Sequence Hypergraphs

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

[1] Institute of Theoretical Computer Science, ETH Zürich, Zürich, Switzerland
{katerina.boehmova,widmayer}@inf.ethz.ch
[2] LIF, CNRS & Aix-Marseille University, Marseille, France
jeremie.chalopin@lif.univ-mrs.fr
[3] Department of Knowledge Engineering, Maastricht University, 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 (imagine it as a directed path). Note that this differs substantially from the standard definition of directed hypergraphs.
Sequence hypergraphs are motivated by problems in public transportation networks, as they conveniently represent transportation lines. We study the complexity of some classic 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, 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 the reverse order. Finally, we briefly discuss other algorithmic problems (e.g., finding a minimum spanning tree, or connected components).

Keywords: Colored Graphs, Labeled Problems, Oriented Hypergraphs, Algorithms, Complexity

1 Introduction

Consider a public transportation network, e.g. a bus network, where each bus line is specified as a fixed sequence of stops. Clearly, one can travel in the network by taking a bus and following the stops in the order fixed by the corresponding line. 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 kind of questions are traditionally modeled by algorithmic graph theory, but we lacked a model that would capture all the necessary aspects of the problems formulated as above. We propose the following non-standard, but a 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 $E = (v_{i_1}, v_{i_2}, \ldots, v_{i_l})$ 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, 13], where each directed hyperedge is a pair $(\mathrm{From}, \mathrm{To})$ of disjoint subsets of $\mathcal{V}$.[5] We note that the order of vertices in a sequence hyperedge does not imply any order of the vertices of other hyperedges. Furthermore, the sequence hypergraphs do 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 $E_i = (v_1, v_2, \ldots, v_l)$ from $\mathcal{E}$, the multigraph G contains $l - 1$ edges (v_j, v_{j+1}) for $j = 1, \ldots, l - 1$, all colored with color $c(E_i)$. Therefore, each edge of G is colored by one of the $k = |\mathcal{E}|$ 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 $\mathcal{H}$.

In this paper, we study some 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; finding a *minimum st-hypercut*: an st-cut that uses the minimum number of sequence hyperedges; or finding a *maximum st-hyperflow*: a maximum number of hyperedge-disjoint st-hyperpaths. We note that the shortest st-hyperpath problem was already considered by Böhmová et al. [5] in the setting of finding good routes in public transportation networks (studied under a quite different terminology), who mainly focused on the problem of listing shortest paths in public transportation networks, but also showed that minimizing the number of lines in an st-path is hard to approximate.

In the present paper we show that the shortest st-hyperpath can be found in polynomial time if the given sequence hypergraph is acyclic. 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. We then consider sequence hypergraphs with sequence hyperedges of constant length (defined as the number of vertices minus one). We note that the shortest st-hyperpath problem remains

[5] To avoid confusion with directed hypergraphs, we prefer the term *sequence hypergraphs* to refer to the hypergraphs with hyperedges formed as sequences of vertices.

Table 1. Summary of the complexity of 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 indicate our contribution.

	Colored/Labeled Graphs		Sequence Hypergraphs			
	General	Span 1	General	Acyclic	Backward	Length≤ 2
Shortest *st*-path	APX-hard [8, 17]	P [8]	APX-hard [5]	P	P	P
Minimum *st*-cut	APX-hard [8, 23]	P [8]	APX-hard	APX-hard	NP-hard	P
Maximum *st*-flow	APX-hard [18]	P [8]	APX-hard	APX-hard	NP-hard	P
MaxFlow-MinCut Duality	$\times$ [8]	$\surd$ [8]	$\times$	$\times$	$\times$	$\surd$

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. 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. Finally, we briefly study the complexity of other algorithmic problems (finding minimum spanning tree, or connected components) in sequence hypergraphs. For a summary of the results see Table 1. The table also shows known results for related labeled graphs (discussed below).

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 simultaneously and a goal is to find resilient solutions. Similar situation 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. Some of the studies consider a slightly different definition of a colored graphs, where to each edge corresponds a set of colors instead of a single color. Since the computational complexity problems may differ in the two models, the transformations between the two models have been investigated [9].

