Enumerating Spanners in Directed Temporal Graphs

Lapo Cioni[1], Andrea Marino[1],
Jason Schoeters[2], and Takeaki Uno[3]

[1]Università di Firenze, Italy
[2]Deepform, United Kingdom
[3]National Institute of Informathics, Tokyo, Japan

July 6, 2026
Algorithmic Aspects of Temporal Graphs IX
Satellite workshop of ICALP 2026,
Egham, UK

Introduction

It has been published here:

Lapo Cioni, Andrea Marino, Jason Schoeters, Takeaki Uno:
Enumerating Spanners in Directed Temporal Graphs., IWOCA 2026: 204-219

Maybe you have already seen the presentation of Lapo at IWOCA or at Dagstuhl last month. These are almost the same slides (Thanks Lapo!).

Inspired by a previous work on undirected spanners.

Kazuhiro Kurita, Andrea Marino, Jason Schoeters, Takeaki Uno:
Spanner Enumeration for Temporal Graphs., SAND 2025: 9:1-9:21

Enumeration / Exhaustive Generation

Generate **all** feasible substructures (e.g., spanning trees of a graph).

Typical complexity guarantees:

Enumeration / Exhaustive Generation

Generate **all** feasible substructures (e.g., spanning trees of a graph).

Typical complexity guarantees:

- **Polynomial delay**: the time between any two successive outputs is bounded by a polynomial in the input size. ☺

Enumeration / Exhaustive Generation

Generate **all** feasible substructures (e.g., spanning trees of a graph).

Typical complexity guarantees:

- **Polynomial delay**: the time between any two successive outputs is bounded by a polynomial in the input size. ☺
- **Output-polynomial**: the total running time is polynomial in both the input size n and the number of solutions α, ☺
  - e.g., $O(n^c \alpha^k)$ for fixed constants c, k.
  - Polynomial delay implies output-polynomial time, namely $O(n^c \alpha)$.

Enumeration / Exhaustive Generation

Generate **all** feasible substructures (e.g., spanning trees of a graph).

Typical complexity guarantees:

- **Polynomial delay**: the time between any two successive outputs is bounded by a polynomial in the input size. ☺
- **Output-polynomial**: the total running time is polynomial in both the input size n and the number of solutions α, ☺
  - e.g., $O(n^c \alpha^k)$ for fixed constants c, k.
  - Polynomial delay implies output-polynomial time, namely $O(n^c \alpha)$.
- **No output-polynomial algorithm** exists unless $P = NP$. ☹

Enumeration / Exhaustive Generation

Generate **all** feasible substructures (e.g., spanning trees of a graph).

Typical complexity guarantees:

- **Polynomial delay**: the time between any two successive outputs is bounded by a polynomial in the input size. ☺
- **Output-polynomial**: the total running time is polynomial in both the input size n and the number of solutions α, ☺
  - e.g., $O(n^c \alpha^k)$ for fixed constants c, k.
  - Polynomial delay implies output-polynomial time, namely $O(n^c \alpha)$.
- **No output-polynomial algorithm** exists unless $P = NP$. ☹
- DUAL-hard: at least as hard as the famous HYPERGRAPH DUALIZATION problem (enumerating minimal hitting sets), for which no output-polynomial algorithm is known. ?

Enumeration / Exhaustive Generation

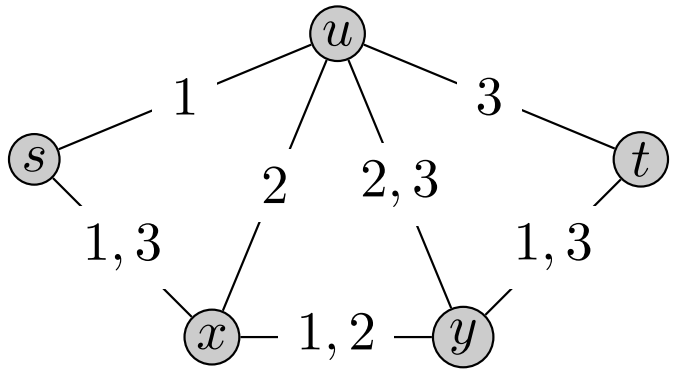
Generate **all** feasible substructures (e.g., spanning trees of a graph).

