Sequence Hypergraphs: Paths, Flows, and Cuts*

Kateřina Böhmová[1], Jérémie Chalopin[2], Matúš Mihalák[3], Guido Proietti[4], and Peter Widmayer[1]

[1] Department of Computer Science, ETH Zurich, Zurich, Switzerland
asitak.kat@gmail.com widmayer@inf.ethz.ch
[2] Aix-Marseille Université, CNRS, Université de Toulon, LIS, Marseille, France
jeremie.chalopin@lis-lab.fr
[3] Department of Data Science and Knowledge Engineering, Maastricht University, Maastricht, The Netherlands matus.mihalak@maastrichtuniversity.nl
[4] DISIM, Università degli Studi dell'Aquila, Italy; and IASI, CNR, Roma, Italy
guido.proietti@univaq.it

Abstract. We introduce *sequence hypergraphs* by extending the concept of a directed edge (from simple directed graphs) to hypergraphs. Specifically, every hyperedge of a sequence hypergraph is defined as a sequence of vertices (not unlike a directed path). Sequence hypergraphs are motivated by problems in public transportation networks, as they conveniently represent transportation lines. We study the complexity of several fundamental algorithmic problems, arising (not only) in transportation, in the setting of sequence hypergraphs. In particular, we consider the problem of finding a *shortest st-hyperpath*: a minimum set of hyperedges that "connects" (allows to travel to) t from s; finding a *minimum st-hypercut*: a minimum set of hyperedges whose removal "disconnects" t from s; or finding a *maximum st-hyperflow*: a maximum number of hyperedge-disjoint st-hyperpaths. We show that many of these problems are APX-hard, even in acyclic sequence hypergraphs or with hyperedges of constant length. However, if all the hyperedges are of length at most 2, we show that these problems become polynomially solvable. We also study the special setting in which for every hyperedge there also is a hyperedge with the same sequence, but in reverse order. Finally, we briefly discuss other algorithmic problems such as finding a minimum spanning tree, or connected components.

Keywords: Sequence hypergraphs; colored graphs; labeled problems; transportation lines; algorithms; complexity

1 Introduction

Consider a public transportation network, e.g., a bus network, where every vertex corresponds to a bus stop, and where every bus line is specified as a fixed sequence

* An extended abstract of this paper appeared at WG 2016, and it was at a workshop of this lovely series on graph-theoretic concepts in computer science where the last author had the joy of meeting the jubilarian for the first time.

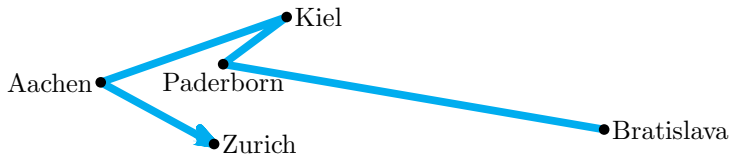

Fig. 1. A bus line (actually, the career path of the jubilarian) in a transportation network, and the corresponding hyperedge.

of bus stops. One can travel in the network by taking a bus and then following the stops in the order that is fixed by the corresponding bus line. See Fig. 1 for an illustration. Note that we think of a line as a sequence of stops in one direction only, since there might be one-way streets or other obstacles that cause that the bus can travel the stops in a single direction only. Then, interesting questions arise: How can one travel from s to t using the minimum number of lines? How many lines must break down, so that t is not reachable from s? Are there two ways to travel from s to t that both use different lines?

These kinds of questions are traditionally modeled and studied by algorithmic graph theory, but no model appears to capture all the necessary aspects of the problems above. We propose the following very natural way to extend the concept of directed graphs to hypergraphs.

A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ with an ordering of the vertices of every hyperedge is called a *sequence hypergraph*. Formally, the sequence hypergraph $\mathcal{H}$ consists of the set of vertices $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, and the set of *(sequence) hyperedges* $\mathcal{E} = \{E_1, E_2, \ldots, E_k\}$, where each hyperedge $E_i = (v_{i_1}, v_{i_2}, \ldots, v_{i_l})$ is defined as a sequence of vertices without repetition. We remark that this definition substantially differs from the commonly used definition of directed hypergraphs [1, 2, 14], where each directed hyperedge is a pair $(\mathrm{From}, \mathrm{To})$ of disjoint subsets of $\mathcal{V}$. We note that the order of vertices in a sequence hyperedge does not imply any order of the same vertices in other hyperedges. Furthermore, the sequence hypergraph does not impose any global order on $\mathcal{V}$.

There is another way to look at sequence hypergraphs coming from our motivation in transportation. For a sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, we construct a *directed colored multigraph* $G = (V, E, c)$ as follows. The set of vertices V is identical to $\mathcal{V}$, and for a hyperedge $E_i = (v_{i_1}, v_{i_2}, \ldots, v_{i_l})$ from $\mathcal{E}$, the multigraph G contains $l-1$ edges $(v_{i_j}, v_{i_{j+1}})$ for $j = 1, \ldots, l-1$, all colored with color $c(E_i)$, with $c(E_i) \neq c(E_{i'})$ for $i \neq i'$. Therefore, each edge of G is colored by one of the $k = |\mathcal{E}|$ different colors $\mathcal{C} = \{c(E_1), c(E_2), \ldots, c(E_k) \mid E_i \in \mathcal{E}\}$. Clearly, the edges of each color form a directed path in G. We refer to G as the *underlying colored graph* of $\mathcal{H}$. We denote by m the number of edges of G.

In this article, we study several standard graph-algorithmic problems in the setting of sequence hypergraphs. In particular, we consider the problem of finding a *shortest st-hyperpath*: an st-path that uses the minimum number of sequence hyperedges; the problem of finding a *minimum st-hypercut*: an st-cut that uses

Table 1. Summary of the complexities admitted by some classic problems in the setting of colored (labeled) graphs and sequence hypergraphs. The last row indicates whether the sizes of the maximum *st*-flow and the minimum *st*-cut equal in the considered setting. The cells in gray show our contribution.

	Colored/Labeled Graphs		Sequence Hypergraphs			
	General	Span 1	General	Acyclic	Backward	Length≤ 2
Shortest *st*-path	APX-hard	P	APX-hard	P	P	P
Minimum *st*-cut	APX-hard	P	APX-hard	APX-hard	NP-hard	P
Maximum *st*-flow	APX-hard	P	APX-hard	APX-hard	NP-hard	P
MaxFlow-MinCut Duality	×	√	×	×	×	√

the minimum number of sequence hyperedges; and the problem of finding a *maximum st-hyperflow*: a maximum number of hyperedge-disjoint *st*-hyperpaths.

We show that the shortest *st*-hyperpath is NP-hard to approximate within a factor of $(1-\varepsilon)\ln n$, for any $\varepsilon > 0$, in general sequence hypergraphs, but can be found in polynomial time if the given sequence hypergraph is acyclic (Section 3). On the other hand, we show that both maximum *st*-hyperflow and minimum *st*-hypercut are APX-hard to find even in acyclic sequence hypergraphs (Sections 4 and 5). We then consider sequence hypergraphs with sequence hyperedges of constant length, where the length of a hyperedge is the number of its vertices minus one. We note that the shortest *st*-hyperpath problem remains hard to approximate even with hyperedges of length at most 5, and we show that the maximum *st*-hyperflow problem remains APX-hard even with hyperedges of length at most 3. On the other hand, we show that if all the hyperedges are of length at most 2, all 3 problems become polynomially solvable (Section 6). We also study the complexity in a special setting in which for each hyperedge there also is a hyperedge with the same sequence, but in the opposite direction. We show that the shortest *st*-hyperpath problem becomes polynomially solvable, but both maximum *st*-hyperflow and minimum *st*-hypercut are NP-hard to find also in this setting, and we give a 2-approximation algorithm for the minimum *st*-hypercut problem (Section 7). Finally, we briefly study the complexity of other algorithmic problems, namely, finding a minimum spanning tree, or connected components, in sequence hypergraphs (Section 8). For a summary of the results see Table 1. The table also shows known results for the related labeled graphs (discussed below). The result on the APX-hardness of the shortest *st*-hyperpath problem (Theorem 1) appeared also, in a different context, in [4].

2 Related Work

Recently, there has been a lot of research concerning optimization problems in (multi)graphs with colored edges, where the cost of a solution is measured by the number of colors used, e.g., one may ask for an *st*-path using the minimum number of colors. The motivation comes from applications in optical or other communication networks, where a group of links (i.e., edges) can fail simultane-

ously and the goal is to find resilient solutions. Similar situations may occur in economics, when certain commodities are sold (and priced) in bundles.

Formally, *colored graphs* or *labeled graphs* are (mostly undirected) graphs where each edge has one color, and in general there is no restriction on a set of edges of the same color. Note that some of the studies consider a slightly different definition of colored graphs, where to each edge a set of colors is associated instead of a single color. Since the computational complexity of some problems may differ in the two models, the transformations between the two models have been investigated [8].

The *minimum label path* problem, which asks for an st-path with a minimum number of colors, is NP-hard and hard to approximate [5–7, 16, 17, 22]. The *2 label disjoint paths* problem, which asks for a pair of st-paths such that the sets of colors appearing on the two paths are disjoint, is NP-hard [18]. The *minimum label cut* problem, which asks for a set of edges with a minimum number of colors that forms an st-cut, is NP-hard and hard to approximate [7, 23]. The *minimum label spanning tree* problem, which asks for a spanning tree using edges of minimum number of colors, is NP-hard and hard to approximate [17, 20].

Hassin et al. [17] give a $\log(n)$-approximation algorithm for the minimum label spanning tree problem and a $\sqrt{n}$-approximation algorithm for the minimum label path problem, where n is the number of vertices of the input colored graph. Zhang et al. [23] give a $\sqrt{m}$-approximation algorithm for the minimum label cut problem, where m is the number of edges of the input colored graph. Fellows et al. [13] study the parameterized complexity of minimum label problems. Coudert et al. [7, 8] consider special cases when the *span* is 1, i.e., each color forms a connected component; or when the graph has a *star property*, i.e., the edges of every color are adjacent to one vertex.

Note that since most of these results consider undirected labeled graphs, they provide almost no implications on the complexity of similar problems in the setting of sequence hypergraphs. In our setting, not only we work with *directed* labeled graphs, but we also require edges of each color to form a directed path, which implies a very specific structure that, to the best of our knowledge, has not been considered in the setting of labeled graphs.