The *minimum label path* problem, which asks for an *st*-path of a minimum number of colors, is NP-hard and hard to approximate [6–8, 15, 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 of minimum number of colors that forms an st-cut, is NP-hard and hard to approximate [8, 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. Zhang et al. [23] give a $\sqrt{m}$-approximation algorithm for the minimum label cut problem. Fellows et al. study the parameterized complexity of minimum label problems [12]. Coudert et al. [8, 9] 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* label 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, this type of hypergraphs are usually not explored 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. [4] 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. Erdös et al. [11] study Sperner-families and as an application of a derived result they study the maximum number of edges of a so called *directed Sperner-hypergraph*.

Finally, a special case of sequence hypergraphs arose as a generalization to tournaments [3, 16]: A k-hypertournament can be seen as a sequence hypergraph where for every subset of k vertices there is exactly one sequence hyperedge. Gutin et al. [16] studied the Hamiltonicity of k-hypertournaments.

2 On the Shortest *st*-Hyperpath

In this section, we briefly discuss the complexity of 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 sequence 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 an underlying st-path of P.*

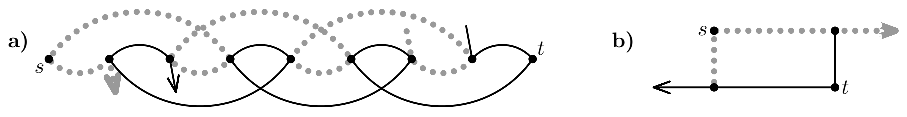

Fig. 1. In both figures, the grey-dotted curve, and the black curve depict two sequence hyperedges. **a)** The length of the st-hyperpath is 2, but the number of switches is 7. **b)** The st-hyperpath consists of two sequence hyperedges that also form a hypercycle.

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 1a).

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.*

This can be done, 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).

On the other hand, by a reduction from the set cover problem, Böhmová et al. [5] showed the following result (in a slightly different setting).

Theorem 1 ([5]). *Shortest st-hyperpath in sequence hypergraphs is NP-hard to approximate within a factor of $(1-\epsilon)\ln n$, unless $P = NP$.*

However, if the given sequence hypergraph is acyclic, we show that the shortest st-hyperpath can be found in polynomial time.

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 forms both a hyperpath from a to b, and a hyperpath from b to a. A sequence hypergraph without hypercycles is called* acyclic*.*

Observe that an st-hyperpath may also be a hypercycle (see Figure 1b).

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 $\{e_i = (v_i, v_{i+1})$ for $i = 1, \ldots, k-1\}$ 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.*

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

Proof. (Sketch.) Observe that for every st-hyperpath, there is an underlying path where all the edges of each hyperedge appear consecutively. Thus, 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. □

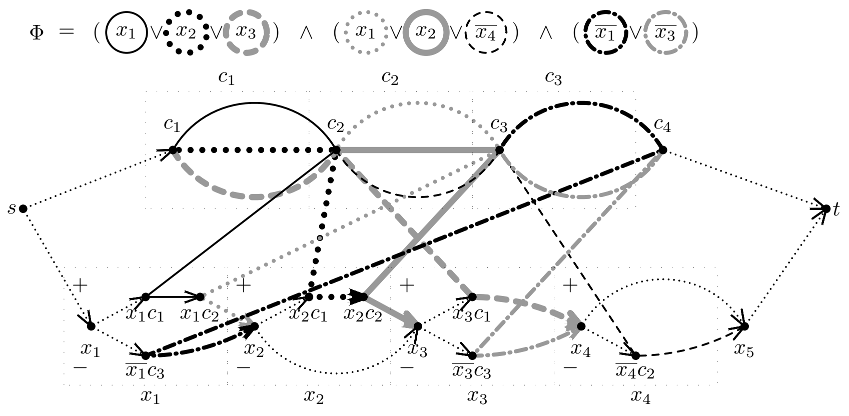

Fig. 2. Deciding st-hyperflow of size 2 is as hard as 3-Sat.