Typical complexity guarantees:

- **Polynomial delay**: the time between any two successive outputs is bounded by a polynomial in the input size. ☺
- **Output-polynomial**: the total running time is polynomial in both the input size n and the number of solutions α, ☺
  - e.g., $O(n^c \alpha^k)$ for fixed constants c, k.
  - Polynomial delay implies output-polynomial time, namely $O(n^c \alpha)$.
- **No output-polynomial algorithm** exists unless $P = NP$. ☹
- DUAL-hard: at least as hard as the famous HYPERGRAPH DUALIZATION problem (enumerating minimal hitting sets), for which no output-polynomial algorithm is known. ?

In static directed graphs, spanning trees can be listed in $O(n + m + \alpha)$, using gray codes [Gawrychowski and Knapik, 2026].

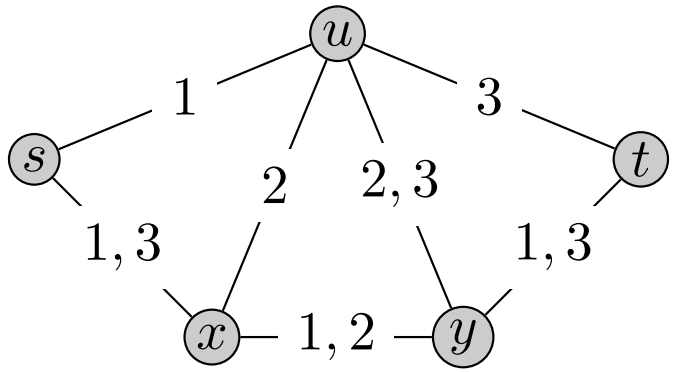
Temporal Graph



Definitions: Temporal Graph

- A *temporal graph* is a pair (G, λ) with G a simple graph, and $\lambda : E(G) \to \mathcal{P}(\mathbb{N})$ a function assigning time labels to edges;

Temporal Graph

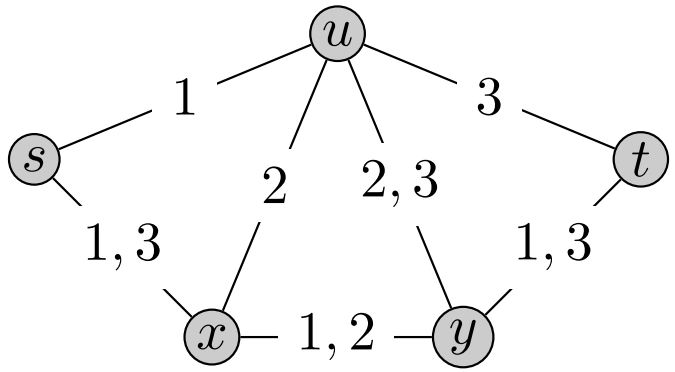


$\tau = 3$

Definitions: Lifetime

- A *temporal graph* is a pair (G, λ) with G a simple graph, and $\lambda : E(G) \to \mathcal{P}(\mathbb{N})$ a function assigning time labels to edges;
- the maximum value τ appearing among time labels is called *lifetime*;

Temporal Graph

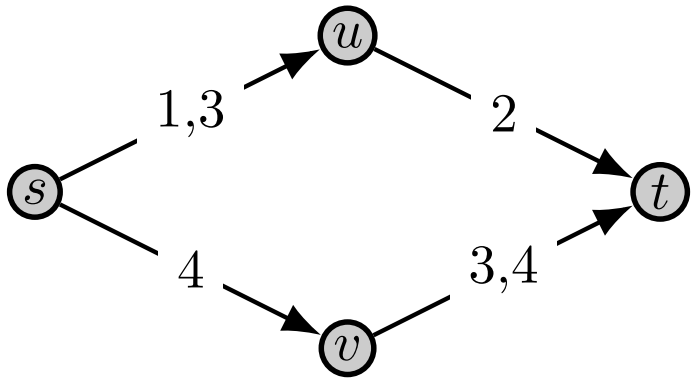


$\tau = 3$ $\ell = 2$

Definitions: Temporality

- A *temporal graph* is a pair (G, λ) with G a simple graph, and $\lambda : E(G) \to \mathcal{P}(\mathbb{N})$ a function assigning time labels to edges;
- the maximum value τ appearing among time labels is called *lifetime*;
- the maximum amount ℓ of times on labels, $\max_{e \in E(G)} |\lambda(e)|$, is called *temporality*.

