Spanner Enumeration for Temporal Graphs

Kazuhiro Kurita
Nagoya University, Nagoya, Japan

Andrea Marino
Dipartimento di Statistica, Informatica, Applicazioni, Università degli Studi di Firenze, Italy

Jason Schoeters
Dipartimento di Statistica, Informatica, Applicazioni, Università degli Studi di Firenze, Italy

Takeaki Uno
National Institute of Informatics, Tokyo, Japan

Abstract

A spanner of a temporal graph is a subset of edges that preserves connectivity over time between vertices. A minimal spanner is one in which no additional edges can be removed without breaking this connectivity. Our focus is on enumerating minimal spanners for a given temporal graph. We explore several variations of this problem based on the type of connectivity that must be maintained, ranging from one-to-all connectivity to one-to-all-to-one, many-to-all, and finally all-to-all connectivity. We establish that these problems become progressively harder: *(i)* We present a polynomial-delay enumeration algorithm for one-to-all connectivity; *(ii)* We prove Dual-hardness for both one-to-all-to-one and many-to-all connectivity, even in the restricted case of two-to-all; *(iii)* Finally, for all-to-all connectivity, we show that enumeration cannot be performed in output-polynomial time unless P = NP.

2012 ACM Subject Classification Mathematics of computing → Graph enumeration; Theory of computation → Sparsification and spanners; Mathematics of computing → Graph algorithms; Theory of computation → Problems, reductions and completeness; Networks → Network dynamics

Keywords and phrases temporal graphs, temporal spanners, one-to-all connectivity, all-to-all connectivity enumeration, NP-completeness, Dual-hardness, binary partition tree, flashlight search, polynomial delay

Digital Object Identifier 10.4230/LIPIcs.SAND.2025.9

Funding *Kazuhiro Kurita*: JSPS Kakenhi Grant Numbers JP21K17812, JP23K24806, and JP25K21273, and JST ACT-X Grant Number JPMJAX2105.
Andrea Marino: Italian PNRR CN4 Centro Nazionale per la Mobilità Sostenibile, NextGeneration EU – CUP, B13C22001000001. MUR of Italy, under PRIN Project n. 2022ME9Z78 – NextGRAAL: Next-generation algorithms for constrained GRAph visuALization, PRIN PNRR Project n. P2022NZPJA – DLT-FRUIT: A user centered framework for facilitating DLTs FRUITion.
Jason Schoeters: PRIN PNRR Project n. P2022NZPJA – DLT-FRUIT: A user centered framework for facilitating DLTs FRUITion.

Acknowledgements We would like to thank Lapo Cioni for the joining in the verification of constructions and proofs during coffee breaks.

1 Introduction

Spanning trees provide an efficient way to maintain connectivity within a network. They are minimal in the sense that removing any edge would break connectivity, and they are also minimum in terms of edge count. Efficient computation of a spanning tree is possible using greedy algorithms such as Prim's, Kruskal's, and Dijkstra's algorithms (see [33, 30, 20]).

A more general structure is a spanner, a subgraph that preserves connectivity while enforcing specific bounds on distance distortion, such as multiplicative or additive stretch factors. Unlike spanning trees, spanners are not necessarily trees. For a recent survey on graph spanners, see [1].

Temporal Spanners. In a seminal paper, Kempe, Kleinberg, and Kumar [24] introduced the study of spanning substructures in temporal graphs, networks that evolve over time. In such graphs, connectivity is established through temporal paths, which are paths where edge labels are strictly increasing. Unlike traditional spanning trees, these substructures do not necessarily form a tree and have been termed temporal spanners by the research community. A minimal spanner is one in which no further edges can be removed without breaking connectivity. For simplicity, all spanners discussed in this paper are assumed to be minimal unless stated otherwise. Building on prior research in graph spanners, as well as insights from broadcasting and gossip theory (see survey [23]), numerous studies have since explored temporal spanners. This includes results on (in)approximability [2], counterexamples or existence of sparse spanners [3, 16, 18], adding stretch, blackout-resilience, or underlying structure [6, 5, 14], to sharp thresholds in temporal Erdös–Rényi random graphs [17].
In this paper, we formalise a temporal graph $\mathcal{G}$ as a graph G with an edge time function λ, associating to each edge of G a non-empty set of discrete times (see Section 2 for full definitions). We consider the following types of connectivity for spanners:

▶ **Definition 1.** *Given a temporal graph $\mathcal{G} = (G, \lambda)$, $E' \subseteq E(G)$ is an X spanner of $\mathcal{G}$ if it is a minimal set*[1] *such that in $\mathcal{G}[E']$*[2]*:*
if X =** ONE2ALL**: given $s \in V(G)$, s reaches v for all $v \in V$;
if X =** ONE2ALL2ONE**: given $s \in V(G)$, s reaches v and v reaches s for all $v \in V$;
if X =** MANY2ALL**: Given $s_1, \ldots, s_k$, s_i reaches v for all $v \in V$, and all $i \in [k]$;
if X =** ALL2ALL**: u reaches v and v reaches u for all $u, v \in V$.

Most related work on temporal spanners focuses on ALL2ALL spanners. However, studies also exist on temporal structures with different types of connectivity, though they are not explicitly termed "spanners". For example, the paths resulting from the algorithms in [35] and the temporal branchings in [11] both correspond to specific cases of ONE2ALL spanners. There are also temporal spanners which are defined as subsets of time edges or labels (and so minimality is defined on time edges as well) instead of on edges. Some other works, such as [16, 17], consider the special case of so-called "simple" temporal graphs, where each edge only has one assigned time, and thus the distinction disappears. We do not consider such spanners in this paper.

Problem Definition. In this paper we study the problem of enumerating, i.e. listing exactly once, all the temporal spanners in Definition 1 for a given temporal graph in input. In the field of enumeration algorithms, significant research has focused on efficiently enumerating structures maintaining connectivity in static graphs, as discussed later. However, no prior work has tackled the following enumeration problem, where X refers to ONE2ALL, ONE2ALL2ONE, MANY2ALL, and ALL2ALL.

▶ **Problem 1** (ENUMERATION)**.** *Given temporal graph $\mathcal{G}$ and connectivity X, enumerate all X spanners of $\mathcal{G}$.*

[1] No proper subset satisfies the same property
[2] In Section 2, we define temporal graph $\mathcal{G}[E']$ to be temporal graph $\mathcal{G}$ restricted to the edge set $E' \subseteq E(\mathcal{G})$.

With the aim of obtaining our results on the enumeration problems, we also study the following corresponding extension problems.

▶ **Problem 2** (Extension Problem). *Given temporal graph $\mathcal{G}$, connectivity X, and edge set $E'' \subseteq E(G)$, decide whether there is $E' \supseteq E''$ that is a X spanner of $\mathcal{G}$.*[3]

▶ **Problem 3** (Connected Extension Problem). *Given temporal graph $\mathcal{G}$, connectivity X, and edge set $E'' \subseteq E(G)$ s.t. $G[E'']$ is connected, decide whether there is $E' \supseteq E''$ that is a X spanner of $\mathcal{G}$.*[3]

Indeed, being able to solve efficiently the (Connected) Extension Problem implies being able to solve efficiently the corresponding enumeration problem [32].

Some background on Enumeration Algorithms. In order to classify the complexity of enumeration problems, the classes we will use in this paper are the standard ones: *output-polynomial time*, where all output can be enumerated in time polynomial in the input and the output size; *incremental polynomial time*, where, for all i, the ith output can be produced in polynomial time in the input size and in the number i; and *polynomial delay*, where the delay between two consecutive outputs is polynomial in the input size. Additionally, a famous unresolved problem in the enumeration field is whether Hypergraph Dualization, or Dual for short, which asks for finding all minimal hitting sets for a given hypergraph [4], can be solved in output-polynomial time for general hypergraphs. This problem has received considerable attention in the literature, since it is known to be polynomially or quasi-polynomially equivalent with many problems in various areas [22]. When we reduce a problem from Dual, we say the problem is Dual-hard, meaning that if our problem could be solved in output-polynomial time, then Dual would also admit an algorithm running in output-polynomial time. See [34, 21] for a survey on these topics.

Enumeration problems related to connectivity for graphs are widely studied and considered important. One example of enumerating minimal subgraphs that satisfy a connectivity constraint is the minimal Steiner tree enumeration. For a restricted input, a polynomial-delay enumeration algorithm has been developed, using a polynomial-time algorithm for a corresponding extension problem as a subroutine [27]. Furthermore, similar algorithms are known to work for several variants, such as minimal directed Steiner trees, minimal terminal Steiner tree, and minimal Steiner forests [27, 29]. Besides edge connectivity, enumeration problems concerning vertex connectivity have also been studied. The problem of enumerating minimal spanning k-vertex-connected spanning subgraphs is known to be solvable in incremental polynomial time for fixed k [8]. Additionally, it is known that vertex connectivity can make an enumeration problem significantly harder. For instance, while the minimal vertex cover enumeration is known to be solvable with polynomial delay, the problem of enumerating minimal connected vertex covers, which imposes a connectivity constraint, is known to be Dual-hard [28]. Complexity of enumeration problems related to connectivity also extends to intractable problems. For example, it is known that enumerating minimal edge sets that ensure reachability between two specified vertices in a directed graph cannot be solved in output-polynomial time unless P = NP [25]. Moreover, instead of enumerating minimal sets that satisfy connectivity, enumerating minimal sets that destroy connectivity has also been addressed. In particular, it has been shown that enumerating minimal vertex sets that disconnect a graph and minimal edge sets that break the strong connectivity of a directed graph cannot be solved in output-polynomial time unless P = NP [7, 10].

[3] For $X =$ one2all, one2all2one, many2all, we assume that s or all s_i are contained in $G[E'']$.

Our Contribution. In this paper, we establish the results summarized in Table 1. Notably, we observe that the Extension Problem for all definitions of temporal spanners is NP-complete.[4] As a consequence, we cannot directly apply the standard flashlight method [32], which is commonly used to design output-polynomial enumeration algorithms. The flashlight method systematically explores the entire solution space by partitioning it into subsets that include or exclude a given edge, proceeding recursively only after verifying whether a partial solution can be extended into a full solution. However, since the Extension Problem is intractable, this verification cannot be performed in polynomial time unless P = NP.

On the other hand, if we enforce connectivity in the partial solution, the problem reduces to the Connected Extension Problem. We prove that in the case of one2all spanners, this problem can be solved in polynomial time, allowing us to design a polynomial-delay flashlight method for enumeration. This parallels the case of Steiner subgraphs in static graphs [19], where the Extension Problem is NP-complete, but the Connected Extension Problem is sometimes tractable, enabling an output-polynomial enumeration algorithm in these cases.