On the other hand, we are not the first to define hypergraphs with hyperedges specified as sequences of vertices. However, we are not aware of any work that would consider and explore this type of hypergraphs from an algorithmic graph theory point of view. In fact, mostly, these hypergraphs are taken merely as a tool, convenient to capture certain relations, but they are not studied further. We shortly list a few articles where sequence hypergraphs appeared, but we do not give details, since there is very little relation to our area of study. Berry et al. [3] introduce and describe the basic architecture of a software tool for (hyper)graph drawing. Wachman et al. [21] present a kernel for learning from *ordered hypergraphs*, a formalization that captures relational data as used in Inductive Logic Programming. Erdős et al. [12] study Sperner-families and as an

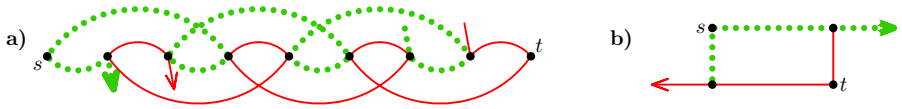

Fig. 2. In both figures, the green-dotted curve and the red solid curve depict two sequence hyperedges. **a)** The length of the st-hyperpath is 2, but the number of switches is 7. **b)** The st-hyperpath consists of two sequence hyperedges that also form a hypercycle.

application of a derived result they study the maximum number of edges of a so-called *directed Sperner-hypergraph*.

3 On the Shortest *st*-Hyperpath

In this section, we consider the shortest st-hyperpath problem in general sequence hypergraphs and in acyclic sequence hypergraphs.

Definition 1 (st-hyperpath). *Let s and t be two vertices of a sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$. A set of hyperedges $P \subseteq \mathcal{E}$ forms a* hyperpath *from s to t if the underlying (multi)graph G' of the subhypergraph $\mathcal{H}' = (\mathcal{V}, P)$ contains an st-path, and P is minimal with respect to inclusion. We call such an st-path an* underlying path *of P. The* length *of an st-hyperpath P is defined as the number of hyperedges in P. The* number of switches *of an st-hyperpath P is the minimum number of changes between the hyperedges of P, when following any underlying st-path of P.*

We note that each hyperpath may have multiple underlying paths. Also note that, even though the number of switches of an st-hyperpath P gives an upper bound on the length of P, the actual length of P can be much smaller than the number of switches of P (see Figure 2a).

Proposition 1. *Given a sequence hypergraph, and two vertices s and t, an st-hyperpath minimizing the number of switches can be found in polynomial time.*

Such an st-hyperpath can be found, e.g., by a modified Dijkstra algorithm (starting from s, following the outgoing sequence hyperedges and for each vertex storing the minimum number of switches necessary to reach it).

Conversely, we show that finding a shortest st-hyperpath (minimizing the number of hyperedges) is hard to approximate. On the other hand, if the given sequence hypergraph is acyclic, we show that the shortest st-hyperpath problem becomes polynomially solvable.

Definition 2 (acyclic sequence hypergraph). *A set of hyperedges $O \subseteq \mathcal{E}$ forms a* hypercycle*, if there are two vertices $a \neq b$ such that O contains both a hyperpath from a to b, and a hyperpath from b to a. A sequence hypergraph without hypercycles is called* acyclic*.*

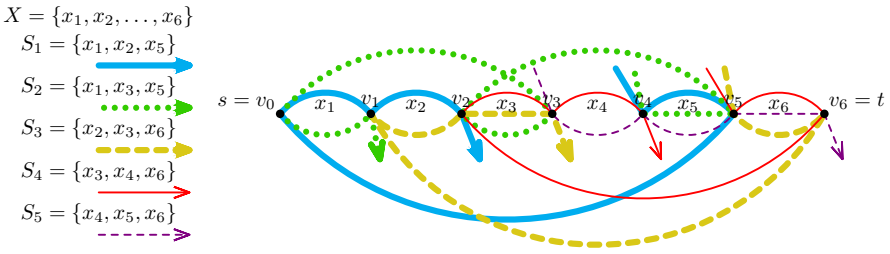

Fig. 3. Finding a shortest *st*-hyperpath is at least as hard as the minimum set cover problem.

Observe that an *st*-hyperpath may also be a hypercycle (see Figure 2b).

Definition 3 (edges of a hyperedge). *Let $E = (v_1, v_2, \ldots, v_k)$ be a hyperedge of a sequence hypergraph $\mathcal{H}$. We call the set of directed edges $\{e_i = (v_i, v_{i+1}) |\ i = 1, \ldots, k-1\}$ the* edges of E. *The edges of E are exactly the edges of color $c(E)$ in the underlying colored graph of $\mathcal{H}$. The* length *of a hyperedge is defined as the number of its edges (which is the number of its vertices minus one).*

For a fixed order $V^O = (v_1, v_2, \ldots, v_n)$ of vertices $\mathcal{V}$, an edge e of a hyperedge E is called a forward *edge with respect to V^O if its orientation agrees with the order V^O. Similarly, e is a* backward *edge if its orientation disagrees with V^O.*

Theorem 1. *Finding a shortest st-hyperpath in sequence hypergraphs is NP-hard to approximate within a factor of $(1-\varepsilon)\ln n$ for any $\varepsilon > 0$, unless $P = NP$. The problem remains APX-hard even if every hyperedge has length at most 5.*

Proof. We construct an approximation-preserving reduction from the set cover problem. The reduction is similar to that presented in [22] for the minimum label path problem in colored graphs. An instance $I = (X, \mathcal{S})$ of the set cover problem is given by a ground set $X = \{x_1, \ldots, x_n\}$, and a family of its subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$. The goal is to find a smallest subset $\mathcal{S}' \subseteq \mathcal{S}$ such that the union of the sets in $\mathcal{S}'$ contains all elements from X. The set cover problem is known to be NP-hard to approximate within a factor of $(1-\epsilon)\ln n$, unless $P = NP$ [10]. Moreover, if each subset of $\mathcal{S}$ is of size at most 3, the problem remains APX-hard [9].

From I we construct a sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ as follows (cf. Figure 3 along with the construction). The set of vertices $\mathcal{V} = \{v_0, v_1, \ldots, v_n\}$ contains one vertex v_i for each element x_i of the ground set X, plus one additional vertex v_0. Let V^O be the order of vertices in $\mathcal{V}$ naturally defined by their indices. The set of sequence hyperedges $\mathcal{E} = \{E_1, \ldots, E_m\}$ contains one hyperedge for each set in $\mathcal{S}$. For a set $S_i \in \mathcal{S}$, consider the set of vertices that correspond to the elements in S_i and order them according to V^O, to obtain a sequence $Q = (v_{i_1}, v_{i_2}, \ldots, v_{i_r})$, where $i_1 < i_2 < \cdots < i_r$. First, let us consider the simplest case where none of the v_{i_j} and $v_{i_{j+1}}$ (for $j = 1, \ldots, r-1$) are consecutive in the order V^O, that is, $i_j + 1 \neq i_{j+1}$. Then the sequence of the hyperedge

E_i corresponding to S_i is $(v_{i_r-1}, v_{i_r}, v_{i_{(r-1)}-1}, v_{i_{(r-1)}}, \ldots, v_{i_1-1}, v_{i_1})$ (e.g., the hyperedge corresponding to S_2 in Figure 3). In other words, E_i consists of forward edges: one forward edge (v_{i_j-1}, v_{i_j}) for each v_{i_j} in Q; and backward edges that connect the forward edges in the order opposite to Q. Now, for the more general case, if for some j, v_{i_j} and $v_{i_{j+1}}$ are consecutive vertices with respect to V^O, i.e., $i_j + 1 = i_{j+1}$, the sequence constructed as above would contain vertices repeatedly, which is not allowed (each sequence hyperedge has to be mapped to a path in the underlying graph). To avoid this, we construct E_i as follows. For simplicity of the explanation, instead of describing the sequence of the hyperedge, we specify E_i by listing the edges of the hyperedge and the path to which E_i is mapped. The hyperedge E_i consists of the same forward edges as before: one forward edge (v_{i_j-1}, v_{i_j}) for each v_{i_j} in Q. Whenever two or more vertices of Q are consecutive in V^O, their corresponding forward edges form a path. Clearly, the forward edges of E_i then determine a set of (non-overlapping) paths $p_1, p_2, \ldots, p_{r'}$ (uniquely ordered according to V^O). The backward edges of E_i then connect these paths in the order opposite to V^O into a single path (which specifies E_i). In particular, the last vertex of $p_{r'}$ connects to the first vertex of $p_{r'-1}$, the last vertex of $p_{r'-1}$ connects to the first vertex of $p_{r'-2}$, $\ldots$, and the last vertex of p_2 connects to the first vertex of p_1.

Note that the length of each sequence hyperedge E_i is bounded by $2|S_i| - 1$, where $|S_i|$ is the size of the set $S_i \in \mathcal{S}$ corresponding to E_i. This follows from the fact that E_i consists of $|S_i|$ forward edges and at most $|S_i| - 1$ backward edges to connect the forward edges. In particular, if each subset of $\mathcal{S}$ is of size at most 3, all the hyperedges are of length at most 5.

We set the source vertex s to v_0, and the target vertex t to v_n, and we show that a shortest st-hyperpath in $\mathcal{H}$ of length k provides a minimum set cover for I of the same size, and vice versa. First, notice that all the forward edges (with respect to V^O) of the hyperedges in $\mathcal{E}$ are of the form (v_i, v_{i+1}) for some $i = 0, \ldots, n-1$. Together with the fact that t is smaller than s in the order V^O, it follows that any path from s to t in the underlying graph of $\mathcal{H}$ goes via all the vertices, in the order V^O. Thus, there is an underlying path p of the shortest st-hyperpath P in $\mathcal{H}$, such that p does not use any backward edges of the hyperedges in $\mathcal{E}$. Clearly, by choosing a hyperedge E_i into the st-hyperpath P, one also chooses its forward edges and this way "covers" some sections of the underlying path of P. Since there is a one-to-one mapping between the hyperedges in $\mathcal{E}$ and the sets in $\mathcal{S}$, by finding an st-hyperpath P of length k, one finds a set cover of size k for the given instance. On the other hand, each set cover of size k can be mapped, using the same direct one-to-one mapping in the opposite direction, to an st-hyperpath of length k.

Thus, the described reduction is approximation-preserving. By reducing from the general set cover problem we obtain the first part of the claim, and by reducing from the set cover problem with all subsets of size at most 3 we obtain the second part of the claim. □

We complement the hardness result with a positive one.

