constraints $\bigwedge_{1\le i<j\le \ell+1}[(x_i, y_i) \ne (x_j, y_j)]$ and $\bigwedge_{1\le i\le \ell+1}[C(x_i, y_i) = 1]$ evaluate to true.) M accepts if and only if all the constraints are satisfied.

We now give a polynomial-time many-one reduction σ from HAMILTONIAN PATH to SUCCINCT-LHYPERPATH. Recall that the many-one reduction σ' from HAMILTONIAN PATH to L-HYPERPATH described in the proof of Theorem 4.4 maps $\langle G, w_1, w_2\rangle$ to $\langle \mathcal{H}, u, v\rangle$, where G is a directed graph, $w_1, w_2 \in V(G)$, and $\mathcal{H}$ is a directed hypergraph such that $\langle G, w_1, w_2\rangle \in$ HAMILTONIAN PATH if and only if $\langle \mathcal{H}, u, v\rangle \in$ L-HYPERPATH. Let $|V(\mathcal{H})| = n$. Then the size of $\mathcal{H}$ is polynomially bounded in n. On input $\langle G, w_1, w_2\rangle$, σ does the following: It computes $\sigma'(\langle G, w_1, w_2\rangle) = \langle \mathcal{H}, u, v\rangle$ and generates a circuit C that succinctly represents $\mathcal{H}$. The circuit C is specified as follows: $C = \bigvee_{e\in E(\mathcal{H})} \left(\bigwedge_{i=1}^{n} X_i(e) \wedge \bigwedge_{j=1}^{n} Y_j(e)\right)$, where

$$X_i(e) = \begin{cases} x_i & \text{if } i \in T(e),\\ \overline{x}_i & \text{if } i \notin T(e),\end{cases} \quad\text{and } Y_j(e) = \begin{cases} y_j & \text{if } j \in H(e),\\ \overline{y}_j & \text{if } j \notin H(e).\end{cases}$$

It is easy to see that $\langle G, w_1, w_2\rangle \in$ HAMILTONIAN PATH if and only if $\langle C, u, v\rangle \in$ SUCCINCT-LHYPERPATH. Hence Theorem 6.3 is proved. ■ (Theorem 6.3)

We next give a polynomial-time many-one reduction from SUCCINCT HAMILTONIAN PATH to k-SUCCINCT-LHYPERPATH to prove that the latter problem is NEXP-hard. The problem SUCCINCT HAMILTONIAN PATH is known to be complete for NEXP (see [25]) and is defined as follows: Given the succinct representation C of a directed graph G and given two vertices u and v of G, does G have a Hamiltonian path from u to v?

Theorem 6.4. *For every $k \ge 2$, k-*SUCCINCT-LHYPERPATH *is* NEXP-*complete.*

Proof. Let $k \ge 2$ be a fixed integer and let M' be a machine defined similarly as M is defined in the proof of Theorem 6.3. It is easy to see that M' is an NEXP machine that accepts k-SUCCINCT-LHYPERPATH.

We now give a polynomial-time many-one reduction σ from SUCCINCT HAMILTONIAN PATH, a polynomial-time many-one complete problem for NEXP, to k-SUCCINCT-LHYPERPATH. On input $\langle C, u, v\rangle$, where C is a circuit succinctly encoding a directed graph G and u, v are vertices of G, σ outputs $\langle C', x, y\rangle$, where C' is the succinct representation of a 2-directed B-hypergraph $\mathcal{H}$ such that $\langle G, u, v\rangle$ reduces to $\langle \mathcal{H}, x, y\rangle$ via the reduction from HAMILTONIAN PATH to L-HYPERPATH described in the proof of Theorem 4.4.