For all other definitions of temporal spanners, however, the Connected Extension Problem remains NP-complete.[5] This is not coincidental: we show that the enumeration of one2all and one2all2one spanners is Dual-hard. Consequently, achieving an output-polynomial time algorithm for these enumeration problems would imply an output-polynomial time for Dual, which is still an open problem since several years [4, 22]. It is important to note that hardness results for both Connected Extension Problem and Enumeration for many2all are tight with respect to the number of sources, as they hold for two sources.

Finally, for all2all spanners, we prove that no output-polynomial enumeration algorithm is possible unless P = NP. In this case, the latter result also implies the hardness of Connected Extension Problem and Extension Problem, as solving them in polynomial time would allow us to design an output-polynomial enumeration algorithm for Enumeration.

Structure of the paper. The organization is as follows: first, in Section 2, we present our models and notation used throughout the rest of the paper; then, in Section 3, we start by studying one2all spanners and obtain our tractability results; in Section 4 and Section 5, the connectivity considered is slightly modified (to one2all2one and many2all resp.) and we observe a sudden shift as the corresponding extension problems are proven to become hard and we prove the enumeration problems to be Dual-hard; next, we consider all2all spanners in Section 6 for which we obtain that the corresponding enumeration problem cannot be solved in output-polynomial time unless P = NP. We conclude in Section 7 and discuss interesting directions for future work.

Due to the soft page limit, results marked with a ($\star$) have the corresponding proofs in the Appendix. For some of these, a proof sketch is present in the main part of the paper as well.

2 Preliminaries

We denote a temporal graph $\mathcal{G}$ as a pair (G, λ) with G the underlying graph of $\mathcal{G}$, and λ the time function of $\mathcal{G}$. Graph G is undirected and simple, and composed of a vertex set $V(\mathcal{G})$ and an edge set $E(\mathcal{G}) \subseteq \binom{V(\mathcal{G})}{2}$, and associated to each edge $e \in E(\mathcal{G})$ is a set of times

[4] Both Extension Problem and Connected Extension Problem are in NP, as we can verify in polynomial time whether a set of edges contains E'' and is a temporal spanner, by checking reachability, and then checking minimality by removing one edge at a time and again checking reachability.

[5] Note that the hardness of Extension Problem is a consequence of the hardness of Connected Extension Problem as it is a restriction.

Table 1 We use m for the number of edges, and τ for the lifetime (i.e. the maximum assigned time to the edges) of the input temporal graph. All results hold even when lifetime τ is constant. Note that, for MANY2ALL ENUMERATION, if $k = n$, then the problem becomes ALL2ALL ENUMERATION. We use the $\Leftarrow$ sign to mean that the corresponding result is implied by the result on the right-hand side in the table.

X	EXTENSION	CONNECTED EXTENSION	ENUMERATION
ONE2ALL	NP-c (Theorem 3)	Poly (Lemma 4)	$O(m^2\tau)$ delay (Theorem 5)
ONE2ALL2ONE	(NP-c $\Leftarrow$)	NP-c (Theorem 6)	Dual-hard (Theorem 7)
MANY2ALL	(NP-c $\Leftarrow$)	NP-c even if $k = 2$ (Theorem 8)	Dual-hard even if $k = 2$ (Theorem 9)
ALL2ALL	(NP-c $\Leftarrow$)	(NP-c $\Leftarrow$)	No output poly algorithm unless P = NP (Theorem 12)

(or labels) $\lambda(e)$. As a slight abuse of notation, we simply use V and E as the vertex and edge set when the corresponding temporal graph is clear from the context. Also, we usually drop the set notation for the times on an edge, especially in figures and when the edge only has one label. The size of a temporal graph is polynomial in $n = |V|$, $m = |E|$, and $\tau = \max \cup_{e \in E} \lambda(e)$, the latter being denoted as the lifetime of the temporal graph. We use $\mathcal{G}[E']$ to denote the temporal graph $\mathcal{G}$ restricted to the edge set $E' \subseteq E$, and G_t for some integer $t \leq \tau$ denotes the subgraph of G such that $\forall e \in G_t : t \in \lambda(e)$, i.e. it represents the temporal graph at the specific time t.

In temporal graphs, paths and connectivity occur over time. This means that a path in the underlying graph G is a valid temporal path in temporal graph $\mathcal{G}$ if and only if the corresponding times on the edges are strictly increasing[6]. Formally, for a temporal path $\mathcal{P} = ((e_1, t_1), (e_2, t_2), ...(e_k, t_k))$ of length k, we have that $P = (e_1, e_2, ...e_k)$ is a path in G, and $\forall 1 \leq i \leq k-1 : t_i < t_{i+1}$. When a temporal path exists from a vertex s to a vertex v, we say s can reach v, and v is reachable from s. Note that even though we work in undirected graphs, the temporal aspect of paths implies that if s can reach v, then v does not necessarily reach s. We define an edge e in a temporal graph $\mathcal{G}$ to be *necessary* when without it, $\mathcal{G}$ is not connected anymore (for one of our notions of connectivity), i.e. if $\mathcal{G}$ is X connected for $X \in \{$ONE2ALL, ONE2ALL2ONE, MANY2ALL, ALL2ALL$\}$, edge e is necessary if $\mathcal{G} \setminus e$ is not X connected. When useful, we can specify which reachability the edge e is necessary for. For example, if it is necessary for some vertex u to reach some vertex v, then it implies that all temporal paths in $\mathcal{G}$ from u to v contain e, and that in $\mathcal{G} \setminus e$ no temporal path exists from u to v. Note that, in a spanner, all edges are necessary because otherwise they could be removed and the spanner wouldn't be minimal. We will use this notion of necessity of edges in our proofs to force edges to be part of all spanners of constructed temporal graphs.

An algorithm we will use and adapt for our tractability results in this paper is the following search algorithm in temporal graphs. Instead of prioritizing closest or furthest vertices (as in `Breadth First Search` or `BFS`, and `Depth First Search` or `DFS` resp.), this algorithm prioritizes earliest times to extend the search. One of the first papers to introduce, analyze, and use it is [35], and since then it is used throughout works on connectivity in temporal graphs, as it allows for computing (earliest) arrival times, as well as the corresponding

[6] Depending on the context, non-decreasing times can be of interest as well, and these two cases, commonly referred to as *strict* and *non-strict*, are both studied in temporal graph theory.

underlying paths, from a vertex s to all other vertices. We refer to it in this paper as `Time First Search` or `TFS` and propose the pseudo code in Algorithm 1 in the appendix. `TFS` runs in $O(m\tau)$ time. For one of our results, we alter the initialization of `TFS`, starting with different arrival times $a(v)$, and with a non-empty selected edge set S.

The reductions we present in this paper are from the Satisfiability Problem (or SAT) and from Hypergraph Dualization (or Dual). The former is a decision problem that has as input a CNF logic formula of n variables x_i and m clauses C_j and asks if the formula evaluates to `true` for some configuration of the variables. The latter is an enumeration problem with the input being a hypergraph $G = (U, H)$ on n vertices and m hyperedges, and the task being to enumerate all minimal transversals (or hitting sets) of the hypergraph, i.e. all minimal subsets $U' \subseteq U$ such that for all $h \in H$, the intersection $U' \cap h$ is non-empty.

The goal of an enumeration problem is to output the set of solutions $\mathcal{S}$. In the *binary partition* approach, we divide this problem into two subproblems. The goal of one subproblem is to output $\mathcal{S}_0$, while the goal of the other subproblem is to output $\mathcal{S}_1$, where $\mathcal{S}_0$ and $\mathcal{S}_1$ are disjoint and $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_0$[7]. The partition procedure can be regarded as a rooted tree, and thus we call this tree structure *the binary partition tree* (or partition tree).

To design an efficient binary partition algorithm, a key problem is to design a dividing procedure such that both $\mathcal{S}_0$ and $\mathcal{S}_1$ are non-empty. This is where designing and solving a corresponding *extension problem* is helpful, as it allows to predict whether some solution exists in $\mathcal{S}_0$ or $\mathcal{S}_1$, abandoning fruitless partition tree branches early which can improve the running time significantly. This is often referred to as the *flashlight* approach.

3 one2all spanners

one2all spanners have the following specific structure which we use in this section and whose proof is provided in the appendix for space constraints.

▶ **Proposition 2.** ($\star$) *A one2all-spanner is a tree.*

This section presents a binary partition approach to enumerate one2all spanners by systematically including or excluding each edge. In order to obtain a polynomial bound on the delay, we explore the corresponding extension problems. We first establish that the general Extension Problem, which could be used by the binary partition algorithm to discover whether the edges included in the partial solution can be completed in a solution, is intractable (Theorem 3). Since the general extension problem is NP-complete, we explore another extension problem where the edges of the binary partition are not included in an arbitrary order, but in an order such that they induce a connected graph including s. In this case, we obtain tractability for the extension problem (Lemma 4). The latter result serves as the foundation for our enumeration algorithm

▶ **Theorem 3.** ($\star$) *one2all Extension Problem is NP-complete, even if the lifetime $\tau = 2$.*

Proof (Sketch). To prove NP-completeness, we reduce from SAT. Let us first describe the transformation of a SAT instance into a spanner extension instance. We prove that a positive SAT instance corresponds to a positive instance in the spanner extension instance ($\Longrightarrow$), and vice versa ($\Longleftarrow$) in the complete proof in the appendix.

[7] In some cases, the problem may be divided into two or more subproblems.

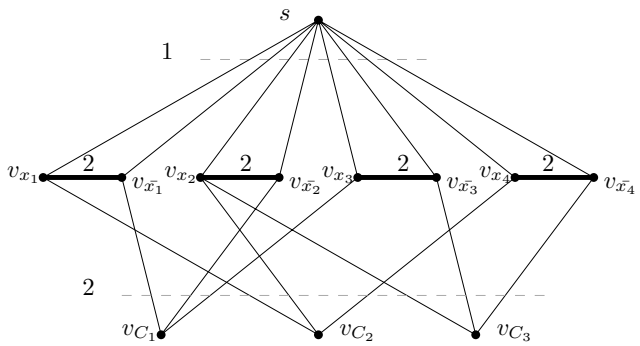


Figure 1 Example transformation of SAT instance $(\bar{x_1} \vee \bar{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \bar{x_3} \vee \bar{x_4})$ to a ONE2ALL spanner extension instance. The fat edges between vertices v_{x_i} and $v_{\bar{x_i}}$ are the edges E'' that have to be in a solution spanner. The dashed gray lines represent that all the edges incident to s have time 1, and all edges adjacent to some v_{C_j} have time 2.