Theorem 2. *The problem of finding the shortest st-hyperpath in acyclic sequence hypergraphs can be solved in polynomial time.*

Proof. Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be an acyclic sequence hypergraph. Since $\mathcal{H}$ is acyclic, let V^O be an order of the vertices $\mathcal{V}$ such that all the edges of each hyperedge are forward edges with respect to this order. This implies that for every st-hyperpath, there is an underlying path where all the edges of each hyperedge appear consecutively (the last edge of a hyperedge E appearing in an underlying path is reachable by E from the first appearing edge of E). Therefore, finding the shortest st-hyperpath P in $\mathcal{H}$ is the same as finding a hyperpath minimizing the number of switches, which can be done in polynomial time by Proposition 1. □

4 On the Maximum *st*-Hyperflow

We consider the problem of finding a number of hyperedge-disjoint st-hyperpaths. Capturing a similar relation as in graphs (between a set of k edge-disjoint st-paths and an st-flow of size k, when all the capacities are 1), for simplicity and brevity, we refer to a set of hyperedge-disjoint st-hyperpaths as an st-hyperflow.

Definition 4 (st-hyperflow). *Let s and t be two vertices of a sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$. Let $\mathcal{F} \subseteq 2^{\mathcal{E}}$ be a set of pairwise hyperedge-disjoint st-hyperpaths $\mathcal{F} = \{P_1, \ldots, P_k\}$. Then, $\mathcal{F}$ is an* st-hyperflow *of size $|\mathcal{F}| = k$.*

We show that deciding whether a given sequence hypergraph contains an st-hyperflow of size 2 is NP-hard, and thus finding a maximum st-hyperflow is inapproximable within a factor of $2 - \varepsilon$ unless P=NP. This remains true even for acyclic sequence hypergraphs with all the hyperedges of length at most 3.

Theorem 3. *Given an acyclic sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ with all hyperedges of length at most 3, and two vertices s and t, it is NP-hard to decide whether there are two hyperedge-disjoint st-hyperpaths.*

Proof. We construct a reduction from the NP-complete 3-Sat problem [15]. Let I be an instance of the 3-Sat problem, given as a set of m clauses $C = \{c_1, \ldots, c_m\}$ over a set $X = \{x_1, \ldots, x_n\}$ of Boolean variables. Recall that the goal of the 3-Sat problem is to find an assignment to the variables of X that satisfies all clauses of C.

From I we construct a sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ as follows (cf. Figure 4 along with the construction). The set $\mathcal{V}$ consists of $2 + (m+1) + (n+1) + \sum_{c_i \in C} |c_i|$ vertices: a source vertex s and a target vertex t; a vertex c_i for each clause $c_i \in C$ and a dummy vertex c_{m+1}; a vertex x_j for each variable $x_j \in X$ and a dummy vertex x_{n+1}; and finally a vertex $x_j c_i$ for each pair (x_j, c_i) such that $x_j \in c_i$, and similarly, $\overline{x_j} c_i$ for each $\overline{x_j} \in c_i$. Let us fix an arbitrary order C^O of the clauses in C. The set $\mathcal{E}$ consists of $4 + 2n + |I|$ hyperedges: There are 2 *source hyperedges* (s, c_1) and (s, x_1), and 2 *target hyperedges* (c_{m+1}, t) and (x_{n+1}, t). There are $2n$ *auxiliary hyperedges* $(x_i, x_i c_k)$ and $(x_i, \overline{x_i} c_{k'})$ for $i = 1, \ldots, n$, where c_k, or $c_{k'}$ is always the first clause (with respect to C^O)

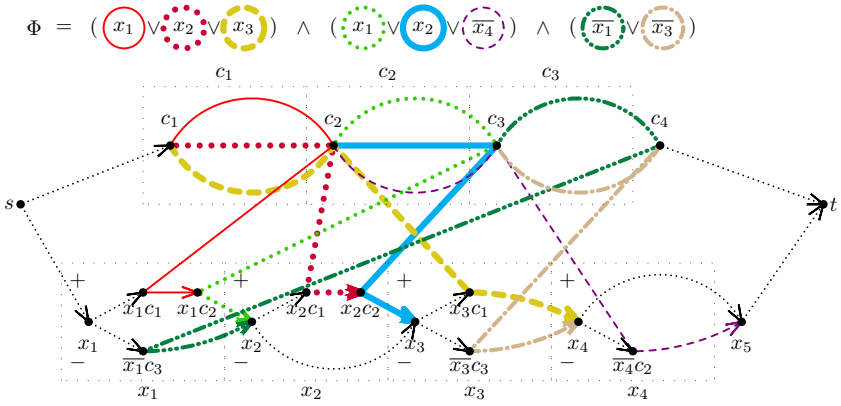

Fig. 4. Deciding st-hyperflow of size 2 is at least as hard as 3-SAT.

containing x_i or $\overline{x_i}$, respectively. If there is no clause containing x_i (or $\overline{x_i}$), the corresponding auxiliary hyperedge is (x_i, x_{i+1}). Finally, there are $|I|$ *lit-in-clause hyperedges* as follows. For each appearance of a variable x_j in a clause c_i as a positive literal there is one lit-in-clause hyperedge $(c_i, c_{i+1}, x_j c_i, x_j c_k)$, where c_k is the next clause (with respect to C^O) after c_i where x_j appears as a positive literal (in case, there is no such c_k, then the hyperedge ends in x_{j+1} instead). Similarly, if x_j is in c_i as a negative literal, there is one lit-in-clause hyperedge $(c_i, c_{i+1}, \overline{x_j} c_i, \overline{x_j} c_k)$, where c_k is the next clause containing the negative literal $\overline{x_j}$ (or it ends in x_{j+1}).

Clearly, each hyperedge is of length at most 3. We now observe that the constructed sequence hypergraph $\mathcal{H}$ is acyclic. All the hyperedges of $\mathcal{H}$ agree with the following order: the source vertex s; all the vertices $c_i \in C$ ordered according to C^O, and the dummy vertex c_{m+1}; the vertex x_1 followed by all the vertices $x_1 c_i$ ordered according to C^O, and then followed by the vertices $\overline{x_1} c_i$ again ordered according to C^O; the vertex x_2 followed by all $x_2 c_i$ and then all $\overline{x_2} c_i$; …; the vertex x_n followed by all $x_n c_i$ and then all $\overline{x_n} c_i$; and finally the dummy vertex x_{n+1}; and the target vertex t.

We show that the formula I is satisfiable if and only if $\mathcal{H}$ contains two hyperedge-disjoint st-hyperpaths. There are 3 possible types of st-paths in the underlying graph of $\mathcal{H}$: the first one leads through all the vertices $c_1, c_2, \ldots, c_{m+1}$ in this order; the second one leads through all the vertices $x_1, x_2, \ldots, x_{m+1}$ in this order and between x_j and x_{j+1} it goes either through all the $x_j c_*$ (here, and later, $*$ is used as a wildcard) vertices or through all the $\overline{x_j} c_*$ vertices (this may differ for different j); and the third possible st-path starts the same as the first option and ends as the second one. Based on this observation, notice that there can be at most 2 hyperedge-disjoint st-hyperpaths: necessarily, one of them has an underlying path of the first type, while the other one has an underlying path of the second type.

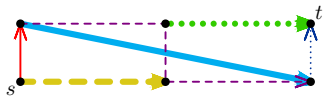

Fig. 5. Acyclic sequence hypergraph with minimum st-hypercut of size 2, and no two hyperedge-disjoint st-hyperpaths.

From a satisfying assignment A to the variables of I we can construct the two disjoint st-hyperpaths as follows. The underlying path of one hyperpath leads from s to t via the vertices $c_1, c_2, \ldots, c_{m+1}$, and to move from c_i to c_{i+1} it uses a lit-in-clause hyperedge that corresponds to a pair (l, c_i) such that l is one of the literals that satisfy the clause c_i in A. The second hyperpath has an underlying path of the second type, it leads via $x_1, x_2, \ldots, x_{n+1}$, and from x_j to x_{j+1} it uses the vertices containing only the literals that are not satisfied by the assignment A. Thus, the second hyperpath uses only those lit-in-clause hyperedges that correspond to pairs containing literals that are not satisfied by A. This implies that the two constructed st-hyperpaths are hyperedge-disjoint.

Let P and Q be two hyperedge-disjoint st-hyperpaths of $\mathcal{H}$. Let P have an underlying path p of the first type and Q have an underlying path q of the second type. We can construct a satisfying assignment for I by setting to `TRUE` the literals opposite to those that occur in the vertices on q. Then, the hyperpath P describes how the clauses of I are satisfied by this assignment. □

5 On the Minimum *st*-Hypercut

Quite naturally, we define an st-hypercut of a sequence hypergraph $\mathcal{H}$ as a set C of hyperedges whose removal from $\mathcal{H}$ leaves s and t disconnected.

Definition 5 (st-hypercut). *Let s and t be two vertices of a sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$. A set of hyperedges $X \subseteq \mathcal{E}$ is an* st-hypercut *if the subhypergraph $\mathcal{H}' = (\mathcal{V}, \mathcal{E} \setminus X)$ does not contain any hyperpath from s to t. The* size *of an st-hypercut X is $|X|$, i.e., the number of hyperedges in X.*

For directed (multi)graphs, the famous MaxFlow-MinCut Duality Theorem [11] states that the size of a maximum st-flow is equal to the size of a minimum st-cut. In sequence hypergraphs, this duality does not hold, not even in acyclic sequence hypergraphs as Figure 5 shows. But, of course, the size of any st-hyperflow is a lower bound on the size of any st-hypercut. We showed the maximum st-hyperflow problem to be APX-hard even in acyclic sequence hypergraphs (see Theorem 3). It turns out that also the minimum st-hypercut problem in acyclic sequence hypergraphs is APX-hard.

Theorem 4. *Minimum st-hypercut in acyclic sequence hypergraphs is NP-hard to approximate within a factor of $2 - \varepsilon$ under UGC, or within a factor $7/6 - \varepsilon$ unless P=NP.*

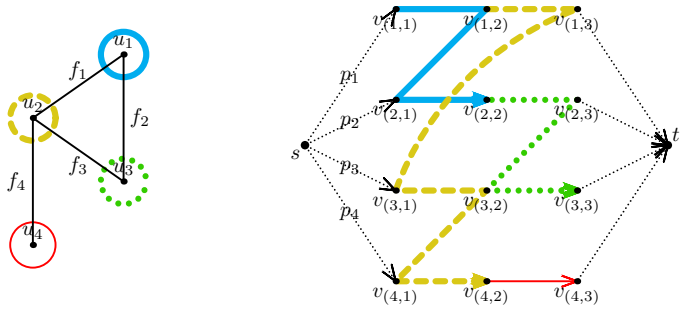

Fig. 6. Minimum st-hypercut is at least as hard as minimum vertex cover.