Given two bit-strings, with $k\lceil\log(n+1)\rceil$ bits in each, representing $U_1 \subseteq V(\mathcal{H})$ and $U_2 \subseteq V(\mathcal{H})$, C' can compute whether $(U_1, U_2) \in E(\mathcal{H})$ by checking that (U_1, U_2) is a B-hyperedge with $|U_1| \le 2$, determining the indices of the vertices connected by the hyperedge, and using the circuit C to evaluate the adjacency relation in G. Thus, a circuit C' encoding the directed hypergraph $\mathcal{H}$ can be constructed in polynomial time from C. It follows that k-SUCCINCT-LHYPERPATH is NEXP-complete. This completes the proof of Theorem 6.4. ■ (Theorem 6.4)

Thus, Theorems 6.3 and 6.4 show that a simple restriction on the nature of directed hypergraphs being considered leads to a big jump in the computational complexity of the succinct version of the L-HYPERPATH problem. It is interesting to note that while k-SUCCINCT-LHYPERPATH is PSPACE-complete for the case $k = 1$ [28,37,38], we have shown here that the same problem is NEXP-complete for the case $k \ge 2$.

7. Conclusion

In this paper, we introduced the notion of L-hyperconnection in directed hypergraphs and showed that this notion can be used to model problems in diverse domains. We proved that several problems related to L-hyperpaths and L-hypercycles are likely to be computationally hard, since these problems are complete for complexity classes such as NP, Σ_2^p, Π_2^p, and DP. We now mention some research directions.

Acharya [1] showed a connection between the cyclomatic number and the planarity of an undirected hypergraph. Does the L-cyclomatic number of directed hypergraphs have any connection with notions in the theory of directed hypergraphs?

We showed two problem domains where L-hyperpaths can be used to model computational problems. Are there other problem domains where L-hyperpaths can be used?

Acknowledgments

We thank Edith Hemaspaandra, Lane Hemaspaandra, Christopher Homan, Proshanto Mukherji, Srinivasan Parathasarathy, Len Schubert, and Holger Spakowski for discussions related to this work. We also thank the anonymous referees for their helpful comments.