For the following transformation, refer to Figure 1 as well. Let the SAT instance have n variables x_i and m clauses C_j. Start by constructing an (initially) empty temporal graph $\mathcal{G}$ and add vertices for each possible literal, so v_{x_i} and $v_{\bar{x_i}}$ for all x_i. Now, for each clause C_j, create vertex v_{C_j} and connect this vertex with an edge to each vertex corresponding to a literal of C_j. Add label 2 to these edges as well. Then, add vertex s and add edges $\{s, v_\ell\}$ for each possible literal v_ℓ. Assign label 1 to these edges. Finally, for each v_{x_i}, add edge $\{v_{x_i}, v_{\bar{x_i}}\}$ with label 2, and let these edges be the set of edges E'' that have to be in a spanner solution. The transformation is now complete.

We note that, by construction, at least one of edges $\{s, v_{x_i}\}$ and $\{s, v_{\bar{x_i}}\}$ has to be in a solution for vertices x_i and $\bar{x_i}$ to be reachable from s. In fact, exactly one of the two must be in a solution because if both are then a cycle in the underlying graph would be created along with forced edge $\{v_{x_i}, v_{\bar{x_i}}\} \in E''$, which would contradict Proposition 2. The intuition behind our reduction is that the choice between edges $\{s, v_{x_i}\}$ and $\{s, v_{\bar{x_i}}\}$ encodes the choice of setting variable x_i to true or false respectively. Note also that it is necessary for the reachability of each v_{C_j} to select at least one pair of edges $\{s, v_\ell\}$ and $\{v_\ell, v_{C_j}\}$. The correctness is provided in the full proof in the appendix. ◀

▶ **Lemma 4.** *ONE2ALL CONNECTED EXTENSION PROBLEM is solvable in time $O(m\tau)$.*

Proof. We start by running TFS on $\mathcal{G}' = \mathcal{G}[E'']$ to obtain the earliest arrival times $a_{\mathcal{G}'}(v)$ from s to each vertex $v \in \mathcal{G}'$. Afterwards, we run TFS on temporal graph $\mathcal{G}$, but change the *initialization* part of the algorithm (see Algorithm 1 in the Appendix). Initialize the earliest arrival times $a(v)$ as $a_{\mathcal{G}'}(v)$ for every vertex v of $\mathcal{G}'$, and initialize the selected edge set S as E''. This makes it so that these arrival times are never updated throughout TFS, as only arrival times that are $+\infty$ get updated. Also, S, the resulting ONE2ALL spanner (if one exists) uses E''. TFS takes time $O(m\tau)$, and thus so does the described algorithm. ◀

As discussed before, we obtain our enumeration algorithm based on binary partition. This algorithm has polynomial delay thanks to the flashlight method using Lemma 4. For the sake of space, the proof of the following result is provided in the appendix.

▶ **Theorem 5.** (⋆) *ONE2ALL ENUMERATION can be solved with $O(m^2\tau)$ delay.*

4 ONE2ALL2ONE spanners

In this section, we explore the enumeration of spanners that not only allow a vertex s to reach all other vertices, as in the previous section, but also ensure that all vertices can reach s. These are called ONE2ALL2ONE spanners. Importantly, we do not require the temporal path from any vertex v to s to occur after the temporal path from s to v arrives.

We lose the underlying tree structure of ONE2ALL spanners in Proposition 2, as it can easily be observed in the following temporal graph: take the cycle graph on four vertices s, u, v, and w and time the edges $\{s,u\}$, $\{u,v\}$, $\{v,w\}$, $\{w,s\}$, with 1, 2, 3, and 4 resp. The only ONE2ALL2ONE spanner is the temporal graph itself, which is not a tree.

Another difference is that even if the edge set E'' for the extension problem is connected, then the problem remains intractable as opposed to becoming tractable as was the case for ONE2ALL spanners.

▶ **Theorem 6.** *ONE2ALL2ONE CONNECTED EXTENSION PROBLEM is* NP*-complete, even if the lifetime is constant.*

Proof (Sketch). For proving NP-hardness, consider the same transformation as in Theorem 3, from SAT (see again Figure 1). Then, for each vertex v_z where z is some literal or clause, add vertices v_z^1 and v_z^2 and edges $\{v_z, v_z^1\}$, $\{v_z^1, v_z^2\}$, and $\{v_z^2, s\}$, with times 3, 4, 5 resp. Add to edge set E'' (which already contains edges $\{v_{x_i}, v_{\bar{x_i}}\}$ for all variables x_i) these newly created edges. Note that E'' is connected. The transformation is now complete.

Since all edges $\{v_z, v_z^1\}$, $\{v_z^1, v_z^2\}$, and $\{v_z^2, s\}$ are in E'', and they allow vertices v_z to reach s, and they do not allow s to reach vertices v_z (nor aid in the reachability between these vertices that could indirectly help s to reach them), essentially what remains to decide in this reduction is how to connect s to the other vertices. This is exactly the problem in Theorem 3, and we use the exact same structure for this (ignoring the new edges we added as we just argued they are not useful for the reachability of s). Hence we conclude that the result from the reduction in Theorem 3, being NP-hardness, transfers for ONE2ALL2ONE CONNECTED EXTENSION PROBLEM. ◀

▶ **Theorem 7.** (⋆) *ONE2ALL2ONE ENUMERATION is* Dual*-hard, even if the lifetime is constant.*

Proof (Sketch). We reduce HYPERGRAPH DUALIZATION, or Dual, to our problem. Let us first describe the transformation of a Dual instance into a ONE2ALL2ONE spanner enumeration instance. Then, it is possible to prove that each minimal transversal in the Dual instance corresponds to a ONE2ALL2ONE spanner in the ONE2ALL2ONE spanner enumeration instance, and vice versa. This is proven in the appendix for space constraints.

For the following transformation, refer to Figure 2 as well. Let the Dual instance G be composed of n vertices u_i and m hyperedges h_j. Start by constructing an initially empty temporal graph $\mathcal{G}$ and add vertex v_{u_i} for each vertex $u_i \in G$, and vertex v_{h_j} for each hyperedge $h_j \in G$. For all i, j, if $u_i \in h_j$, then create a vertex v_{u_i,h_j} in $\mathcal{G}$ and edges $\{v_{u_i}, v_{u_i,h_j}\}$ and $\{v_{u_i,h_j}, v_{h_j}\}$ with times $\{3,4\}$ and $\{1,4\}$ resp. Add vertex s to $\mathcal{G}$. For each v_{u_i}, add vertices $v_{u_i}^0$, $v_{u_i}^1$, and $v_{u_i}^2$, and edges $\{s, v_{u_i}^0\}$, $\{v_{u_i}^0, v_{u_i}\}$ with times 1 and 2 resp. as well as edges $\{s, v_{u_i}^1\}$, $\{v_{u_i}^1, v_{u_i}^2\}$, and $\{v_{u_i}^2, v_{u_i}\}$ with times $\{1,3\}$, 2, and $\{1,3\}$ resp. Finally, add vertex t and edges $\{v_{h_j}, t\}$ with time 2 for all h_j, and edge $\{t, s\}$ with time 3. The transformation is now complete.

By construction, we obtain that all edges except for edges $\{v_{u_i}^0, v_{u_i}\}$ are necessary, i.e. that they are part of any ONE2ALL2ONE spanner of $\mathcal{G}$ w.r.t. source vertex s. Indeed, we have that: *(i)* Each edge $\{s, v_{u_i}^0\}$ is necessary for s to reach vertex $v_{u_i}^0$; *(ii)* Each edge $\{s, v_{u_i}^1\}$ is

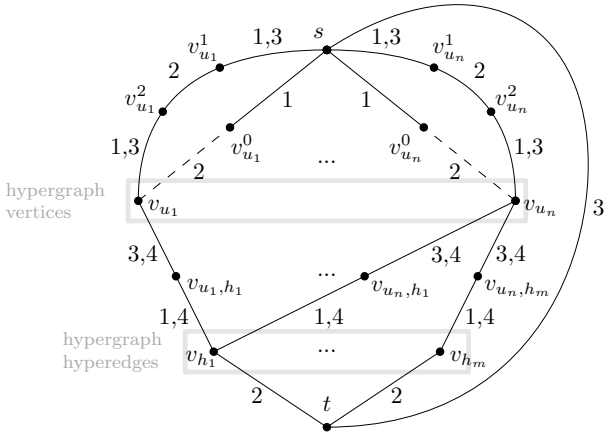


Figure 2 Illustration of the transformation of a Dual instance to a ONE2ALL2ONE spanner enumeration instance. All solid edges are necessary in all spanners, while specific subsets of dashed edges determine specific spanners. Note that in this specific illustration, hyperedge h_1 contains (at least) vertices u_1 and u_n, and hyperedge h_m contains (at least) vertex u_n.

necessary for vertex v_{u_i} to reach s; *(iii)* And likewise for each edge $\{v^1_{u_i}, v^2_{u_i}\}$ and each edge $\{v^2_{u_i}, v_{u_i}\}$; *(iv)* Each edge $\{v_{u_i}, v_{u_i,h_j}\}$ is necessary for s to reach vertex v_{u_i,h_j}; *(v)* Each edge $\{v_{u_i,h_j}, v_{h_j}\}$ is necessary for vertex v_{u_i,h_j} to reach s; *(vi)* Each edge $\{v_{h_j}, t\}$ is necessary for vertex v_{h_j} to reach s; *(vii)* And finally, edge $\{t, s\}$ is necessary for t to reach s.

Let $\mathcal{G}'$ denote the temporal graph composed of these edges, i.e. $\mathcal{G}' = \mathcal{G} \setminus \{v^0_{u_i}, v_{u_i}\}$, for all u_i (in Figure 2, this corresponds to the illustrated temporal graph without the dashed edges). We have that in $\mathcal{G}'$, all vertices can be reached by s as well as reach s, except for vertices v_{h_j}, as these can only reach s but not be reached by s. Note that adding some edge $\{v^0_{u_i}, v_{u_i}\}$ to $\mathcal{G}'$ will allow s to reach exactly vertices v_{h_j} such that $u_i \in h_j$. We thus observe that any spanner of $\mathcal{G}$ is composed of $\mathcal{G}'$ with some minimal selection of edges $\{v^0_{u_i}, v_{u_i}\}$. Correctness is provided in the complete proof in the appendix. ◀

5 MANY2ALL spanners

This section explores another natural variant of a spanner, where instead of having only a single source vertex s that has to visit all vertices, we consider multiple such source vertices $s_1, ..., s_k$ in MANY2ALL spanners, each needing to visit all vertices.