Proof. We construct an approximation-preserving reduction from the vertex cover problem, which has the claimed inapproximability [19]. An instance of the vertex cover problem is an undirected graph $I = (U, F)$, with the vertex set $U = \{u_1, \ldots, u_n\}$ and the edge set $F = \{f_1, \ldots, f_m\}$. The goal is to find a smallest subset $U' \subseteq U$ such that U contains at least one vertex from each edge $f \in F$.

To construct from the instance I an instance I' of the minimum st-hypercut problem in acyclic sequence hypergraphs, we fix an order of the vertices and edges of I as follows. Let $U^O = (u_1, \ldots, u_n)$ be an arbitrary order on the vertices of U. From every edge in F we create an ordered edge, where the vertices of the edge are ordered naturally according to U^O. Let F' denote the set of the created ordered edges, and let F^O be the edges F' ordered lexicographically according to U^O.

We construct the sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ of I' as follows (cf. Figure 6 along with the construction). The set of vertices $\mathcal{V}$ consists of $3m + 2$ vertices: a source vertex s, a target vertex t, and for each edge $f_i \in F^O$, $i = 1, \ldots, m$, there are three vertices $v_{(i,1)}, v_{(i,2)}$, and $v_{(i,3)}$. The set of hyperedges $\mathcal{E}$ consists of $2m+n$ hyperedges. There are m *source hyperedges* of the form $(s, v_{(i,1)})$, each of them connects s to one vertex $v_{(i,1)}$. There are m *target hyperedges* of the form $(v_{(i,3)}, t)$, each of them connects one vertex $v_{(i,3)}$ to t. And finally, there are n *vertex hyperedges*, each corresponding to one of the vertices in U, and is constructed as follows.

For a vertex $u \in U$ of I, we describe the sequence of vertices in the corresponding vertex hyperedge E_u iteratively. We start with an empty sequence. We consider the edges F^O in order, and for each edge $f_i \in F^O$ that contains u we prolong the sequence of E_u as follows. If f_i contains u as the first vertex, we append $v_{(i,1)}$ and $v_{(i,2)}$ to E_u. Otherwise, f_i contains u as the second vertex, and we append $v_{(i,2)}$ and $v_{(i,3)}$ to E_u. Now consider the edges of the obtained hyperedge E_u and let us distinguish two types. First, there are edges of the form $(v_{(i,1)}, v_{(i,2)})$ and $(v_{(i,2)}, v_{(i,3)})$ for some i, and second, there are edges of the form $(v_{(i,2)}, v_{(j,1)})$, $(v_{(i,3)}, v_{(j,2)})$, and $(v_{(i,3)}, v_{(j,1)})$ for some $i < j$. Note that, due to the fact that the edges in F^O are ordered lexicographically, all the

edges of the vertex hyperedge E_u take one of the forms described above. Also note that, due to the direct correspondence between the vertices in U and vertex hyperedges, each tuple $(v_{(i,j)}, v_{(i,j+1)})$ is part of (i.e., an edge of) exactly one of the hyperedges.

Clearly, by the construction, the sequence hypergraph is acyclic, since the ordering $V^O = (s, v_{(1,1)}, v_{(1,2)}, v_{(1,3)}, \ldots, v_{(m,1)}, v_{(m,2)}, v_{(m,3)}, t)$ is a topological sorting of the underlying graph G.

Let us now observe that there always exists a minimum st-hypercut that does not contain any source or target hyperedges. Notice that in the underlying graph G of $\mathcal{H}$, the only outgoing edge from $v_{(i,1)}$ leads to $v_{(i,2)}$, for $i = 1, \ldots, m$, and similarly, the only incoming edge to $v_{(i,3)}$ comes from $v_{(i,2)}$. Since for each i, the tuple $(v_{(i,j)}, v_{(i,j+1)})$ is an edge of exactly one hyperedge, every source hyperedge $(s, v_{(i,1)})$ in an st-hypercut C can be substituted by the vertex hyperedge containing the edge $(v_{(i,1)}, v_{(i,2)})$; and every target hyperedge $(v_{(i,3)}, t)$ can be substituted by the vertex hyperedge containing the edge $(v_{(i,2)}, v_{(i,3)})$; and the resulting set is an st-hypercut of size equal or smaller than C. Thus, there exists an optimal solution that contains only vertex hyperedges.

Now observe that any minimum st-hypercut C consisting of vertex hyperedges only, must for each $i = 1, \ldots, m$ "hit" either the edge $(v_{(i,1)}, v_{(i,2)})$ or $(v_{(i,2)}, v_{(i,3)})$ (i.e., one of those two edges is an edge of some hyperedge in C). Otherwise, the underlying graph of $(\mathcal{V}, \mathcal{E} \setminus C)$ would contain an st-path $p_i = s, v_{(i,1)}, v_{(i,2)}, v_{(i,3)}, t$.

We show that the described construction gives us an approximation-preserving reduction: for an st-hypercut of size k, we can construct a solution for the vertex cover problem of the same size, and vice versa. Let $\mathcal{S}' \subseteq \mathcal{H}$ be an optimal solution to the instance I' that contains only vertex hyperedges. Recall that there is a direct one-to-one mapping between the vertex hyperedges and the vertices U of the instance I. There is also a direct mapping between each triple $(v_{(i,1)}, v_{(i,2)}, v_{(i,3)})$ and an edge from F. Since the solution $\mathcal{S}'$ hits one of the edges $(v_{(i,1)}, v_{(i,2)})$ or $(v_{(i,2)}, v_{(i,3)})$ for each i, we can use the mapping to construct a solution $\mathcal{S} \subseteq U$ to the instance I of the original minimum vertex cover problem, such that $|\mathcal{S}| = |\mathcal{S}'|$. On the other hand, every solution to the original vertex cover instance can be mapped, using the same direct one-to-one mapping in the opposite direction, to an st-hypercut of $\mathcal{H}$ of the same size. □

6 Sequence Hypergraphs with Hyperedges of Length ≤ 2

We have seen that some of the classic, polynomially solvable problems in (directed) graphs become APX-hard in sequence hypergraphs. Note that this often remains true even if all the hyperedges are of constant length. In particular, Theorem 1 states that the shortest st-hyperpath is hard to approximate even if all the hyperedges are of length at most 5; Figure 4 illustrates that the duality between minimum st-hypercut and maximum st-hyperflow breaks already with a single hyperedge of length 3; and Theorem 3 yields that the maximum st-hyperflow is hard to approximate even if all hyperedges are of length at most 3.

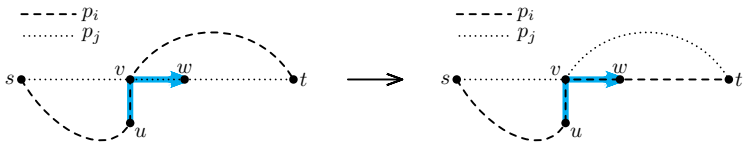

Fig. 7. Transforming st-paths into hyperedge-disjoint st-hyperpaths.

It is an interesting question to investigate the computational complexity of the problems for hyperedge lengths smaller than 5 or 3. We show that, if all the hyperedges of the given sequence hypergraph are of length at most 2, the shortest st-hyperpath, the minimum st-hypercut, and the maximum st-hyperflow can all be found in polynomial time.

Theorem 5. *The shortest st-hyperpath problem in sequence hypergraphs with hyperedges of length at most 2 can be solved in polynomial time.*

Proof. Consider a shortest st-hyperpath P in a given sequence hypergraph with hyperedges of length at most 2. Clearly, whenever both edges of a hyperedge are part of an underlying path of P, they must appear consecutively on it. Thus, all the edges of each hyperedge appear consecutively on any underlying path of P. Therefore, the length of the shortest st-hyperpath P is again the same as the minimum number of switches of P, and such a shortest st-hyperpath can be found in polynomial time (Proposition 1). □

Theorem 6. *The maximum st-hyperflow problem in sequence hypergraphs with hyperedges of length at most 2 can be solved in polynomial time.*

Proof. Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a sequence hypergraph with hyperedges of length at most 2, and let s and t be two of its vertices. Then, using standard graph algorithms we can find a maximum st-flow f in the underlying directed multigraph G of $\mathcal{H}$ with edge capacities 1. Thus, the flow f of size $|f|$ gives us a set of $|f|$ edge-disjoint st-paths $p_1, \ldots, p_{|f|}$ in G (note that any directed cycle in f can be easily removed).

We iteratively transform $p_1, \ldots, p_{|f|}$ into a set of st-paths such that all the edges of each hyperedge appear on only one of these paths. Let $E = (u, v, w)$ be a hyperedge that lies on two different paths (see Figure 7), i.e., $(u, v) \in p_i$ and $(v, w) \in p_j$, for some $i, j \in [|f|]$. Then, p_i consists of an su-path, edge (u, v), and a vt-path. Similarly, p_j consists of an sv-path, edge (v, w), and a wt-path. Since all these paths and edges are pairwise edge-disjoint, by setting p_i to consist of the su-path, edge (u, v), edge (v, w), and the wt-path, and at the same time setting p_j to consist of the sv-path, and the vt-path, we again obtain two edge-disjoint st-paths p_i and p_j. However, now the hyperedge E is present only on p_i. At the same time, since each hyperedge is of length at most 2, all the edges of any hyperedge appear on any st-path consecutively, and any hyperedge that was present on only one of p_i or p_j is not affected by the above rerouting and still is present on one of the two paths only.