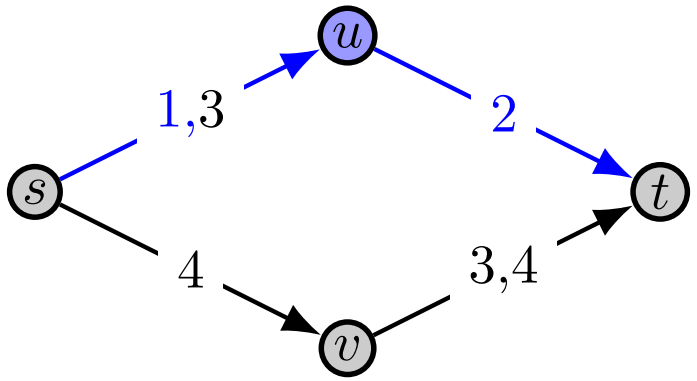
Connectivity in temporal graphs



Strict Model: Valid walks are the ones whose labels are *strictly increasing*.

Application: Strict Model can model for instance connections made by train. In a time instant, we can be only in one place or only on one train.

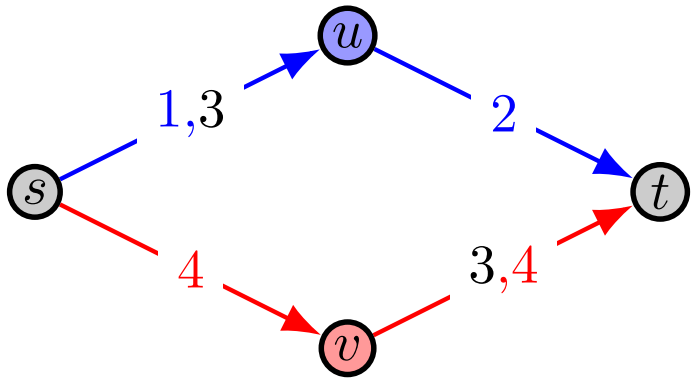
Connectivity in temporal graphs



Strict Model: Valid walks are the ones whose labels are *strictly increasing*.

Application: Strict Model can model for instance connections made by train. In a time instant, we can be only in one place or only on one train.