For the sake of space, the proofs of the following results are provided in the appendix as they are similar to the ones in Section 4. The challenge here is to obtain tight reductions in terms of sources, as for only one source we know that CONNECTED EXTENSION PROBLEM is polynomial and ENUMERATION admits a polynomial delay algorithm.

▶ **Theorem 8.** (⋆) *MANY2ALL CONNECTED EXTENSION PROBLEM is* NP*-complete, even if the lifetime is constant and the number of source vertices is 2.*

▶ **Theorem 9.** (⋆) *MANY2ALL ENUMERATION is* Dual*-hard, even if the lifetime is constant, and the number of source vertices is 2.*

6 ALL2ALL spanners

This section is concerned with ALL2ALL spanners, which is where all vertices play the role of source, i.e. all vertices need to reach each other. We present a reduction inspired by the one used in [25] to prove that their problem of enumerating disjunctions and conjunctions of paths cannot be done in output-polynomial time unless P = NP.

We introduce the following auxiliary problem, used only in this section. Let ANOTHER ALL2ALL SPANNER be the decision problem defined as: given a temporal graph $\mathcal{G}$ and a set of k ALL2ALL spanners S, does there exist another ALL2ALL spanner of $\mathcal{G}$ which is not in S? We first prove this problem to be NP-complete, through a technical reduction. Afterwards, we use this result to show that if an output-polynomial time algorithm exists for ALL2ALL ENUMERATION, it would imply that a polynomial time algorithm exists for ANOTHER ALL2ALL SPANNER.

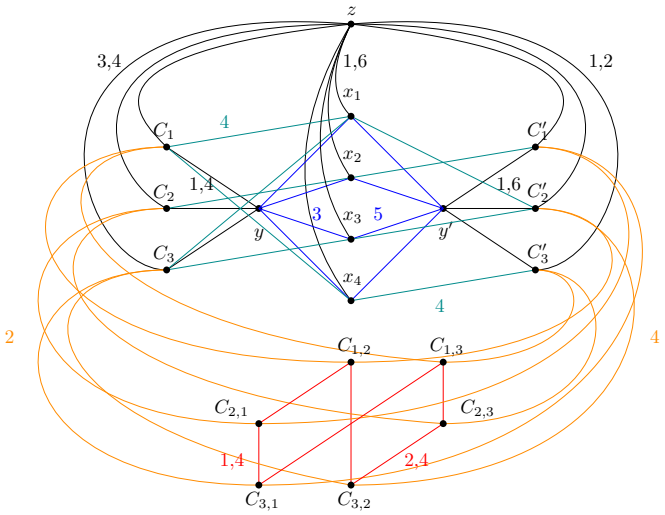
Construction of the temporal graph. To prove NP-completeness for ANOTHER ALL2ALL SPANNER, we reduce from SAT. Let us first present how to construct temporal graph $\mathcal{G}$ for the ANOTHER ALL2ALL SPANNER instance from a given SAT instance ϕ, and analyze the specific structure it admits in Lemma 10.

For the following transformation, refer to Figure 3 as well. Let the SAT instance be CNF formula ϕ on n variables x_i and m clauses C_j. Start by creating the (initially empty) temporal graph $\mathcal{G}$ and add vertices for each possible variable, so v_{x_i} for all x_i. Now, for each clause C_j, create vertices v_{C_j} and $v_{C'_j}$. Add vertex v_y and vertex $v_{y'}$ as well. Now add edges from v_y to all vertices v_{C_j} with times $\{1, 4\}$, and add edges from $v_{y'}$ to all vertices $v_{C'_j}$ with times $\{1, 6\}$. Create vertex v_z and connect it to all vertices v_{C_j} with times $\{3, 4\}$, to all vertices v_{x_i} with times $\{1, 6\}$, and to all vertices $v_{C'_j}$ with times $\{1, 2\}$. (The transformation now corresponds to the black edges in Figure 3a and Figure 3b. We will now add the other vertices, and the orange and red edges.) For all ordered pairs $(v_{C_i}, v_{C'_j})$ such that $i \neq j$, create vertex $v_{C_{i,j}}$ and connect it to vertex v_{C_i} with time 2, and to $v_{C'_j}$ with time 4. For all pairs of vertices $(v_{C_{i,j}}, v_{C_{h,k}})$, if $j = k$, connect them with an edge with times $\{1, 4\}$; and if $i = k$ and $j = h$ (i.e. $v_{C_{h,k}} = v_{C_{j,i}}$) connect them with an edge with times $\{2, 4\}$. We will use $\mathcal{G}'$ to refer to the temporal subgraph of $\mathcal{G}$ containing only the edges created until now. (The construction at this point now corresponds to Figure 3b.) We finish the construction of $\mathcal{G}$ by adding the following final edges. For all i, j, if literal $x_i \in C_j$, connect v_{x_i} to v_{C_j} with an edge, and if $\bar{x}_i \in C_j$, connect v_{x_i} to $v_{C'_j}$. Assign time 4 to these edges. Connect vertex v_y with an edge of time 3 to each vertex v_{x_i}. Similarly, connect vertex $v_{y'}$ with an edge of time 5 to each vertex v_{x_i}. This concludes the construction of the temporal graph $\mathcal{G}$. Let $\mathcal{G}''$ be the temporal graph $\mathcal{G}$ without $\mathcal{G}'$, i.e. the temporal graph containing only the final edges added (or the temporal graph corresponding to Figure 3c).

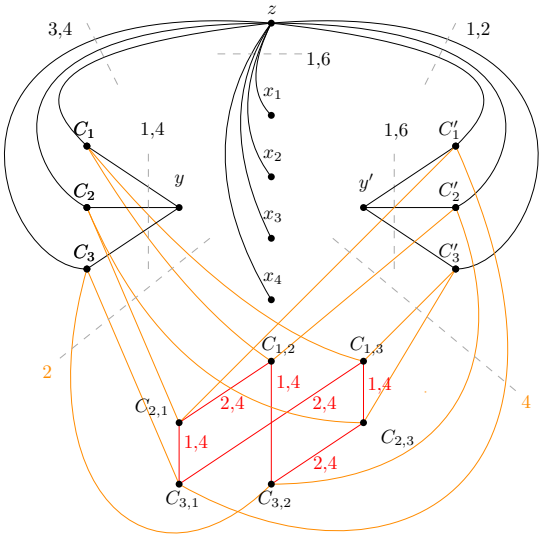
▶ **Lemma 10.** ($\star$) *Temporal graph $\mathcal{G}$ has the following structural properties:*

- ***Reachability:*** *In $\mathcal{G}$, all vertices can reach each other, while in $\mathcal{G}'$, all vertices can reach each other except for vertices v_{C_i} which do not reach corresponding vertices $v_{C'_i}$, for all i;*
- ***Necessity:*** *In $\mathcal{G}$ and in $\mathcal{G}'$, all edges of $\mathcal{G}'$ are necessary for some vertex pair reachability;*
- ***Spanner:*** *Any spanner of $\mathcal{G}$ must be composed of $\mathcal{G}'$ and some minimal edge set of $\mathcal{G}''$, and adding to $\mathcal{G}'$ edges $\{v_y, v_{x_i}\}$ and $\{v_{x_i}, v_{y'}\}$, for any i, results in a spanner.*

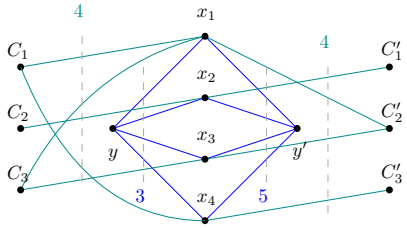
For the following result, note that ANOTHER ALL2ALL SPANNER is in NP because one can verify a solution spanner in polynomial time, by checking if it is contained in S, checking reachability, and then checking minimality by removing one edge at a time and again checking reachability.



(a) Temporal graph $\mathcal{G}$, composed of $\mathcal{G}'$, presented in more detail in (b), and $\mathcal{G}''$, presented in detail in (c). The main aim of this subfigure is to show how $\mathcal{G}'$ and $\mathcal{G}''$ connect to form $\mathcal{G}$.



(b) Temporal subgraph $\mathcal{G}'$ of $\mathcal{G}$. The black edges allow for most reachability such as between all C_i and all x_j, and from all C'_k to all C_i. The orange edges allow the very specific reachability of all C_i to all C'_j such that $i \neq j$, and the red edges ensure reachability for vertices $C_{i,j}$.



(c) Temporal subgraph $\mathcal{G}''$ of $\mathcal{G}$. The green edges with time 4 depend on the structure of the clauses of the SAT instance, while the blue edges encode the assignment of the SAT variables in a solution spanner.

Figure 3 Example construction of temporal graph $\mathcal{G}$ for SAT formula $(x_1 \vee \bar{x}_2 \vee x_4) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee x_3 \vee \bar{x}_4)$. For clarity, "$v$" notation for the vertices is dropped (e.g. vertex v_z is depicted simply as z), and dashed lines indicate that the corresponding times are assigned to all touched edges of the same color. The edges are color coded to group edges with a similar purpose together, as well as for clarity.

▶ **Lemma 11.** *Another all2all Spanner is* NP*-complete, even if* $k = O(|\mathcal{G}|)$.

Proof. As mentioned before, to prove NP-hardness, we reduce SAT to our problem. Let the polynomial transformation of a SAT instance into a Another all2all Spanner instance be as described beforehand to obtain temporal graph $\mathcal{G}$, and let S contain all the spanners described in the spanner property of Lemma 10, so for each $1 \leq i \leq n$, add to S spanner s_i composed of $\mathcal{G}'$ with added edges $\{v_y, v_{x_i}\}$ and $\{v_{x_i}, v_{y'}\}$. Note that now $|S| = k = n$. The transformation is now complete. We finish by proving that a positive SAT instance implies a positive Another all2all Spanner instance ($\implies$), and vice versa ($\impliedby$).