Thus, the rerouting decreased the number of hyperedges present on more paths, and after at most $|\mathcal{E}|$ iterations of this transformation we obtain $|f|$ hyperedge-disjoint st-paths, which gives us an st-hyperflow of size $|f|$. It is easy to observe that the size of the hyperflow is bounded from above by the size of the flow in the underlying multigraph. Thus, we obtain a maximum st-hyperflow in $\mathcal{H}$. □

Theorem 7. *The minimum st-hypercut problem in sequence hypergraphs with hyperedges of length at most 2 can be solved in polynomial time.*

Proof. Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a sequence hypergraph with hyperedges of length at most 2, and let s and t be two of its vertices. As in proof of Theorem 6, we find a maximum st-flow f (of size $|f|$) in the underlying directed multigraph G of $\mathcal{H}$ and obtain a maximum st-hyperflow F in $\mathcal{H}$ of the same size, i.e., $|F| = |f|$. Since in directed multigraphs the size of the minimum cut equals the size of the maximum flow [11], it follows that we can find $|F|$ edges $e_1, \ldots, e_{|F|}$ of G that form a minimum cut of G. Observe that each of these edges corresponds to exactly one hyperedge. Thus, we obtain a set C of at most $|F|$ hyperedges that forms an st-hypercut. Since the size of any st-hypercut is bounded from below by the size of the hyperflow, C is a minimum st-hypercut. □

Note that we proved Theorem 7 by first constructing an st-hyperflow and then finding an st-hypercut of the same size. Since this is always possible, it follows that the equivalent of the MaxFlow-MinCut Duality Theorem holds in this setting with hyperedges of length at most 2.

7 Sequence Hypergraphs with Backward Hyperedges

We consider a special class of sequence hypergraphs where for every hyperedge, there is the exact same hyperedge, but oriented in the opposite direction.

Definition 6 (backward hyperedges). *Let $E = (v_1, v_2, \ldots, v_k)$ be a hyperedge of a sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$. We say that E' is a* backward hyperedge[5] *of E if $E' = (v_k, \ldots, v_2, v_1)$. If for every E of $\mathcal{E}$, there is exactly one backward hyperedge in $\mathcal{E}$, we refer to $\mathcal{H}$ as* sequence hypergraph with backward hyperedges.

Such a situation arises naturally in urban public transportation networks, for instance most of the tram lines of the city of Zurich have also a "backward" line (which has the exact same stops as the "forward" line, but goes in the opposite direction). We study the complexities of shortest st-hyperpath, minimum st-hypercut, and maximum st-hyperflow under this setting.

We show that, in this setting, we can find a shortest st-hyperpath in polynomial time. On the other hand, we show that minimum st-hypercut and maximum st-hyperflow remain NP-hard, and we give a 2-approximation algorithm

[5] Note, if E' is a backward hyperedge of E, also E is a backward hyperedge of E'.

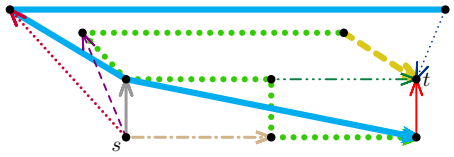

Fig. 8. Sequence hypergraph with backward hyperedges with minimum st-hypercut of size 4, and only three hyperedge-disjoint st-hyperpaths. For every displayed hyperedge, there is also a backward hyperedge, which is for simplicity omitted from the figure.

for the minimum st-hypercut. Also observe in Figure 8 that the equivalent of the MaxFlow-MinCut Duality Theorem does not hold in sequence hypergraphs with backward hyperedges. The positive results in this section are based on existing algorithms for standard hypergraphs, the negative results are obtained by a modification of the hardness proofs given in Sections 4 and 5.

Theorem 8. *The shortest st-hyperpath problem in sequence hypergraphs with backward hyperedges can be solved in polynomial time.*

Proof. Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a sequence hypergraph with backward hyperedges, and let s and t be two vertices of $\mathcal{H}$. We construct a (standard) hypergraph $\mathcal{H}^* = (\mathcal{V}^* = \mathcal{V}, \mathcal{E}^*)$ from $\mathcal{H}$ in such a way that for each sequence hyperedge E of $\mathcal{E}$, $\mathcal{E}^*$ contains a (non-oriented) hyperedge E^* that corresponds to the set of vertices of E. Note that E and its backward hyperedge E' consist of the same set of vertices, thus the corresponding E^* and E'^* are the same. A shortest st-hyperpath[6] P^* in the (standard) hypergraph $\mathcal{H}^*$ can be found in polynomial time. Observe that the size of P^* gives us a lower bound $|P^*|$ on the length of the shortest path in the sequence hypergraph $\mathcal{H}$.

In fact, we can construct from P^* an st-hyperpath in $\mathcal{H}$ of size $|P^*|$ as follows. Let us fix p^* to be an underlying path of P^*. Let $(s = v_1, v_2, \ldots, v_{|P^*|+1} = t)$ be a sequence of vertices, subsequence of p^*, such that for each $i = 1, \ldots, |P^*|$, there is a hyperedge E^* in P^* that contains both v_i and v_{i+1}, v_i is the first vertex of E^* seen on p^*, and v_{i+1} is the last vertex of E^* seen on p^*. Since every hyperedge E^* of $\mathcal{E}^*$ corresponds to the set of vertices of some hyperedge E of $\mathcal{E}$, there is a sequence of sequence hyperedges $(E_1, E_2, \ldots, E_{|P^*|})$, $E_i \in \mathcal{E}$, such that v_i, v_{i+1} are vertices in E_i. Since $\mathcal{H}$ is a sequence hypergraph with backward hyperedges, for every hyperedge E of $\mathcal{E}$ and a pair of vertices v_i and v_{i+1} of E, there is a $v_i v_{i+1}$-hyperpath in $\mathcal{H}$ of size 1, which consists of E or its backward hyperedge E'. Therefore, there is an st-hyperpath of size $|P^*|$ in $\mathcal{H}$. □

Theorem 9. *The maximum st-hyperflow problem in sequence hypergraphs with backward hyperedges is NP-hard.*

[6] An st-hyperpath P^* and its underlying path are defined as in sequence hypergraphs.

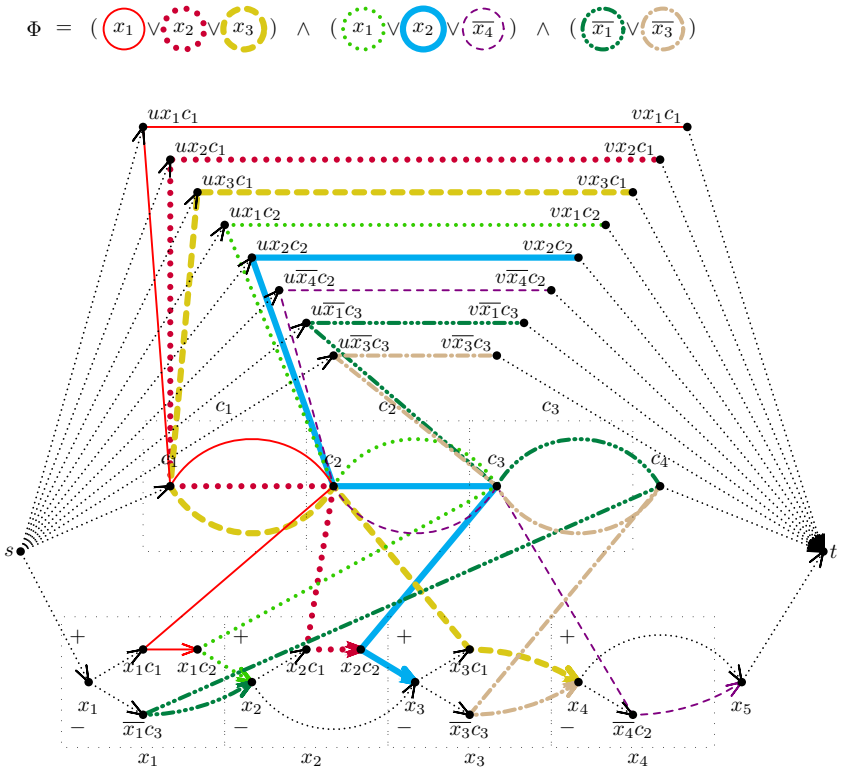

Fig. 9. Maximum *st*-hyperflow in sequence hypergraphs with backward hyperedges is NP-hard.

Proof. We construct a reduction from the NP-complete 3-Sat problem [15]. Let I be an instance of the 3-Sat problem, given as a set of m clauses $C = \{c_1, \ldots, c_m\}$ over a set $X = \{x_1, \ldots, x_n\}$ of Boolean variables.

From I we construct a sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ as follows (cf. Figure 9 along with the construction). The construction is very similar to that in the proof of Theorem 3, so we highlight the changes in bold. One major change is that now for every hyperedge there is a backward hyperedge. For simplicity, we divide all the sequence hyperedges into pairs of mutually backward sequence hyperedges, and we refer to one sequence hyperedge of each pair as *forward* hyperedge (and to the other as its *backward* hyperedge). For simplicity of the construction, we describe explicitly only the forward hyperedges, and each of them implicitly defines a backward hyperedge.

Let $|I|$ be the size of I (i.e., $|I| = \sum_{c_i \in C} |c_i|$). The set of vertices $\mathcal{V}$ consists of $2+(m+1)+(n+1)+\mathbf{3}|I|$ vertices: a source vertex s and a target vertex t; a vertex c_i for each clause $c_i \in C$ and a dummy vertex c_{m+1}; a vertex x_j for each variable $x_j \in X$ and a dummy vertex x_{n+1}; and finally **three** vertices $x_j c_i$, $\mathbf{u x_j c_i}$, and $\mathbf{v x_j c_i}$ for each pair (x_j, c_i) such that $x_j \in c_i$, and similarly, $\overline{x_j} c_i$, $\mathbf{u \overline{x_j} c_i}$, and $\mathbf{v \overline{x_j} c_i}$

for each $\overline{x_j} \in c_i$. Let us fix an arbitrary order C^O of the clauses in C. The set of hyperedges $\mathcal{E}$ contains $4+2n+\mathbf{3}|I|$ forward hyperedges (plus the same amount of the corresponding backward hyperedges that we do not specify explicitly). There are $2 + |\mathbf{I}|$ *source hyperedges:* (s, c_1) and (s, x_1); for each pair (x_j, c_i), $x_j \in c_i$, there is $(\mathbf{s}, \mathbf{ux_jc_i})$, and for each $\overline{x_j} \in c_i$, there is $(\mathbf{s}, \mathbf{u\overline{x_j}c_i})$. There are $2 + |\mathbf{I}|$ *target hyperedges:* (c_{m+1}, t) and (x_{n+1}, t); for each pair (x_j, c_i), $x_j \in c_i$, there is $(\mathbf{vx_jc_i}, \mathbf{t})$, and for each $\overline{x_j} \in c_i$, there is $(\mathbf{v\overline{x_j}c_i}, \mathbf{t})$. There are $2n$ *auxiliary hyperedges* (x_i, x_ic_k) and $(x_i, \overline{x_i}c_{k'})$ for $i = 1, \ldots, n$, where c_k or $c_{k'}$ is always the first clause (with respect to C^O) containing x_i or $\overline{x_i}$, respectively. In case there is no clause containing x_i (or $\overline{x_i}$), the corresponding auxiliary hyperedge is (x_i, x_{i+1}). Finally, there are $|I|$ *lit-in-clause hyperedges* as follows. For each appearance of a variable x_j in a clause c_i as a positive literal there is one lit-in-clause hyperedge $(\mathbf{vx_jc_i}, \mathbf{ux_jc_i}, c_i, c_{i+1}, x_jc_i, x_jc_k)$, where c_k is the next clause (with respect to C^O) after c_i where x_j appears as a positive literal (in case there is no such c_k, the hyperedge ends in x_{j+1} instead). Similarly, if x_j is in c_i as a negative literal, there is one lit-in-clause hyperedge $(\mathbf{v\overline{x_j}c_i}, \mathbf{u\overline{x_j}c_i}, c_i, c_{i+1}, \overline{x_j}c_i, \overline{x_j}c_k)$, where c_k is the next clause containing the negative literal $\overline{x_j}$ (or the hyperedge ends in x_{j+1}).