3 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 the 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 $2 - \epsilon$ 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-complete to decide whether there are two hyperedge-disjoint st-hyperpaths.*

Proof. We construct a reduction from the NP-complete 3-Sat problem [14]. 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. The goal 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 2 along with the construction). The set of vertices $\mathcal{V}$ 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_jc_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 of hyperedges $\mathcal{E}$ 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) 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 $(c_i, c_{i+1}, x_j c_i, x_j c_k)$, 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 $\overline{x_2} c_i$; $\ldots$; 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 the sequence hypergraph $\mathcal{H}$ contains two hyperedge-disjoint st-hyperpaths. There are 3 possible types of st-paths in the underlying graph of $\mathcal{H}$: first one leads through all the vertices $c_1, c_2, \ldots, c_{m+1}$ in this order; second one leads through all the vertices $x_1, x_2, \ldots, x_{m+1}$ in this order and between x_j, x_{j+1} it goes either through all the $x_j c_*$ vertices or through all the $\overline{x_j} c_*$ vertices; 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. 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 corresponds 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 has 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 `FALSE` the literals that occur in the vertices on q. Then, the hyperpath P suggests how the clauses of I are satisfied by this assignment. $\square$

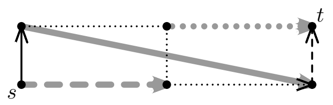

Fig. 3. Acyclic sequence hypergraph with minimum st-hypercut of size 2, and no two hyperedge-disjoint st-hyperpaths.

4 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 $|X|$, that is the number of hyperedges in X.*

For directed (multi)graphs, the famous MaxFlow-MinCut Duality Theorem [10] 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, even in acyclic sequence hypergraphs as Figure 3 shows. But, of course, the size of an st-hyperflow is a lower bound on the size of an st-hypercut. We showed maximum st-hyperflow to be APX-hard even in acyclic sequence hypergraphs. It turns out that also minimum st-hypercut problem in acyclic sequence hypergraphs is APX-hard.

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

Proof. We construct an approximation preserving reduction from the vertex cover problem, which has the claimed inapproximability [19]. □

5 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, the shortest st-hyperpath problem is APX-hard even if all the hyperedges are of length at most 5 (the proof given in [5] needs just a slight modification); Figure 2 illustrates that the duality between minimum st-hypercut and maximum st-hyperflow breaks already with a single hyperedge of length 3; and Theorem 3 holds even if all hyperedges are of length at most 3.

It is an interesting question to investigate the 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.*

The proof is based on similar ideas as in the proof of Theorem 2.

Theorem 6. *The maximum st-hyperflow problem and the minimum st-hypercut problem can be solved in polynomial time in sequence hypergraphs with hyperedges of length at most 2. The size of the maximum st-hyperflow then equals the size of the minimum st-hypercut.*

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 a 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 cycles 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, 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 a hyperedge appear on any st-path consecutively, and any hyperedge that was present on only one of p_i, 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 than one paths, and after at most $|\mathcal{E}|$ iterations of this transformation we obtain $|f|$ hyperedge-disjoint st-paths, which gives us an st-hyperflow F of size $|F| = |f|$. 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 obtained a maximum st-hyperflow in $\mathcal{H}$.

Since in directed multigraphs the size of the minimum cut equals the size of the maximum flow [10], it follows that we can find $|F|$ edges $e_1, \ldots, e_{|F|}$ of G that forms 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 any st-hypercut is bounded from below by the size of the hyperflow, C is a minimum st-hypercut of size $|C| = |F|$. □

6 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 $E = (v_1, v_2, \dots, v_k)$ be a hyperedge of a sequence hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$. We say that E' is a* backward hyperedge[6] *of E, if $E' = (v_k, \dots, v_2, v_1)$. 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 arise naturally in urban public transportation networks, for instance most of the tram lines have also a "backward" line (which has the exact same stops as the "forward" line, but goes in the opposite order). We study the complexity 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 for the minimum st-hypercut. The positive results are based on existing algorithms for standard hypergraphs, the negative results are obtained by a modification of the hardness proofs in Sections 3 and 4.