SAT $\implies$ **Another** all2all **Spanner:**
Suppose a solution assignment a for the SAT instance. We construct a spanner s such that $s \notin S$ as follows. By the spanner property in Lemma 10, s must contain all edges of $\mathcal{G}'$. For each clause C_j, some literal must be `true` in a. Consider one of these literals. If it is a positive variable, so x_i, then add to s edges $\{v_{C_j}, v_{x_i}\}$ and $\{v_{x_i}, v_{y'}\}$ (unless already in s). These edges are now necessary for v_{C_j} to reach $v_{C'_j}$. After doing this for all clauses, we claim s is indeed a spanner: all edges in s are necessary, so s is minimal, and all vertices can reach each other (all vertices could already reach each other before going over the clauses, except for v_{C_j} to $v_{C'_j}$, which can reach through the added edges afterwards). We note that $s \notin S$ since by construction no spanner in S uses any edge $\{v_{C_j}, v_{x_i}\}$ or $\{v_{x_i}, v_{C'_j}\}$, for any i, j.

SAT $\impliedby$ **Another** all2all **Spanner:**
Suppose a solution all2all spanner s for the Another all2all Spanner instance. Again by the spanner property of Lemma 10, we know it must contain all edges from $\mathcal{G}'$. Also, it cannot contain both edges $\{v_y, v_{x_i}\}$ and $\{v_{x_i}, v_{y'}\}$ for some i, because that would mean $s_i \in S$ is a subgraph of s, and thus s is not minimal (if a proper subgraph) or already in S (if equal to s_i). Thus, s must have at most one of the edges $\{v_y, v_{x_i}\}$ and $\{v_{x_i}, v_{y'}\}$ for all i. We construct a solution for SAT in the following manner: for all i, if $\{v_y, v_{x_i}\} \in s$, assign x_i to be `false`, and if $\{v_{x_i}, v_{y'}\} \in s$, then assign x_i to be `true`. If neither edge is in s, then assign x_i an arbitrary value, say `true`. We now claim this assignment evaluates ϕ to `true`. It should be clear that no x_i is assigned both `true` and `false`. For each clause C_j, there must be some path from v_{C_j} to $v_{C'_j}$, which must pass either through some edge $\{v_y, v_{x_i}\}$ or through some edge $\{v_{x_i}, v_{y'}\}$. In the first case, the other edge of the temporal path is $\{v_{x_i}, v_{C'_j}\}$ which by construction implies $\bar{x}_i \in C_j$, and since $\{v_y, v_{x_i}\} \in s$, x_i is assigned `false`, and so clause C_j evaluates to `true`. Similarly, in the second case, the other edge of the temporal path is $\{v_{C_j}, v_{x_i}\}$ which by construction implies $x_i \in C_j$, and since $\{v_{x_i}, v_{y'}\} \in s$, x_i is assigned `true`, and so clause C_j evaluates to `true`. ◀

Now, to obtain the main result of this section, we use the following folklore reasoning from the area of enumeration algorithms [31, 12, 26, 10, 9]. Suppose by contradiction that all2all Spanner Enumeration can be done in output-polynomial time, say through algorithm A running in time at most $(|\mathcal{G}|N)^c$, for an input temporal graph $\mathcal{G}$, with N the number of solutions, and some constant c. Now, consider an instance of Another all2all Spanner, composed of temporal graph $\mathcal{G}$ and a set of spanners S such that $|S| = k = O(\mathcal{G})$. Run algorithm A on $\mathcal{G}$ for $(|\mathcal{G}|k)^c$ steps. Since $k = O(|\mathcal{G}|)$, we run A for polynomial time. Now, if the algorithm terminates and enumerated only the spanners from S, then that implies the Another all2all Spanner instance is negative. Otherwise, if the algorithm does not terminate in this time or has enumerated a spanner not in S, it implies that the instance is positive. Thus, Another all2all Spanner could be solved in polynomial time, which, unless P = NP, is our contradiction because Lemma 11 proves it is NP-complete.

▶ **Theorem 12.** *ALL2ALL ENUMERATION cannot be done in output-polynomial time unless* P = NP.

7 Conclusion

In this paper, we have studied enumeration of minimal spanners in temporal graphs. We considered multiple variants, as well as on different kinds of connectivity: single source to all vertices; single source to all vertices and vice versa; multiple sources to all vertices; and all to all connectivity. We obtained multiple results for corresponding extension problems and enumeration problems, which were presented in Table 1.

In future works, it may be interesting to consider other variants of connectivity. In [15] (and more recently updated in [13]), multiple forms of connectivity for temporal graphs are presented in a hierarchy, some of which correspond to the ones studied in this paper, such as $\mathcal{TC}$ corresponding to ALL2ALL connectivity. Another interesting direction could be to consider time-edge spanners which, as discussed in the introduction, are not subsets of edges but subsets of time-edges, i.e. one is allowed to remove redundant times from edges. We believe our results do not directly adapt for this setting, e.g. some of our reductions crucially abuse the fact that when an edge is necessary concerning some time on it, we can then use the other times to create temporal paths which are useful for the reduction.

References

1 Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov, and Richard Spence. Graph spanners: A tutorial review. *Computer Science Review*, 37:100253, 2020. doi:10.1016/J.COSREV.2020.100253.

2 Eleni C Akrida, Leszek Gąsieniec, George B Mertzios, and Paul G Spirakis. The complexity of optimal design of temporally connected graphs. *Theory of Computing Systems*, 61(3), 2017. doi:10.1007/S00224-017-9757-X.

3 Kyriakos Axiotis and Dimitris Fotakis. On the size and the approximability of minimum temporally connected subgraphs. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.

4 Claude Berge. *Hypergraphs: combinatorics of finite sets*, volume 45. Elsevier, 1984.

5 Davide Bilò, Gianlorenzo D'Angelo, Luciano Gualà, Stefano Leucci, and Mirko Rossi. Sparse temporal spanners with low stretch. In *30th Annual European Symposium on Algorithms (ESA 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

6 Davide Bilò, Gianlorenzo D'Angelo, Luciano Gualà, Stefano Leucci, and Mirko Rossi. Blackout-tolerant temporal spanners. *Journal of Computer and System Sciences*, 141:103495, 2024. doi:10.1016/J.JCSS.2023.103495.

7 E. Boros, K. Elbassioni, V. Gurvich, and L. Khachiyan. Enumerating minimal dicuts and strongly connected subgraphs and related geometric problems. In Daniel Bienstock and George Nemhauser, editors, *Integer Programming and Combinatorial Optimization*, pages 152–162, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

8 Endre Boros, Konrad Borys, Khaled Elbassioni, Vladimir Gurvich, Kazuhisa Makino, and Gabor Rudolf. Generating minimal k-vertex connected spanning subgraphs. In Guohui Lin, editor, *Computing and Combinatorics*, pages 222–231, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

9 Endre Boros and Kazuhisa Makino. Generating minimal redundant and maximal irredundant subhypergraphs. *Discret. Appl. Math.*, 358:217–229, 2024. doi:10.1016/J.DAM.2024.07.006.

10 Caroline Brosse, Oscar Defrain, Kazuhiro Kurita, Vincent Limouzy, Takeaki Uno, and Kunihiro Wasa. On the hardness of inclusion-wise minimal separators enumeration. *Inf. Process. Lett.*, 185:106469, 2024. doi:10.1016/J.IPL.2023.106469.

11 Daniela Bubboloni, Costanza Catalano, Andrea Marino, and Ana Silva. On computing optimal temporal branchings and spanning subgraphs. *J. Comput. Syst. Sci.*, 148:103596, 2025. doi:10.1016/J.JCSS.2024.103596.

12 Fritz Bökler, Matthias Ehrgott, Christopher Morris, and Petra Mutzel. Output-sensitive complexity of multiobjective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 24(1-2):25–36, 2017. doi:10.1002/mcda.1603.

13 Arnaud Casteigts. Finding structure in dynamic networks. *CoRR*, abs/1807.07801, 75p, 2018. URL: http://arxiv.org/abs/1807.07801.

14 Arnaud Casteigts and Timothée Corsini. In search of the lost tree: Hardness and relaxation of spanning trees in temporal graphs. In *International Colloquium on Structural Information and Communication Complexity*, pages 138–155. Springer, 2024. doi:10.1007/978-3-031-60603-8_8.

15 Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012. doi:10.1080/17445760.2012.668546.

16 Arnaud Casteigts, Joseph G Peters, and Jason Schoeters. Temporal cliques admit sparse spanners. *Journal of Computer and System Sciences*, 121:1–17, 2021. doi:10.1016/J.JCSS.2021.04.004.

17 Arnaud Casteigts, Michael Raskin, Malte Renken, and Viktor Zamaraev. Sharp thresholds in random simple temporal graphs. *SIAM Journal on Computing*, 53(2):346–388, 2024. doi:10.1137/22M1511916.

18 Esteban Christiann, Eric Sanlaville, and Jason Schoeters. On inefficiently connecting temporal networks. In *3rd Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2024)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.

19 Alessio Conte, Roberto Grossi, Mamadou Moustapha Kanté, Andrea Marino, Takeaki Uno, and Kunihiro Wasa. Listing induced steiner subgraphs as a compact way to discover steiner trees in graphs. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 73:1–73:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.MFCS.2019.73.

20 Edsger W Dijkstra. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: his life, work, and legacy*, pages 287–290. 2022. doi:10.1145/3544585.3544600.

21 Thomas Eiter, Kazuhisa Makino, and Georg Gottlob. Computational aspects of monotone dualization: A brief survey. *Discrete Applied Mathematics*, 156(11):2035–2049, 2008. In Memory of Leonid Khachiyan (1952 - 2005). doi:10.1016/j.dam.2007.04.017.

22 Khaled Elbassioni, Imran Rauf, and Saurabh Ray. A global parallel algorithm for enumerating minimal transversals of geometric hypergraphs. *Theoretical Computer Science*, 767:26–33, 2019. doi:10.1016/J.TCS.2018.09.027.

23 Hovhannes Harutyunyan, Arthur Liestman, Joseph Peters, and Dana Richards. Broadcasting and Gossiping. In Ping Zhang, editor, *Handbook of Graph Theory, Second Edition*, volume 20134658, pages 1477–1494. Chapman and Hall/CRC, December 2013. Series Title: Discrete Mathematics and Its Applications. doi:10.1201/b16132-87.

24 David Kempe, Jon Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 504–513, 2000. doi:10.1145/335305.335364.

25 Leonid Khachiyan, Endre Boros, Khaled Elbassioni, Vladimir Gurvich, and Kazuhisa Makino. Enumerating disjunctions and conjunctions of paths and cuts in reliability theory. *Discrete Applied Mathematics*, 155(2):137–149, 2007. 29th Symposium on Mathematical Foundations of Computer Science MFCS 2004. doi:10.1016/j.dam.2006.04.032.