We show that the formula I is satisfiable if and only if the sequence hypergraph $\mathcal{H}$ contains $2 + |\mathbf{I}|$ hyperedge-disjoint st-hyperpaths. Since there are exactly $2 + |\mathbf{I}|$ source hyperedges, and no other hyperedge (including backward hyperedges) originates from the source vertex s, all these source hyperedges have to be used to get $2 + |I|$ hyperedge-disjoint st-hyperpaths. Similarly, all the target hyperedges have to be used. But then, each of the $|I|$ vertices vx_jc_i or $v\overline{x_j}c_i$ has to be on one of the underlying st-paths. However, vx_jc_i can only be reached (unless passing via t) from ux_jc_i using a backward hyperedge of a lit-in-clause hyperedge. Similarly, $v\overline{x_j}c_i$ can only be reached from $u\overline{x_j}c_i$ using a backward hyperedge of a lit-in-clause hyperedge. This all implies that there can be $2 + |I|$ hyperedge-disjoint st-hyperpaths only if $|I|$ of them are composed in one of the two following ways: a source hyperedge (s, ux_jc_i), a backward hyperedge of some lit-in-clause hyperedge to get from ux_jc_i to vx_jc_i, and a target hyperedge (vx_jc_i, t); or a source hyperedge $(s, u\overline{x_j}c_i)$, a lit-in-clause backward hyperedge to get from $u\overline{x_j}c_i$ to $v\overline{x_j}c_i$, and a target hyperedge $(v\overline{x_j}c_i, t)$. Thus, the backward hyperedges of all $|I|$ lit-in-clause hyperedges are used and cannot appear in the remaining st-hyperpaths. Also note that all other backward hyperedges are useless to reach t from s, since they lead only backwards. This implies a situation equivalent to that in the proof of Theorem 3. That is, the formula I is satisfiable if and only if the sequence hypergraph $\mathcal{H}$ contains 2 hyperedge-disjoint st-hyperpaths, when considering forward hyperedges only, without the $|I|$ source and $|I|$ target hyperedges already used above.

Then, there are 3 possible types of st-paths in the underlying graph of $\mathcal{H}$: the first one leads through all the vertices $c_1, c_2, \ldots, c_{m+1}$ in this order; the second one leads through all the vertices $x_1, x_2, \ldots, x_{m+1}$ in this order and between x_j and x_{j+1} it goes either through all the x_jc_* vertices or through all the $\overline{x_j}c_*$ vertices (again, $*$ is used here as a wildcart); and the third possible st-path

starts the same as the first option and ends as the second one. Based on this observation, notice that there can be at most 2 hyperedge-disjoint st-hyperpaths: necessarily, one of them has an underlying path of the first type, while the other one has an underlying path of the second type.

From a satisfying assignment A of I we can construct the two disjoint st-hyperpaths as follows. One hyperpath leads from s to t via the vertices $c_1, c_2, \ldots, c_{m+1}$, and to move from c_i to c_{i+1} it uses a lit-in-clause hyperedge that corresponds to a pair (l, c_i) such that l is one of the literals that satisfy the clause c_i in A. The second hyperpath has an underlying path of the second type, it leads via $x_1, x_2, \ldots, x_{n+1}$, and from x_j to x_{j+1} it uses the vertices containing only the literals that are not satisfied by the assignment A. Thus, the second hyperpath uses only those lit-in-clause hyperedges that correspond to pairs containing literals that are not satisfied by A. This implies that the two constructed st-hyperpaths are hyperedge-disjoint.

Let P and Q be two hyperedge-disjoint st-hyperpaths of $\mathcal{H}$, considering only the forward hyperedges, without the source hyperedges that lead to vertices ux_*c_* or $u\bar{x_*}c_*$. Let P have an underlying path p of the first type and Q has an underlying path q of the second type. We can construct a satisfying assignment for I by setting to `TRUE` the literals opposite to those that occur in the vertices on q. Then, the hyperpath P suggests how the clauses of I are satisfied by this assignment. □

Theorem 10. *The minimum st-hypercut problem in sequence hypergraphs with backward hyperedges is NP-hard.*

Proof. We construct a reduction from the NP-hard vertex cover problem. Let I be an instance of the vertex cover problem, given as an undirected graph $I = (U, F)$, with the vertex set $U = \{u_1, \ldots, u_n\}$, and the edge set $F = \{f_1, \ldots, f_m\}$.

To construct from I an instance I' of the minimum st-hypercut problem in acyclic sequence hypergraphs, we fix an order of the vertices and edges of I as follows. Let $U^O = (u_1, \ldots, u_n)$ be an arbitrary order on the vertices of U. From every edge in F we create an ordered edge, where the vertices of the edge are ordered naturally according to U^O. Let F' denote the set of the created ordered edges, and let F^O be the edges F' ordered lexicographically according to U^O.

We construct the sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ of I' as follows (cf. Figure 10 along with the construction). The construction is very similar to that in the proof of Theorem 4, so we highlight the changes in bold. One major change is that now for every hyperedge there is a backward hyperedge. For simplicity, we divide all the sequence hyperedges into pairs of mutually backward sequence hyperedges, and we refer to one sequence hyperedge of each pair as *forward* hyperedge (and to the other as its backward hyperedge). For simplicity of the construction, we describe explicitly only the forward hyperedges, and each of them implicitly defines a backward hyperedge.

The set of vertices $\mathcal{V}$ consists of $3m + 2 + \mathbf{2n}$ vertices: a source vertex s and a target vertex t; for each edge $f_i \in F^O$, $i = 1, \ldots, m$, there are three vertices $v_{(i,1)}, v_{(i,2)}$, and $v_{(i,3)}$; and finally, for each vertex $u_k \in U^O$, $k = 1, \ldots, n$, there

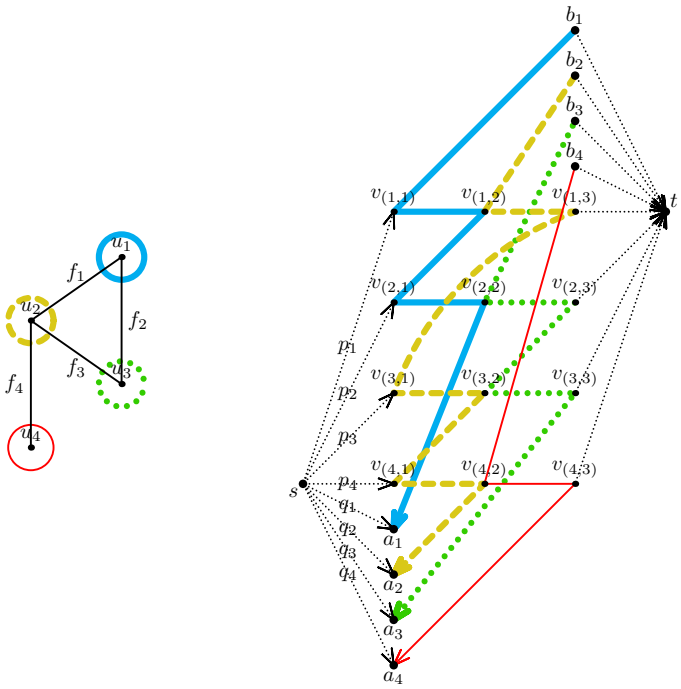

Fig. 10. Minimum st-hypercut in sequence hypergraphs with backward hyperedges is NP-hard.

are two vertices $\mathbf{a_k}$ and $\mathbf{b_k}$. The set of hyperedges $\mathcal{E}$ contains $2m + \mathbf{3}n$ forward hyperedges (plus the same amount of the corresponding backward hyperedges that we do not specify explicitly). There are m *type-1 source hyperedges* of the form $(s, v_{(i,1)})$, each of them connects s to one vertex $v_{(i,1)}$, and there are n *type-2 source hyperedges* of the form $(\mathbf{s}, \mathbf{a_k})$, connecting s to $\mathbf{a_k}$, for $k = 1, \ldots, n$. There are m *type-1 target hyperedges* of the form $(v_{(i,3)}, t)$, each of them connects one vertex $v_{(i,3)}$ to t, and there are n *type-2 hyperedges* of the form $(\mathbf{b_k}, \mathbf{t})$, connecting $\mathbf{b_k}$ to t, for $k = 1, \ldots, n$. And finally, there are n *vertex hyperedges*, each corresponds to one of the vertices in U, that are constructed as follows.

For a vertex $u_k \in U$ of the graph I, we describe the sequence of vertices in the corresponding vertex hyperedge E_{u_k} iteratively. We start with a sequence containing $E_{u_k} = \mathbf{b_k}$. We consider the edges F^O in order, and for each edge $f_i \in F^O$ that contains u_k we prolong the sequence of E_{u_k} as follows. If f_i contains u_k as the first vertex, we append $v_{(i,1)}$ and $v_{(i,2)}$ to E_{u_k}. Otherwise, f_i contains u_k as the second vertex, and we append $v_{(i,2)}$ and $v_{(i,3)}$ to E_{u_k}. Once all edges containing u_k have been considered, we append $\mathbf{a_k}$ to E_{u_k}. We denote by E'_{u_k} the corresponding backward vertex hyperedge.