References

[1] B. Acharya, On the cyclomatic number of a hypergraph, Discrete Mathematics 27 (1979) 111–116.
[2] G. Ausiello, A. D'Atri, D. Saccá, Graph algorithms for functional dependency manipulation, Journal of the ACM 30 (4) (1983) 752–766.
[3] G. Ausiello, A. D'Atri, D. Saccá, Minimal representation of directed hypergraphs, SIAM Journal on Computing 15 (2) (1986) 418–431.
[4] G. Ausiello, P. Franciosa, D. Frigioni, Partially dynamic maintenance of minimum weight hyperpaths, Journal of Discrete Algorithms 3 (1) (2005) 27–46.
[5] P. Alimonti, E. Feuerstein, U. Nanni, Linear time algorithms for liveness and boundedness in conflict-free Petri nets, in: Proceedings of the 1st Latin American Symposium on Theoretical Informatics, in: Lecture Notes in Computer Science, vol. 583, Springer-Verlag, 1992, pp. 1–14.
[6] G. Ausiello, R. Giaccio, On-line algorithms for satisfiability formulae with uncertainty, Theoretical Computer Science 171 (1–2) (1997) 3–24.
[7] G. Ausiello, R. Giaccio, G. Italiano, U. Nanni, Optimal traversal of directed hypergraphs, Manuscript, 1997.
[8] G. Ausiello, G. Italiano, U. Nanni, Hypergraph traversal revisited: Cost measures and dynamic algorithms, in: Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science, in: Lecture Notes in Computer Science, vol. 1450, Springer-Verlag, 1998, pp. 1–16.
[9] C. Berge, Graphs and Hypergraphs, North-Holland, 1973.
[10] G. Gallo, G. Longo, S. Pallottino, S. Nguyen, Directed hypergraphs and applications, Discrete Applied Mathematics 42 (2–3) (1993) 177–201.
[11] G. Gallo, G. Rago, A hypergraph approach to logical inference for datalog formulae, Technical Report 28/90, Dip. di Informatica, Univ. of Pisa, Italy, 1990.
[12] G. Gallo, M. Scutella, Directed hypergraphs as a modelling paradigm, Technical Report TR-99-02, Dipartimento di Informatica, 1999.
[13] H. Galperin, A. Wigderson, Succinct representations of graphs, Information and Control 56 (3) (1983) 183–198.
[14] G. Italiano, U. Nanni, Online maintenance of minimal directed hypergraphs, in: Proceedings of the 3rd Italian Conference on Theoretical Computer Science, World Scientific Co., 1989, pp. 335–349.
[15] N. Jones, Y. Lien, W. Laaser, New problems complete for nondeterministic log space, Mathematical Systems Theory 10 (1) (1976) 1–17.
[16] N. Jones, Space-bounded reducibility among combinatorial problems, Journal of Computer and System Sciences 11 (1) (1975) 68–85.
[17] R. Karp, Reducibilities among combinatorial problems, in: R. Miller, J. Thatcher (Eds.), Complexity of Computer Computations, 1972, pp. 85–103.
[18] D. Klein, C. Manning, Parsing and hypergraphs, in: Proceedings of the 7th International Workshop on Parsing Technologies, IWPT, 2001.
[19] D. Knuth, A generalization of Dijkstra's algorithm, Information Processing Letters 6 (1) (1977) 1–5.
[20] T. Lakshman, D. Stiliadis, High-speed policy-based packet forwarding using efficient multi-dimensional range matching, ACM Computer Communication Review 28 (4) (1998) 203–214.
[21] B. Lampson, V. Srinivasan, G. Varghese, IP lookups using multiway and multicolumn search, in: Proceedings of IEEE INFOCOM 1998, 17th Annual Joint Conference of the IEEE Computer and Communications Societies, 1998, pp. 1248–1256.
[22] N. Nilson, Principles of Artificial Intelligence, Springer Verlag, 1982.
[23] S. Nguyen, S. Pallottino, Hyperpaths and shortest hyperpaths, Combinatorial Optimization 1403 (1989) 258–271.
[24] L. Nielsen, D. Pretolani, A remark on the definition of a B-hyperpath, Technical Report, Department of Operations Research, University of Aarhus, 2001.
[25] C. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.
[26] C. Petri, Communication with automata, Technical Report Supplement 1 to Tech. Report RADC-TR-65-377, 1, Univ. of Bonn, 1962.
[27] C. Papadimitriou, M. Yannakakis, The complexity of facets (and some facets of complexity), Journal of Computer and System Sciences 28 (2) (1984) 244–259.
[28] C. Papadimitriou, M. Yannakakis, A note on succinct representations of graphs, Information and Control 71 (3) (1986) 181–185.
[29] G. Ramalingam, T. Reps, An incremental algorithm for a generalization of the Shortest Path problem, Journal of Algorithms 21 (2) (1996) 267–305.
[30] A. Shamir, How to share a secret, Communications of the ACM 22 (11) (1979) 612–613.
[31] G. Simmons, An introduction to shared secret and/or shared control schemes and their application, in: G. Simmons (Ed.), Contemporary Cryptology, The Science of Information Integrity, IEEE Press, 1992, pp. 441–497.
[32] V. Srinivasan, S. Suri, G. Varghese, Packet classification using tuple space search, in: Proceedings of ACM SIGCOMM 1999, 1999, pp. 135–146.
[33] L. Stockmeyer, The polynomial-time hierarchy, Theoretical Computer Science 3 (1) (1976) 1–22.
[34] T. Tantau, A note on the complexity of the reachability problem for tournaments, Technical Report 01-092, Electronic Colloquium on Computational Complexity, 2001. http://www.eccc.uni-trier.de/eccc/.
[35] O. Temkin, A. Zeigarnik, D. Bonchev, Chemical Reaction Networks: A Graph-Theoretical Approach, CRC Press, 1996.
[36] J. Ullman, Principles of Database Systems, Computer Science Press, 1982.
[37] K. Wagner, The complexity of problems concerning graphs with regularities, in: Proceedings of the 11th Symposium on Mathematical Foundations of Computer Science, in: Lecture Notes in Computer Science, vol. 176, Springer-Verlag, 1984, pp. 544–552.
[38] K. Wagner, The complexity of combinatorial problems with succinct input representations, Acta Informatica 23 (3) (1986) 325–356.
[39] C. Wrathall, Complete sets and the polynomial-time hierarchy, Theoretical Computer Science 3 (1977) 23–33.
[40] A. Zeigarnik, On hypercycles and hypercircuits in hypergraphs, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 51 (2000) 377–383.