26 Leonid Khachiyan, Endre Boros, Khaled M. Elbassioni, and Vladimir Gurvich. On enumerating minimal dicuts and strongly connected subgraphs. *Algorithmica*, 50(1):159–172, 2008. doi:10.1007/S00453-007-9074-X.

27 Benny Kimelfeld and Yehoshua Sagiv. Efficiently enumerating results of keyword search over data graphs. *Inf. Syst.*, 33(4-5):335–359, 2008. doi:10.1016/J.IS.2008.01.002.

28 Yasuaki Kobayashi, Kazuhiro Kurita, Yasuko Matsui, and Hirotaka Ono. Enumerating minimal vertex covers and dominating sets with capacity and/or connectivity constraints. In Adele Anna Rescigno and Ugo Vaccaro, editors, *Combinatorial Algorithms - 35th International Workshop, IWOCA 2024, Ischia, Italy, July 1-3, 2024, Proceedings*, volume 14764 of *Lecture Notes in Computer Science*, pages 232–246. Springer, 2024. doi:10.1007/978-3-031-63021-7_18.

29 Yasuaki Kobayashi, Kazuhiro Kurita, and Kunihiro Wasa. Linear-delay enumeration for minimal steiner problems. In Leonid Libkin and Pablo Barceló, editors, *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 301–313. ACM, 2022. doi:10.1145/3517804.3524148.

30 Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

31 E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Generating all maximal independent sets: Np-hardness and polynomial-time algorithms. *SIAM Journal on Computing*, 9(3):558–565, 1980. doi:10.1137/0209042.

32 Arnaud Mary and Yann Strozecki. Efficient enumeration of solutions produced by closure operations. *Discret. Math. Theor. Comput. Sci.*, 21(3), 2019. doi:10.23638/DMTCS-21-3-22.

33 Robert Clay Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401, 1957.

34 Yann Strozecki. Enumeration complexity. *Bull. EATCS*, 129, 2019. URL: http://bulletin.eatcs.org/index.php/beatcs/article/view/596/605.

35 B Bui Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003. doi:10.1142/S0129054103001728.

Algorithm 1 `Time First Search` (from [35], among others)

Input: temporal graph $\mathcal{G}$ and source vertex $s \in V$
Output: Earliest arrival times $a(v)$ from s for all vertices v, and edge set S with corresponding temporal paths

```
// initialization
a(s) ← −∞
for v in V do
    a(v) ← +∞
S ← newSet()
// computation
for t ← 1 to τ do
    for e = {u, v} in E do
        if λ(e).contains(t) and a(u) < t and a(v) == +∞ then
            a(v) ← t
            S.add(e)
return a, S
```

A Omitted Proofs

A.1 Proof of Proposition 2

▶ **Proposition 2.** (⋆) *A* ONE2ALL*-spanner is a tree.*

Proof. Suppose by contradiction that a ONE2ALL-spanner S exists which is not a tree. We show that S is not minimal by computing a ONE2ALL-spanner S' in $\mathcal{G}[S]$ which is a tree. Apply TFS in $\mathcal{G}[S]$, and consider the output edge set to be S'. We claim that S' is a ONE2ALL spanner, i.e. it allows for s to reach all vertices, and it is minimal. Indeed, since s can reach all vertices in S, it can also do so through S', as TFS added the earliest possible edges from s to the other vertices to S'. Moreover, in the condition of an edge being added to S' implies that at all times S' is a tree, so S' is indeed minimal because removing any edge would break connectivity. ◀

A.2 Proof of Theorem 3

▶ **Theorem 3.** ($\star$) *ONE2ALL EXTENSION PROBLEM is* NP*-complete, even if the lifetime* $\tau = 2$.

Proof. To prove NP-completeness, we reduce from SAT. Let us first describe the transformation of a SAT instance into a spanner extension instance. We prove that a positive SAT instance corresponds to a positive instance in the spanner extension instance ($\implies$), and vice versa ($\impliedby$) in the complete proof in the appendix.

For the following transformation, refer to Figure 1 as well. Let the SAT instance have n variables x_i and m clauses C_j. Start by constructing an (initially) empty temporal graph $\mathcal{G}$ and add vertices for each possible literal, so v_{x_i} and $v_{\bar{x_i}}$ for all x_i. Now, for each clause C_j, create vertex v_{C_j} and connect this vertex with an edge to each vertex corresponding to a literal of C_j. Add label 2 to these edges as well. Then, add vertex s and add edges $\{s, v_\ell\}$ for each possible literal v_ℓ. Assign label 1 to these edges. Finally, for each v_{x_i}, add edge $\{v_{x_i}, v_{\bar{x_i}}\}$ with label 2, and let these edges be the set of edges E'' that have to be in a spanner solution. The transformation is now complete.

We note that, by construction, at least one of edges $\{s, v_{x_i}\}$ and $\{s, v_{\bar{x_i}}\}$ has to be in a solution for vertices x_i and $\bar{x_i}$ to be reachable from s. In fact, exactly one of the two must be in a solution because if both are then a cycle in the underlying graph would be created along with forced edge $\{v_{x_i}, v_{\bar{x_i}}\} \in E''$, which would contradict Proposition 2. The intuition behind our reduction is that the choice between edges $\{s, v_{x_i}\}$ and $\{s, v_{\bar{x_i}}\}$ encodes the choice of setting variable x_i to true or false respectively. Note also that it is necessary for the reachability of each v_{C_j} to select at least one pair of edges $\{s, v_\ell\}$ and $\{v_\ell, v_{C_j}\}$.

SAT $\implies$ ONE2ALL Spanner Extension:
Suppose a solution S for the SAT instance. We construct solution S' for the spanner extension instance by selecting edge $\{s, v_{x_i}\}$ if x_i is set to `true` in S, and $\{s, v_{\bar{x_i}}\}$ otherwise, and this for all x_i. Furthermore, since S is a solution for the SAT instance, each clause C_j must have some literal that is assigned a true value. Find such a literal for each clause C_j, denote it ℓ_j, and select edge $\{v_{\ell_j}, v_{C_j}\}$ to be included in S'. Note that $\{s, v_{\ell_j}\}$ must be selected as well. We now claim that S' is a spanner. Indeed, all vertices v_ℓ for any literal ℓ are reachable from s since either edge $\{s, v_\ell\}$ is selected (through which it is directly reachable) or edge $\{s, v_{\bar{\ell}}\}$ is selected (in which case it is reachable through vertex $v_{\bar{\ell}}$). The clause vertices v_{C_j} are all reachable as well since some edge $\{v_\ell, v_{C_j}\}$ is selected, and since corresponding edge $\{s, v_\ell\}$ is selected too, it implies s can reach v_{C_j} through v_ℓ. S' is minimal by the minimality observations in the previous paragraph.

SAT $\impliedby$ ONE2ALL Spanner Extension:
Suppose a solution ONE2ALL spanner S' for the spanner extension problem. We construct solution S for the SAT instance as follows. By the minimality observations, either $\{s, v_{x_i}\}$ or $\{s, v_{\bar{x_i}}\}$ is selected in S' for each variable x_i. If $\{s, v_{x_i}\} \in S'$, then set x_i to `true`, otherwise

set it to `false` in S. We now claim that S is a valid solution for the SAT instance. By construction, all variables are assigned to be either `true` or `false`. Moreover, since S' is a spanner, for each clause vertex v_{C_j} there must be some pair of edges $\{s, v_{\ell_j}\}$ and $\{v_{\ell_j}, v_{C_j}\} \in S'$, meaning literal $\ell_j \in C_j$ is assigned `true` and so all clauses evaluate to `true`. ◀

A.3 Proof of Theorem 5

▶ **Theorem 5.** (⋆) *ONE2ALL ENUMERATION can be solved with $O(m^2\tau)$ delay.*

Proof. Our binary partition constructs each spanner incrementally by including (or excluding) an edge, one at a time. Start by considering the partial spanner P containing only vertex s. Let this state be the root node of our binary partition tree. Then, while an edge e exists that can be added to P such that the resulting spanner would remain connected, add two nodes to the binary partition tree connected to the current state node, one representing the inclusion of e to P, and the other representing the exclusion by removal from $\mathcal{G}$. For the former we apply the algorithm from Lemma 4 for the ONE2ALL CONNECTED SPANNER EXTENSION instance with $E'' = E(P) \cup e$. If the instance is positive, include e in to P, and repeat the for loop on this state node of the binary partition tree. If the instance is negative, then backtrack in the partition tree. For the latter, check if all vertices are still reachable from s in $\mathcal{G} \setminus e$ (using TFS for example). If all vertices are still reachable, remove e from $\mathcal{G}$ and repeat the for loop at this state node in the partition tree. If not, then backtrack up the tree. After adding n edges, P is a spanner, and we return it and backtrack.

This enumerates all ONE2ALL spanners, since any such a spanner can be formed by adding edges while staying connected, and removing edges from the graph. Since each spanner corresponds to some leaf node of the tree, which has a depth of $O(m)$, and progressing from one node to another node one layer down takes time $O(m\tau)$ (independently from including or excluding the edge), we obtain that the delay is bounded by $O(m^2\tau)$. ◀

A.4 Proof of Theorem 7

▶ **Theorem 7.** (⋆) *ONE2ALL2ONE ENUMERATION is* Dual*-hard, even if the lifetime is constant.*

Proof. We reduce HYPERGRAPH DUALIZATION, or Dual, to our problem. Let us first describe the transformation of a Dual instance into a ONE2ALL2ONE spanner enumeration instance. Then, we prove that each minimal transversal in the Dual instance corresponds to a ONE2ALL2ONE spanner in the ONE2ALL2ONE spanner enumeration instance ($\Longrightarrow$), and vice versa ($\Longleftarrow$).