Now consider the edges of the obtained hyperedge E_{u_k} and note the form they can have. The first edge is an edge $(\mathbf{b_k}, v_{(i,j)})$ for some $j \in \{1, 2\}$ and

some $i \in [m]$. The last edge is an edge $(\mathbf{v_{(i,j)}}, a_k)$ for some $j \in \{2,3\}$ and some $i \in [m]$. The remaining edges are of two main types: There are edges of the form $(v_{(i,1)}, v_{(i,2)})$ and $(v_{(i,2)}, v_{(i,3)})$ for some $i \in [m]$, and there are edges of the form $(v_{(i,2)}, v_{(j,1)})$, $(v_{(i,3)}, v_{(j,2)})$, and $(v_{(i,3)}, v_{(j,1)})$ for some $i < j$, $i, j \in [m]$. Note that, due to the fact that the edges in F^O are ordered lexicographically, all the edges of the vertex hyperedge E_{u_k} take one of the forms described above. Also note that, due to the direct correspondence between the vertices in U and vertex hyperedges, each tuple $(v_{(i,j)}, v_{(i,j+1)})$ is part of (i.e., an edge of) exactly one of the hyperedges.

Let us now observe that there always exists a minimum st-hypercut that does not contain any type-2 source hyperedges or type-2 target hyperedges. Clearly, none of the backward type-2 source hyperedges or type-2 target hyperedges are in a minimum st-hypercut, since each consists of a single edge that either leads to s or from t. Notice that in the underlying graph G of $\mathcal{H}$, for each $k = 1, \ldots, n$, there is only one outgoing edge from $\mathbf{a_k}$ (other than to s), and this edge belongs only to the backward vertex hyperedge E'_{u_k}. Similarly, there is only one incoming edge from $\mathbf{b_k}$ (other than from t) and this edge belongs only to the backward vertex hyperedge E'_{u_k}. Consequently, every forward type-2 source hyperedge or type-2 target hyperedge in an st-hypercut C can be substituted for a backward vertex hyperedge, and the resulting set is an st-hypercut of size equal to or smaller than C. Thus, there exists a minimum st-hypercut C which does not contain type-2 source or type-2 target hyperedges. Now observe that such an st-hypercut C has to contain *all* the backward vertex hyperedges, as otherwise, for some k, the underlying graph of $(\mathcal{V}, \mathcal{E} \setminus C)$ would contain an st-path using the type-2 source hyperedge from s to $\mathbf{a_k}$, the backward vertex hyperedge E'_{u_k} from $\mathbf{a_k}$ to $\mathbf{b_k}$, and the type-2 target hyperedge from $\mathbf{b_k}$ to t. Now we show that each type-1 source or type-1 target hyperedge in C can be substituted by a forward vertex hyperedge. Let B be the set of all backward vertex hyperedges. Notice that in the underlying graph of $(\mathcal{V}, \mathcal{E} \setminus B)$, the only outgoing edges from $v_{(i,1)}$ lead to s or $v_{(i,2)}$, for $i = 1, \ldots, m$; and similarly, the only incoming edges to $v_{(i,3)}$ come from t or $v_{(i,2)}$. (And clearly, the hyperedges $(v_{(i,1)}, s)$ and $(t, v_{(i,3)})$ are not part of any st-hyperpath.) Since for each i, the tuple $(v_{(i,j)}, v_{(i,j+1)})$ is an edge of exactly one hyperedge, every type-1 source hyperedge $(s, v_{(i,1)})$ in an st-hypercut C can be substituted by the forward vertex hyperedge containing the edge $(v_{(i,1)}, v_{(i,2)})$; and every type-1 target hyperedge $(v_{(i,3)}, t)$ can be substituted for the forward vertex hyperedge containing the edge $(v_{(i,2)}, v_{(i,3)})$, and the resulting set is an st-hypercut of size equal to or smaller than C. Thus, there exists an optimum solution that contains only vertex hyperedges, and in particular, it contains all the backward vertex hyperedges.

Now observe that any minimum st-hypercut C consisting of vertex hyperedges only must for each $i = 1, \ldots, m$ "hit" either the edge $(v_{(i,1)}, v_{(i,2)})$ or $(v_{(i,2)}, v_{(i,3)})$ (i.e., one of those two edges is an edge of some hyperedge in C). Otherwise, the underlying graph of $(\mathcal{V}, \mathcal{E} \setminus C)$ would contain an st-path $p_i = s, v_{(i,1)}, v_{(i,2)}, v_{(i,3)}, t$.

Let $\mathcal{S}' \subseteq \mathcal{H}$ be an optimum solution to the instance I' that contains only vertex hyperedges. Since $\mathcal{S}'$ contains all n backward vertex hyperedges, it contains $|\mathcal{S}'| - n$ forward vertex hyperedges. Recall that there is a direct one-to-one mapping between the forward vertex hyperedges and the vertices U of the instance I. There is also a direct mapping between each triple $(v_{(i,1)}, v_{(i,2)}, v_{(i,3)})$ and an edge from F. Since the solution $\mathcal{S}'$ hits one of the edges $(v_{(i,1)}, v_{(i,2)})$ or $(v_{(i,2)}, v_{(i,3)})$ for each i, we can use the mapping to construct a solution $\mathcal{S} \subseteq U$ to the instance I of the original minimum vertex cover problem, such that $|\mathcal{S}| = |\mathcal{S}'| - n$. On the other hand, every solution of size k to the original vertex cover instance can be mapped, using the same direct one-to-one mapping in the opposite direction, to an st-hypercut of $\mathcal{H}$ of size $k + n$. Therefore, an optimum solution for I' gives us an optimum solution for I. □

Theorem 11. *The minimum st-hypercut problem in sequence hypergraphs with backward hyperedges can be 2-approximated.*

Proof. Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a sequence hypergraph with backward hyperedges, and let s and t be two vertices of $\mathcal{H}$. Note that we can partition $\mathcal{E}$ into $|\mathcal{E}|/2$ pairs of hyperedges (each pair contains a hyperedge and its backward hyperedge). We construct a (standard) hypergraph $\mathcal{H}^* = (\mathcal{V}^* = \mathcal{V}, \mathcal{E}^*)$ from $\mathcal{H}$ in such a way that for each pair of mutually backward sequence hyperedges E and E' of $\mathcal{E}$, $\mathcal{E}^*$ contains exactly one (non-oriented) hyperedge E^* that corresponds to the set of vertices of E (and thus also E'). Note that $\mathcal{E}^*$ may contain multiple hyperedges corresponding to the same set of vertices, and we get that $\mathcal{E}^*$ contains exactly $|\mathcal{E}|/2$ hyperedges.

Next, we find a minimum st-hypercut X^* in $\mathcal{H}^*$ (this can be done in polynomial time by a transformation of $\mathcal{H}^*$ into a directed graph and solving a maximum flow problem in it). Clearly, $|X^*|$ is a lower bound on the size of a minimum st-hypercut in $\mathcal{H}$. Recall that every hyperedge in X^* corresponds to 2 sequence hyperedges in $\mathcal{H}$, thus, by removing all sequence hyperedges corresponding to hyperedges in X^*, we obtain an st-hypercut in $\mathcal{H}$ of size $2\,|X^*|$. □

8 On Other Algorithmic Problems

We briefly consider some other standard graph algorithmic problems in sequence hypergraphs.

Definition 7 (rooted spanning hypergraph). *Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a sequence hypergraph. An* s-rooted spanning hypergraph *T is a subset of $\mathcal{E}$ such that for every $v \in \mathcal{V}$, T is an sv-hyperpath. The* size *of T is $|T|$.*

Theorem 12. *Finding a minimum s-rooted spanning hypergraph in acyclic sequence hypergraphs is NP-hard to approximate within a factor of $(1 - \varepsilon) \ln n$, unless $P = NP$.*

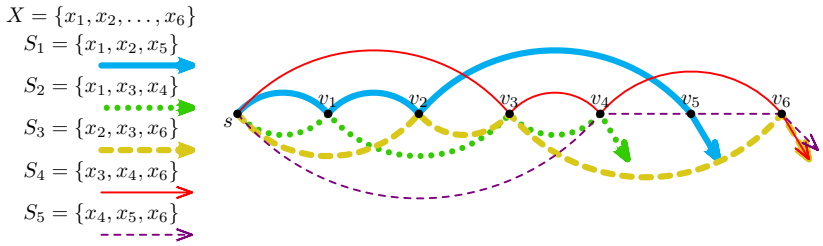

Fig. 11. Minimum s-rooted spanning hypergraph is at least as hard as the minimum set cover problem.

Proof. We construct an approximation-preserving reduction from the set cover problem, which has the claimed inapproximability [10]. Let $I = (X, \mathcal{S})$ be an instance of the set cover problem, given by a ground set $X = \{x_1, \ldots, x_n\}$, and a family of its subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$.

We construct from I an instance I' of the minimum s-rooted spanning hypergraph problem in sequence hypergraphs (see Figure 11 for an example). Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be the following sequence hypergraph. The set of vertices $\mathcal{V} = \{s, v_1, \ldots, v_n\}$ contains one vertex v_i for each $x_i \in X$, and a source vertex s. Let V^O be an arbitrary fixed order on the vertices $\mathcal{V}$. There are m hyperedges in $\mathcal{E} = \{E_1, \ldots, E_m\}$, one for each set in $\mathcal{S}$. For a set $S_i \in \mathcal{S}$, let us take the vertices that correspond to the elements in S_i and order them according to V^O to obtain $(v_{i_1}, v_{i_2}, \ldots, v_{i_r})$. Then the sequence of the hyperedge E_i corresponding to S_i is simply $(s, v_{i_1}, v_{i_2}, \ldots, v_{i_r})$. The constructed sequence hypergraph is acyclic, as all the edges of its hyperedges agree with V^O.