Connectivity in temporal graphs



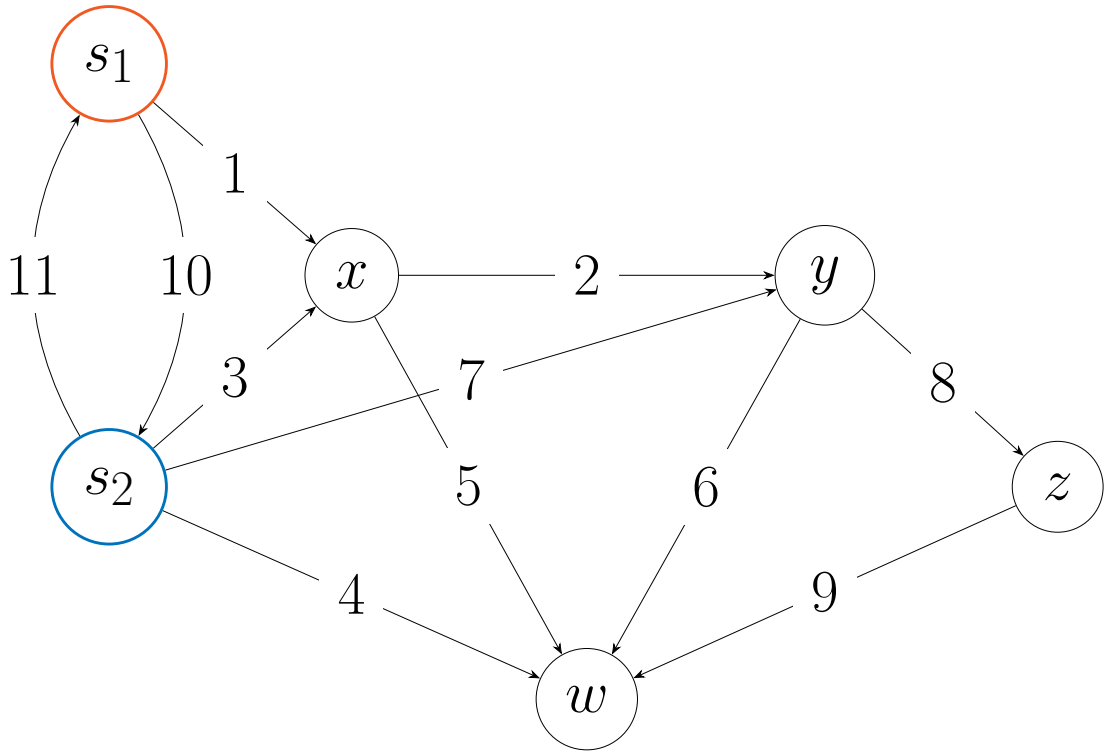
Strict Model: Valid walks are the ones whose labels are *strictly increasing*.
Application: Strict Model can model for instance connections made by train. In a time instant, we can be only in one place or only on one train.
Non-Strict Model: Valid walks are the ones whose labels are *non-strictly increasing*.
Application: Non-strict Model can represent which streets are available in a day: during a single day we can traverse many available streets.

Temporal spanners

Given:

- a directed temporal graph $\mathcal{G} = (G, \lambda)$;
- a subset $S = \{s_1, \ldots, s_k\} \subseteq V(G)$ of *sources*;

we say that $E' \subseteq E(G)$ is a *temporal spanner* for S if it is a **minimal** set of edges such that every source can reach every vertex in the graph using only edges in E'.