For the following transformation, refer to Figure 2 as well. Let the Dual instance G be composed of n vertices u_i and m hyperedges h_j. Start by constructing an initially empty temporal graph $\mathcal{G}$ and add vertex v_{u_i} for each vertex $u_i \in G$, and vertex v_{h_j} for each hyperedge $h_j \in G$. For all i, j, if $u_i \in h_j$, then create a vertex v_{u_i,h_j} in $\mathcal{G}$ and edges $\{v_{u_i}, v_{u_i,h_j}\}$ and $\{v_{u_i,h_j}, v_{h_j}\}$ with times $\{3, 4\}$ and $\{1, 4\}$ resp. Add vertex s to $\mathcal{G}$. For each v_{u_i}, add vertices $v^0_{u_i}$, $v^1_{u_i}$, and $v^2_{u_i}$, and edges $\{s, v^0_{u_i}\}$, $\{v^0_{u_i}, v_{u_i}\}$ with times 1 and 2 resp. as well as edges $\{s, v^1_{u_i}\}$, $\{v^1_{u_i}, v^2_{u_i}\}$, and $\{v^2_{u_i}, v_{u_i}\}$ with times $\{1, 3\}$, 2, and $\{1, 3\}$ resp. Finally, add vertex t and edges $\{v_{h_j}, t\}$ with time 2 for all h_j, and edge $\{t, s\}$ with time 3. The transformation is now complete.

By construction, we obtain that all edges except for edges $\{v^0_{u_i}, v_{u_i}\}$ are necessary, i.e. that they are part of any ONE2ALL2ONE spanner of $\mathcal{G}$ w.r.t. source vertex s. Indeed, we have that:

- Each edge $\{s, v^0_{u_i}\}$ is necessary for s to reach vertex $v^0_{u_i}$;
- Each edge $\{s, v^1_{u_i}\}$ is necessary for vertex v_{u_i} to reach s;
- And likewise for each edge $\{v^1_{u_i}, v^2_{u_i}\}$ and each edge $\{v^2_{u_i}, v_{u_i}\}$;
- Each edge $\{v_{u_i}, v_{u_i,h_j}\}$ is necessary for s to reach vertex v_{u_i,h_j};
- Each edge $\{v_{u_i,h_j}, v_{h_j}\}$ is necessary for vertex v_{u_i,h_j} to reach s;
- Each edge $\{v_{h_j}, t\}$ is necessary for vertex v_{h_j} to reach s;
- And finally, edge $\{t, s\}$ is necessary for t to reach s.

Let $\mathcal{G}'$ denote the temporal graph composed of these edges, i.e. $\mathcal{G}' = \mathcal{G} \setminus \{v^0_{u_i}, v_{u_i}\}$, for all u_i (in Figure 2, this corresponds to the illustrated temporal graph without the dashed edges). We have that in $\mathcal{G}'$, all vertices can be reached by s as well as reach s, except for vertices v_{h_j}, as these can only reach s but not be reached by s. Note that adding some edge $\{v^0_{u_i}, v_{u_i}\}$ to $\mathcal{G}'$ will allow s to reach exactly vertices v_{h_j} such that $u_i \in h_j$. We thus observe that any spanner of $\mathcal{G}$ is composed of $\mathcal{G}'$ with some minimal selection of edges $\{v^0_{u_i}, v_{u_i}\}$.

minimal transversal $\implies$ ONE2ALL2ONE spanner:
Given a minimal transversal T of the Dual instance, we can construct the following ONE2ALL2ONE spanner S. Consider S to be initialized as $\mathcal{G}'$ by the above observation, and then add, for each $u_i \in T$, edge $\{v^0_{u_i}, v_{u_i}\}$ to S. Since T is a minimal transversal for the Dual instance, we have that for every hyperedge h_j, there exists some $u_i \in T$ that hits it. In the same manner, we have that by adding the corresponding edges $\{v^0_{u_i}, v_{u_i}\}$ to S, we obtain a ONE2ALL2ONE spanner such that for every vertex v_{h_j}, there exists some added edge $\{v^0_{u_i}, v_{u_i}\}$ that allows s to reach v_{h_j}. S is minimal because no edge from $\mathcal{G}'$ can be removed, and no edge $\{v^0_{u_i}, v_{u_i}\}$ can be removed either since otherwise T wouldn't be minimal.

minimal transversal $\impliedby$ ONE2ALL2ONE spanner:
Consider a ONE2ALL2ONE spanner S in the transformed instance. It must be composed of $\mathcal{G}'$ with some edges $\{v^0_{u_i}, v_{u_i}\}$. We construct a minimal transversal for the Dual instance T by selecting each vertex $u_i \in G$ such that $\{v^0_{u_i}, v_{u_i}\} \in S$. Indeed, this must correspond to a transversal since by construction, edges $\{v^0_{u_i}, v_{u_i}\}$ allow for s to reach exactly vertices v_{h_j} such that $u_i \in h_j$. T is a minimal transversal since, by contradiction, if some selected u_i could be removed and the result is still a transversal, then the corresponding edge $\{v^0_{u_i}, v_{u_i}\}$ in S could be removed as well, which is a contradiction because S is minimal. ◀

A.5 Proof of Theorem 8

▶ **Theorem 8.** (⋆) *MANY2ALL CONNECTED EXTENSION PROBLEM is* NP*-complete, even if the lifetime is constant and the number of source vertices is 2.*

Proof. To prove NP-hardness, consider the same transformation as in Theorem 3, from SAT (see again Figure 1). Let s be the first source vertex, renamed to s_1. Let a second source vertex s_2 be added to the temporal graph, and connect it to s_1 with an edge of time 3. Further connect s_2 to all other vertices v_z for all literals and clauses z, with an edge which is then subdivided into edges $\{s_2, v_{s_2,z}\}$ and $\{v_{s_2,z}, v_z\}$ with times 2 and 3 resp. Add to edge set E'' (which already contains edges $\{v_{x_i}, v_{\bar{x_i}}\}$ for all variables x_i) all these newly created edges. Note that E'' is connected. The transformation is now complete.

Since edge $\{s_1, s_2\}$ is in E'' which allows the two source vertices to reach each other, and since all edges $\{s_2, v_{s_2,z}\}$, $\{v_{s_2,z}, v_z\}$ are in E'', and they allow source vertex s_2 to reach all vertices v_z, and these edges do not allow s_1 to reach vertices v_z (nor aid in the reachability between vertices v_z that could indirectly help s_1 to reach them), essentially what remains to decide in this reduction is how to connect from s_1 to the other vertices. This is exactly the

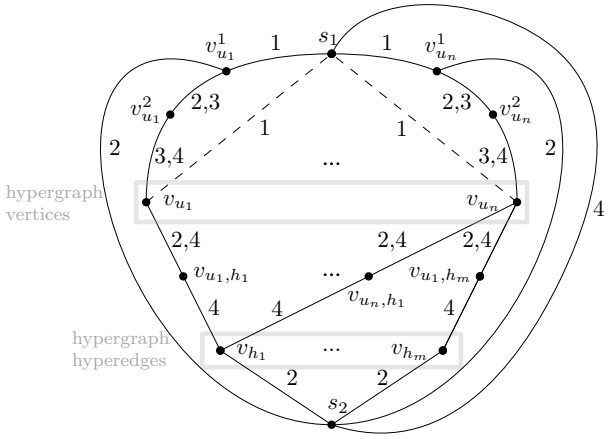


Figure 4 Illustration of the transformation of a Dual instance to a MANY2ALL spanner enumeration instance. All solid edges are necessary in all spanners, while specific subsets of dashed edges determine specific spanners. Note that in this specific illustration, hyperedge h_1 contains (at least) vertices u_1 and u_n, and hyperedge h_m contains (at least) vertex u_n.

problem in Theorem 3, and we use the exact same structure for this (ignoring the new edges we added which we just argued are not useful for this). Hence we conclude that the result, being the NP-completeness, from Theorem 3 transfers for our problem here. ◀

A.6 Proof of Theorem 9

▶ **Theorem 9.** (⋆) *MANY2ALL ENUMERATION is* Dual*-hard, even if the lifetime is constant, and the number of source vertices is 2.*

Proof. We reduce HYPERGRAPH DUALIZATION, or Dual, to our problem. Let us first describe the transformation of a Dual instance into a MANY2ALL spanner enumeration instance. Then, we prove that each minimal transversal in the Dual instance corresponds to a MANY2ALL spanner in the MANY2ALL spanner enumeration instance ($\Longrightarrow$), and vice versa ($\Longleftarrow$).

For the following transformation, refer to Figure 4 as well. Let the Dual instance G be composed of n vertices u_i and m hyperedges h_j. Start by constructing an initially empty temporal graph $\mathcal{G}$ and add vertex v_{u_i} for each vertex $u_i \in G$, and vertex v_{h_j} for each hyperedge $h_j \in G$. For all i, j, if $u_i \in h_j$, then create a vertex v_{u_i,h_j} in $\mathcal{G}$ and edges $\{v_{u_i}, v_{u_i,h_j}\}$ and $\{v_{u_i,h_j}, v_{h_j}\}$ with times $\{2, 4\}$ and 4 resp. Add vertex s_1 to $\mathcal{G}$. For each v_{u_i}, add vertices $v^1_{u_i}$, and $v^2_{u_i}$, and edges $\{s_1, v^1_{u_i}\}$, $\{v^1_{u_i}, v^2_{u_i}\}$, and $\{v^2_{u_i}, v_{u_i}\}$ with times 1, $\{2, 3\}$, and $\{3, 4\}$ resp. Finally, add vertex s_2, edge $\{s_2, s_1\}$ with time 4, edges $\{s_2, v_{h_j}\}$ and $\{s_2, v^1_{u_i}\}$ with time 2 for all h_j and u_i. The transformation is now complete.

By construction, we obtain that all edges except for edges $\{s_1, v_{u_i}\}$ are necessary, i.e. that they are part of any MANY2ALL spanner of $\mathcal{G}$ w.r.t. source vertices s_1 and s_2. Indeed, we have that:

- Each edge $\{s_1, v^1_{u_i}\}$ is necessary for s_1 to reach vertex $v^1_{u_i}$;
- Each edge $\{v^1_{u_i}, v^2_{u_i}\}$ is necessary for s_2 to reach vertex $v^2_{u_i}$;
- Each edge $\{v^2_{u_i}, v_{u_i}\}$ is necessary for s_2 to reach vertex v_{u_i};
- Each edge $\{v_{u_i}, v_{u_i,h_j}\}$ is necessary for s_1 to reach vertex v_{u_i,h_j};
- Each edge $\{v_{u_i,h_j}, v_{h_j}\}$ is necessary for s_2 to reach vertex v_{u_i,h_j};
- Each edge $\{s_2, v^1_{u_i}\}$ is necessary for s_2 to reach vertex $v^1_{u_i}$;
- Each edge $\{s_2, v_{h_j}\}$ is necessary for s_2 to reach vertex v_{h_j};
- And finally, edge $\{s_2, s_1\}$ is necessary for s_2 to reach s_1.