It remains to show that from a minimum s-rooted spanning hypergraph T for I' a minimum set cover for I of the same size can be constructed. By definition, T is the smallest subset of $\mathcal{E}$ that allows to reach from s any other vertex in $\mathcal{V}$. Moreover, since $\mathcal{H}$ is acyclic and each hyperedge starts at s, for each vertex $v \in \mathcal{V}$, it is enough to choose one of the hyperedges in $\mathcal{E}$ (containing v) to reach v from s. This, together with the one-to-one correspondence between the hyperedges in $\mathcal{E}$ and sets in $\mathcal{S}$, provides us with a prescription for a minimum set cover for I of size $|T|$. On the other hand, we can map any solution of the original minimum set cover instance to a minimum s-rooted spanning hypergraph of the same size. □

Definition 8 (strongly connected component). *Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a sequence hypergraph. We say that a set $C \subseteq \mathcal{E}$ forms a* strongly connected component *if for every two vertices $u, v \in \mathcal{V}'$, with $\mathcal{V}'$ being all the vertices of $\mathcal{V}$ present in C, the set C contains a uv-hyperpath. We say that the vertices in $\mathcal{V}'$ are* covered *by C.*

Clearly, we can decide in polynomial time whether a given set of hyperedges C forms a strongly connected component as follows. Consider the underlying graph G of $\mathcal{H}$ induced by the set of sequence hyperedges C and find a maximum

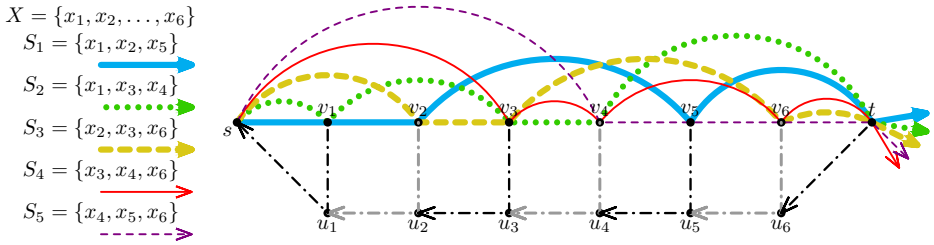

Fig. 12. For a given sequence hypergraph, it is NP-hard to find a minimum non-empty set of sequence hyperedges that forms a strongly connected component.

strongly connected component there. If this component spans the whole G, then C is a strongly connected component in $\mathcal{H}$.

Theorem 13. *Given a sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, it is NP-hard to find a* minimum *number of hyperedges that form a strongly connected component C so that a) C is any non-empty set, or b)* all *the vertices in $\mathcal{V}$ are covered by C.*

Proof. Variant b) is clearly NP-hard by a reduction from the Hamiltonian cycle problem: Consider a standard directed graph $G = (V, E)$ and view it as a sequence hypergraph, where each sequence hyperedge is just an edge of G. Then, by finding a strongly connected component that covers all the vertices of V and uses the minimum number of sequence hyperedges (i.e., normal edges) we can solve the Hamiltonian cycle problem in G (either the component consists of $|V|$ edges or more).

We show NP-hardness of variant a) by a reduction from the APX-hard set cover problem [10]. Let $I = (X, \mathcal{S})$ be an instance of the set cover problem, given by a ground set $X = \{x_1, \ldots, x_n\}$, and a family of its subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$.

From I we construct a sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ as follows (see Figure 12 for an example). The set of vertices $\mathcal{V} = \{s, t, v_1 \ldots, v_n, u_1 \ldots, u_n\}$ contains one vertex v_i and one vertex u_i for each $x_i \in X$, a source vertex s, and a target vertex t. Let X^O be an arbitrary fixed order on the elements of X. The set of sequence hyperedges $\mathcal{E}$ consists of $m + n + 1$ hyperedges. There are n *element hyperedges*: for each element x_i in X (i-th element of X in the order X^O), there is a hyperedge (v_i, u_i, u_{i-1}), the hyperedge (v_1, u_1, s) corresponding to x_1 ends at s instead. There is one *target hyperedge* (t, u_n). Finally, there are m *set hyperedges*, one for each set in $\mathcal{S}$ defined as follows. For every set $S_i \in \mathcal{S}$, we take the vertices from $\{v_1, \ldots, v_n\}$ that correspond to the elements in S_i and order them according to X^O to obtain $(v_{i_1}, v_{i_2}, \ldots, v_{i_r})$. Then the sequence of the hyperedge E_i corresponding to S_i is simply $(s, v_{i_1}, v_{i_2}, \ldots, v_{i_r}, t)$.

It remains to show that a minimum set cover for I can be constructed from a minimum non-empty set of sequence hyperedges of $\mathcal{H}$ that form a strongly connected component. Let O be such a non-empty set of hyperedges. First observe that if O contains any set hyperedge, then O must contain *all* the element hyperedges and the target hyperedge. This follows from the fact that if O con-

tains a set hyperedge, then s and t are covered by O, and O must contain a ts-hyperpath, but the only ts-hyperpath clearly consists of all element hyperedges plus the target hyperedge. Clearly, if O contains an element hyperedge or a target hyperedge, then either some v_i or t is covered by O, and thus O must contain a hyperpath that leads to v_i or t, respectively. However, the only sequence hyperedges that can reach v_i or t are the set hyperedges. Therefore, O always contains at least one set hyperedge and all the element hyperedges and the target hyperedge. However, this implies that *all* the vertices have to be covered by O, which, in particular, implies that O must contain an sv_i-hyperpath for each v_i. Therefore, O consists of all element hyperedges, the target hyperedge, and the smallest subset of the set hyperedges that allows to reach from s any vertex $v_i \in \mathcal{V}$. In other words, for each vertex $v_i \in \mathcal{V}$, O must contain a set hyperedge that can reach v_i from s. This, together with the one-to-one correspondence between the set hyperedges in $\mathcal{E}$ and sets in $\mathcal{S}$, provides us with a prescription for a minimum set cover for I of size $|O| - n - 1$. On the other hand, using the same mapping in the opposite direction, we can map any solution of the original minimum set cover instance to a minimum non-empty set of sequence hyperedges of $\mathcal{H}$ that form a strongly connected component. □

Theorem 14. *Given a sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, finding a* maximum *number of hyperedges that form a strongly connected component C so that a) C is any non-empty set, or b)* all *the vertices in $\mathcal{V}$ are covered by C, is polynomial-time solvable.*

Proof. As discussed above, for a given set of hyperedges, we can decide in polynomial time whether it forms a strongly connected component or not. Thus, in particular, we can check whether the set $\mathcal{E}$ forms a strongly connected component. In case it does, variant b) is solved trivially by taking *all* the hyperedges in $\mathcal{E}$. Otherwise, this variant has no solution, because there is no strongly connected component covering all vertices in $\mathcal{V}$.

To solve variant a), we start with te set of all hyperedges $\mathcal{E}$ and during the process, we iteratively remove those hyperedges that cannot be part of a feasible solution. Iteratively, we find strongly connected components in the underlying graph induced by the current set of hyperedges and remove all the sequence hyperedges that occur in two or more components (as these cannot be a part of a single strongly connected component), and repeat. Once no more hyperedges can be removed, we reached a situation, where each of the remaining hyperedges is in exactly one strongly connected component. Thus, the solution is defined by the component that contains the maximum number of hyperedges. □

References

1. Ausiello, G., Franciosa, P.G., Frigioni, D.: Directed hypergraphs: Problems, algorithmic results, and a novel decremental approach. In: Theoretical Computer Science, pp. 312–328. Springer (2001)
2. Ausiello, G., Giaccio, R., Italiano, G.F., Nanni, U.: Optimal traversal of directed hypergraphs. Tech. rep. (1992)

3. Berry, J., Dean, N., Goldberg, M., Shannon, G., Skiena, S.: Graph drawing and manipulation with LINK. In: GD 1997. pp. 425–437. Springer (1997)
4. Böhmová, K., Mihalák, M., Pröger, T., Sacomoto, G., Sagot, M.F.: Computing and Listing *st*-Paths in Public Transportation Networks. In: CSR 2016. vol. 9691, pp. 102–116. Springer (2016)
5. Broersma, H., Li, X., Woeginger, G., Zhang, S.: Paths and cycles in colored graphs. Australasian journal of combinatorics 31, 299–311 (2005)
6. Carr, R.D., Doddi, S., Konjevod, G., Marathe, M.V.: On the red-blue set cover problem. In: SODA 2000. vol. 9, pp. 345–353. Citeseer (2000)
7. Coudert, D., Datta, P., Pérennes, S., Rivano, H., Voge, M.E.: Shared risk resource group complexity and approximability issues. Parallel Processing Letters 17(02), 169–184 (2007)
8. Coudert, D., Pérennes, S., Rivano, H., Voge, M.E.: Combinatorial optimization in networks with Shared Risk Link Groups. Research report, INRIA (Oct 2015)
9. Crescenzi, P., Kann, V., Halldórsson, M., Karpinski, M., Woeginger, G.: A compendium of NP optimization problems. URL: http://www. nada. kth. se/˜ viggo/problemlist/compendium. html (1997)
10. Dinur, I., Steurer, D.: Analytical Approach to Parallel Repetition. In: STOC 2014. pp. 624–633. ACM, New York, NY, USA (2014)
11. Elias, P., Feinstein, A., Shannon, C.E.: A note on the maximum flow through a network. Information Theory, IRE Transactions on 2(4), 117–119 (1956)
12. Erdős, P.L., Frankl, P., Katona, G.O.: Intersecting Sperner families and their convex hulls. Combinatorica 4(1), 21–34 (1984)
13. Fellows, M.R., Guo, J., Kanj, I.A.: The parameterized complexity of some minimum label problems. In: WG 2009. pp. 88–99. Springer (2009)
14. Gallo, G., Longo, G., Pallottino, S., Nguyen, S.: Directed hypergraphs and applications. Discrete applied mathematics 42(2), 177–201 (1993)
15. Garey, M.R., Johnson, D.S.: Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman & Co., New York, NY, USA (1979)
16. Goldberg, P.W., McCabe, A.: Shortest Paths with Bundles and Non-additive Weights Is Hard. In: CIAC 2013. vol. 7878, pp. 264–275. Springer (2013)
17. Hassin, R., Monnot, J., Segev, D.: Approximation algorithms and hardness results for labeled connectivity problems. Journal Combinatorial Optimization 14(4), 437–453 (2007)
18. Hu, J.Q.: Diverse routing in optical mesh networks. Communications, IEEE Transactions on 51(3), 489–494 (2003)
19. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within 2-epsilon. Journal of Computer and System Sciences 74(3), 335–349 (2008)
20. Krumke, S.O., Wirth, H.C.: On the minimum label spanning tree problem. Information Processing Letters 66(2), 81–85 (1998)
21. Wachman, G., Khardon, R.: Learning from interpretations: a rooted kernel for ordered hypergraphs. In: Proceedings of the 24th international conference on Machine learning. pp. 943–950. ACM (2007)
22. Yuan, S., Varma, S., Jue, J.P.: Minimum-color path problems for reliability in mesh networks. In: INFOCOM 2005. vol. 4, pp. 2658–2669. IEEE (2005)
23. Zhang, P., Cai, J.Y., Tang, L.Q., Zhao, W.B.: Approximation and hardness results for label cut and related problems. Journal Combinatorial Optimization 21(2), 192–208 (2011)