Theorem 7. *The shortest st-hyperpath problem in sequence hypergraphs with backward hyperedges is in P.*

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[7] 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 $(s = v_1, v_2, \dots, v_{|P^*|+1} = t)$ be a sequence of vertices, subsequence of p^*, such that for each $i = 1, \dots, |P^*|$, there is a hyperedge E^* in P^* that contains both v_i and v_{i+1}, and 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 $(E_1, E_2, \dots, E_{|P^*|})$, $E_i \in \mathcal{E}$, such that v_i, v_{i+1} are vertices in E_i. Since $\mathcal{H}$ is sequence hypergraph with backward hyperedges, for every hyperedge E of $\mathcal{E}$ and a pair its of vertices v_i, v_{i+1} of E, there is an $v_i v_{i+1}$-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 8. *The maximum st-hyperflow problem in sequence hypergraphs with backward hyperedges is NP-hard.*

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

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

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

Theorem 10. *The minimum st-hypercut problem in sequence hypergraphs with backward hyperedges can be 2-approximated.*

7 On Other Algorithmic Problems

We briefly consider some other standard graph algorithmic problems.

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

Theorem 11. *Minimum s-rooted spanning hypergraph in acyclic sequence hypergraphs is NP-hard to approximate within a factor of $(1 - \epsilon)\ln n$, unless $P = NP$.*

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 $u, v \in \mathcal{V}'$, $\mathcal{V}'$ being all the vertices of $\mathcal{V}$ present in C, the set C is a uv-hyperpath. We say that the vertices in $\mathcal{V}'$ are* covered *by C.*

Clearly, we can decide in polynomial time whether the 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 strongly connected component there. If this component spans the whole G, then C is a strongly connected component in $\mathcal{H}$.

Theorem 12. *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.*

Theorem 13. *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.*

Acknowledgements. Kateřina Böhmová is supported by a Google Europe Fellowship in Optimization Algorithms. The project has been partially supported by the Swiss National Science Foundation (SNF) under the grant number 200021_156620.
TODO

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 Berlin Heidelberg (2001)

2. Ausiello, G., Giaccio, R., Italiano, G.F., Nanni, U.: Optimal traversal of directed hypergraphs. Tech. rep. (1992)
3. Barbut, E., Bialostocki, A.: A generalization of rotational tournaments. Discrete mathematics 76(2), 81–87 (1989)
4. Berry, J., Dean, N., Goldberg, M., Shannon, G., Skiena, S.: Graph drawing and manipulation with link. In: DiBattista, G. (ed.) GD 1997. pp. 425–437. Springer Berlin Heidelberg (1997)
5. Böhmová, K., Mihalák, M., Pröger, T., Sacomoto, G., Sagot, M.F.: Computing and listing st-paths in public transportation networks. In: Kulikov, S.A., Woeginger, J.G. (eds.) CSR 2016. pp. 102–116. Springer International Publishing (2016)
6. Broersma, H., Li, X., Woeginger, G., Zhang, S.: Paths and cycles in colored graphs. Australasian journal of combinatorics 31, 299–311 (2005)
7. 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)
8. 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)
9. Coudert, D., Pérennes, S., Rivano, H., Voge, M.E.: Combinatorial optimization in networks with Shared Risk Link Groups. Research report, INRIA (Oct 2015)
10. 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)
11. Erdős, P.L., Frankl, P., Katona, G.O.: Intersecting sperner families and their convex hulls. Combinatorica 4(1), 21–34 (1984)
12. Fellows, M.R., Guo, J., Kanj, I.A.: The parameterized complexity of some minimum label problems. In: Paul, C., Habib, M. (eds.) WG 2009. pp. 88–99. Springer Berlin Heidelberg (2010)
13. Gallo, G., Longo, G., Pallottino, S., Nguyen, S.: Directed hypergraphs and applications. Discrete applied mathematics 42(2), 177–201 (1993)
14. 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)
15. Goldberg, P.W., McCabe, A.: Shortest paths with bundles and non-additive weights is hard. In: Spirakis, P.G., Serna, M. (eds.) CIAC 2013. vol. 7878, pp. 264–275. Springer Berlin Heidelberg (2013)
16. Gutin, G., Yeo, A.: Hamiltonian paths and cycles in hypertournaments. Journal of Graph Theory 25(4), 277–286 (1997)
17. Hassin, R., Monnot, J., Segev, D.: Approximation algorithms and hardness results for labeled connectivity problems. J. Comb. Opt. 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. J. Comb. Opt. 21(2), 192–208 (2011)