Let $\mathcal{G}'$ denote the temporal graph composed of these edges, i.e. $\mathcal{G}' = \mathcal{G} \setminus \{s_1, v_{u_i}\}$, for all u_i (in Figure 4, this corresponds to the illustrated temporal graph without the dashed edges). We have that in $\mathcal{G}'$, all vertices can be reached by s_1 as well as by s_2, except for vertices v_{h_j}, as these can only be reached by s_2 but not by s_1. Note that adding some edge $\{s_1, v_{u_i}\}$ to $\mathcal{G}'$ will allow s_1 to reach exactly vertices v_{h_j} such that $u_i \in h_j$. We thus observe that any spanner of $\mathcal{G}$ is composed of $\mathcal{G}'$ with some minimal selection of edges $\{s_1, v_{u_i}\}$.

minimal transversal $\implies$ MANY2ALL spanner:
Given a minimal transversal T of the Dual instance, we can construct the following MANY2ALL spanner S. Consider S to be initialized as $\mathcal{G}'$ by the above observation, and then add, for each $u_i \in T$, edge $\{s_1, v_{u_i}\}$ to S. Since T is a minimal transversal for the Dual instance, we have that for every hyperedge h_j, there exists some $u_i \in T$ that hits it. In the same manner, we have that by adding the corresponding edges $\{s_1, v_{u_i}\}$ to S, we obtain a MANY2ALL spanner such that for every vertex v_{h_j}, there exists some added edge $\{s_1, v_{u_i}\}$ that allows s_1 to reach v_{h_j}. S is minimal because no edge from $\mathcal{G}'$ can be removed, and no edge $\{s_1, v_{u_i}\}$ can be removed either since otherwise T wouldn't be minimal.

minimal transversal $\impliedby$ MANY2ALL spanner:
Consider a MANY2ALL spanner S in the transformed instance. It must be composed of $\mathcal{G}'$ with some edges $\{s_1, v_{u_i}\}$. We construct a minimal transversal for the Dual instance T by selecting each vertex $u_i \in G$ such that $\{s_1, v_{u_i}\} \in S$. Indeed, this must correspond to a transversal since by construction, edges $\{s_1, v_{u_i}\}$ allow for s_1 to reach exactly vertices v_{h_j} such that $u_i \in h_j$. T is a minimal transversal since, by contradiction, if some selected u_i could be removed and the result is still a transversal, then the corresponding edge $\{s_1, v_{u_i}\}$ in S could be removed as well, which is a contradiction because S is minimal. ◀

A.7 Proof of Lemma 10

▶ **Lemma 10.** (⋆) *Temporal graph $\mathcal{G}$ has the following structural properties:*

- ***Reachability:*** *In $\mathcal{G}$, all vertices can reach each other, while in $\mathcal{G}'$, all vertices can reach each other except for vertices v_{C_i} which do not reach corresponding vertices $v_{C'_i}$, for all i;*
- ***Necessity:*** *In $\mathcal{G}$ and in $\mathcal{G}'$, all edges of $\mathcal{G}'$ are necessary for some vertex pair reachability;*
- ***Spanner:*** *Any spanner of $\mathcal{G}$ must be composed of $\mathcal{G}'$ and some minimal edge set of $\mathcal{G}''$, and adding to $\mathcal{G}'$ edges $\{v_y, v_{x_i}\}$ and $\{v_{x_i}, v_{y'}\}$, for any i, results in a spanner.*

Proof. **Reachability:** Let us first study reachability of vertices in $\mathcal{G}'$. All vertices can trivially reach all their neighbors, so we consider the non-adjacent vertices only and give the corresponding temporal paths (which always use the earliest times possible):

- Vertex v_z: $\forall i, j, i \neq j$:
  $v_z \rightsquigarrow v_{C_i} \rightsquigarrow v_y$,
  $v_z \rightsquigarrow v_{C'_i} \rightsquigarrow v_{y'}$,
  $v_z \rightsquigarrow v_{C'_j} \rightsquigarrow v_{C_{i,j}}$;
- Any vertex v_{x_i}: $\forall h, j$:
  $v_{x_i} \rightsquigarrow v_z \rightsquigarrow v_{x_j}$,
  $v_{x_i} \rightsquigarrow v_z \rightsquigarrow v_{C_j}$,
  $v_{x_i} \rightsquigarrow v_z \rightsquigarrow v_{C'_j}$,
  $v_{x_i} \rightsquigarrow v_z \rightsquigarrow v_{C_j} \rightsquigarrow v_y$,
  $v_{x_i} \rightsquigarrow v_z \rightsquigarrow v_{C'_j} \rightsquigarrow v_{y'}$,
  $v_{x_i} \rightsquigarrow v_z \rightsquigarrow v_{C'_j} \rightsquigarrow v_{C_{h,j}}$;
- Any vertex v_{C_i}: $\forall h, j \neq i$:
  $v_{C_i} \rightsquigarrow v_z \rightsquigarrow v_{x_j}$,

$v_{C_i} \rightsquigarrow v_{C_{i,j}} \rightsquigarrow v_{C_{h,j}}$,
$v_{C_i} \rightsquigarrow v_{C_{i,j}} \rightsquigarrow v_{C_{j,i}}$,
$v_{C_i} \rightsquigarrow v_{C_{i,j}} \rightsquigarrow v_{C'_j}$,
$v_{C_i} \rightsquigarrow v_{C_{i,j}} \rightsquigarrow v_{C'_j} \rightsquigarrow v_{y'}$;

- Vertex v_y: $\forall h, i, j \neq i$:
  $v_y \rightsquigarrow v_{C_i} \rightsquigarrow v_z \rightsquigarrow v_{x_j}$,
  $v_y \rightsquigarrow v_{C_i} \rightsquigarrow v_{C_{i,j}} \rightsquigarrow v_{C_{h,j}}$,
  $v_y \rightsquigarrow v_{C_i} \rightsquigarrow v_{C_{i,j}} \rightsquigarrow v_{C_{j,i}}$,
  $v_y \rightsquigarrow v_{C_i} \rightsquigarrow v_{C_{i,j}} \rightsquigarrow v_{C'_j} \rightsquigarrow v_{y'}$;
- Any vertex $v_{C'_i}$: $\forall h, j$:
  $v_{C'_i} \rightsquigarrow v_z \rightsquigarrow v_{x_j}$,
  $v_{C'_i} \rightsquigarrow v_z \rightsquigarrow v_{C_j} \rightsquigarrow v_y$,
  $v_{C'_i} \rightsquigarrow v_{C'_j} \rightsquigarrow v_{C_{h,j}}$;
- Vertex $v_{y'}$: $\forall h, i, j$:
  $v_{y'} \rightsquigarrow v_{C'_i} \rightsquigarrow v_z \rightsquigarrow v_{x_j}$,
  $v_{y'} \rightsquigarrow v_{C'_i} \rightsquigarrow v_z \rightsquigarrow v_{C_j} \rightsquigarrow v_y$,
  $v_{y'} \rightsquigarrow v_{C'_j} \rightsquigarrow v_{C_{i,j}}$;
- Any vertex $v_{C_{i,j}}$: $\forall h, k$:
  $v_{C_{i,j}} \rightsquigarrow v_{C_{h,j}} \rightsquigarrow v_{C_{j,h}} \rightsquigarrow v_{C_{k,h}}$,
  $v_{C_{i,j}} \rightsquigarrow v_{C_{h,j}} \rightsquigarrow v_{C_{j,h}} \rightsquigarrow v_{C'_h}$,
  $v_{C_{i,j}} \rightsquigarrow v_{C'_j} \rightsquigarrow v_{y'}$,
  $v_{C_{i,j}} \rightsquigarrow v_{C_i} \rightsquigarrow v_z \rightsquigarrow v_{C_h}$,
  $v_{C_{i,j}} \rightsquigarrow v_{C_i} \rightsquigarrow v_y$,
  $v_{C_{i,j}} \rightsquigarrow v_{C_i} \rightsquigarrow v_z \rightsquigarrow v_{x_h}$.

This shows that all vertices can reach all others in $\mathcal{G}'$ with the exception of vertex v_{C_i} not being able to reach the corresponding vertex $v_{C'_i}$, for all i. Note that adding edges from $\mathcal{G}''$ can only improve reachability, and adding all edges from $\mathcal{G}''$ results in all vertices reaching all vertices in $\mathcal{G}$, because now temporal path $v_{C_i} \rightsquigarrow v_y \rightsquigarrow v_{x_j} \rightsquigarrow v_{y'} \rightsquigarrow v_{C'_i}$, for any j, exists.

Necessity: Next, let us show that all edges of $\mathcal{G}'$ are necessary in $\mathcal{G}'$, and in $\mathcal{G}$, for the reachability of some pair of vertices. Let us list all these edges and the pairs they are necessary for (in both temporal graphs):

- Any edge $\{v_z, v_{C_i}\}$: necessary for $v_{C'_i}$ to reach v_{C_i};
- Any edge $\{v_z, v_{C'_i}\}$: necessary for $v_{C'_i}$ to reach v_{C_i};
- Any edge $\{v_z, v_{x_i}\}$: necessary for v_{x_i} to reach v_z;
- Any edge $\{v_{C_i}, v_y\}$: necessary for v_{C_i} to reach v_y;
- Any edge $\{v_{C'_i}, v_{y'}\}$: necessary for $v_{y'}$ to reach $v_{C'_i}$;
- Any edge $\{v_{C_i}, v_{C_{i,j}}\}$: necessary for v_{C_i} to reach $v_{C_{i,j}}$;
- Any edge $\{v_{C'_j}, v_{C_{i,j}}\}$: necessary for $v_{C'_j}$ to reach $v_{C_{i,j}}$;
- Any edge $\{v_{C_{i,j}}, v_{C_{j,i}}\}$: necessary for $v_{C_{i,j}}$ to reach $v_{C_{j,i}}$;

Spanner: By the necessity property, all edges of $\mathcal{G}'$ must be part of any spanner of $\mathcal{G}$. By the reachability property, these ensure reachability between almost all vertices. They are however insufficient for any v_{C_i} to reach the corresponding $v_{C'_i}$. Some edges from $\mathcal{G}''$ are thus required for this in a spanner. Moreover, just adding the two edges $\{v_y, v_{x_i}\}$ and $\{v_{x_i}, v'_y\}$ for some i is sufficient, as now temporal path $v_{C_j} \rightsquigarrow v_y \rightsquigarrow v_{x_i} \rightsquigarrow v_{y'} \rightsquigarrow v_{C'_j}$ exists for all j, and this is a minimal spanner because none of these two edges can be removed without breaking these temporal paths. ◀