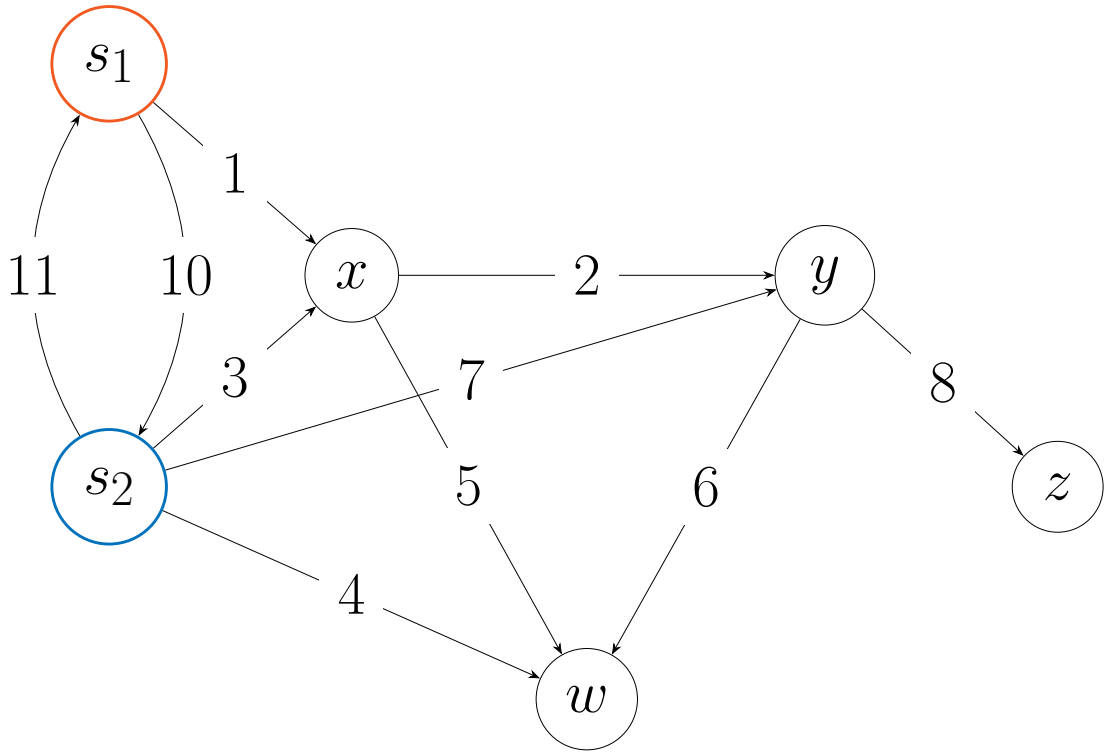


Temporal spanners

Given:

- a directed temporal graph $\mathcal{G} = (G, \lambda)$;
- a subset $S = \{s_1, \ldots, s_k\} \subseteq V(G)$ of *sources*;

we say that $E' \subseteq E(G)$ is a *temporal spanner* for S if it is a **minimal** set of edges such that every source can reach every vertex in the graph using only edges in E'.

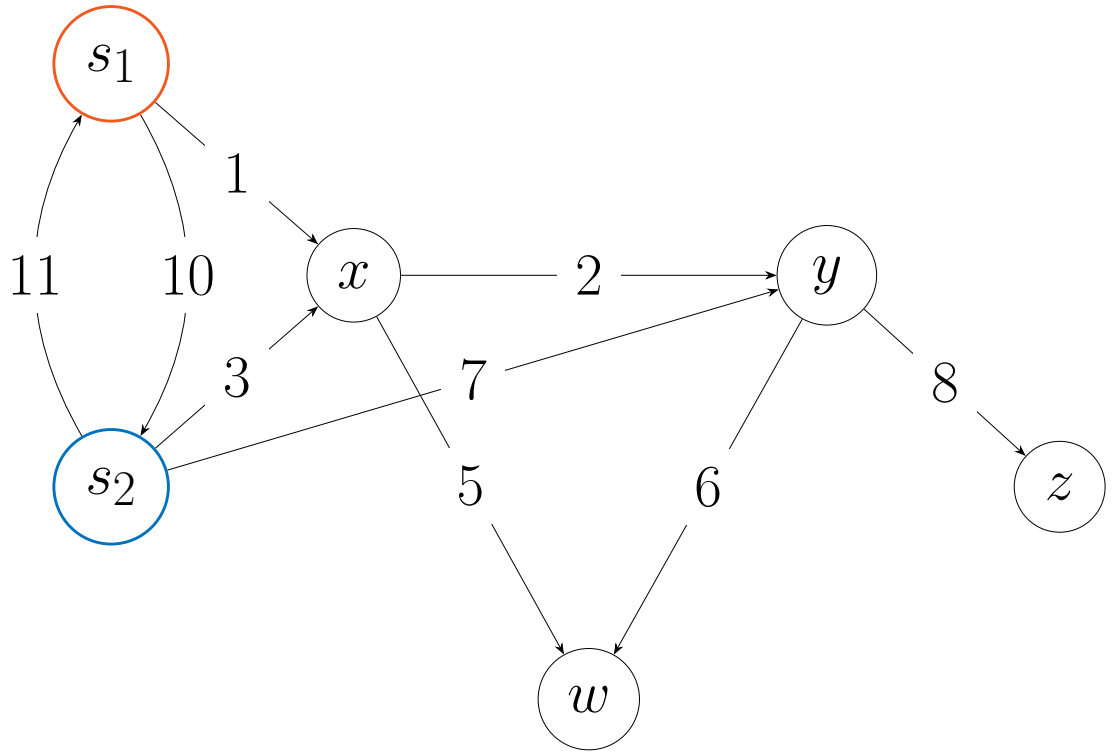


Temporal spanners

Given:

- a directed temporal graph $\mathcal{G} = (G, \lambda)$;
- a subset $S = \{s_1, \ldots, s_k\} \subseteq V(G)$ of *sources*;

we say that $E' \subseteq E(G)$ is a *temporal spanner* for S if it is a **minimal** set of edges such that every source can reach every vertex in the graph using only edges in E'.

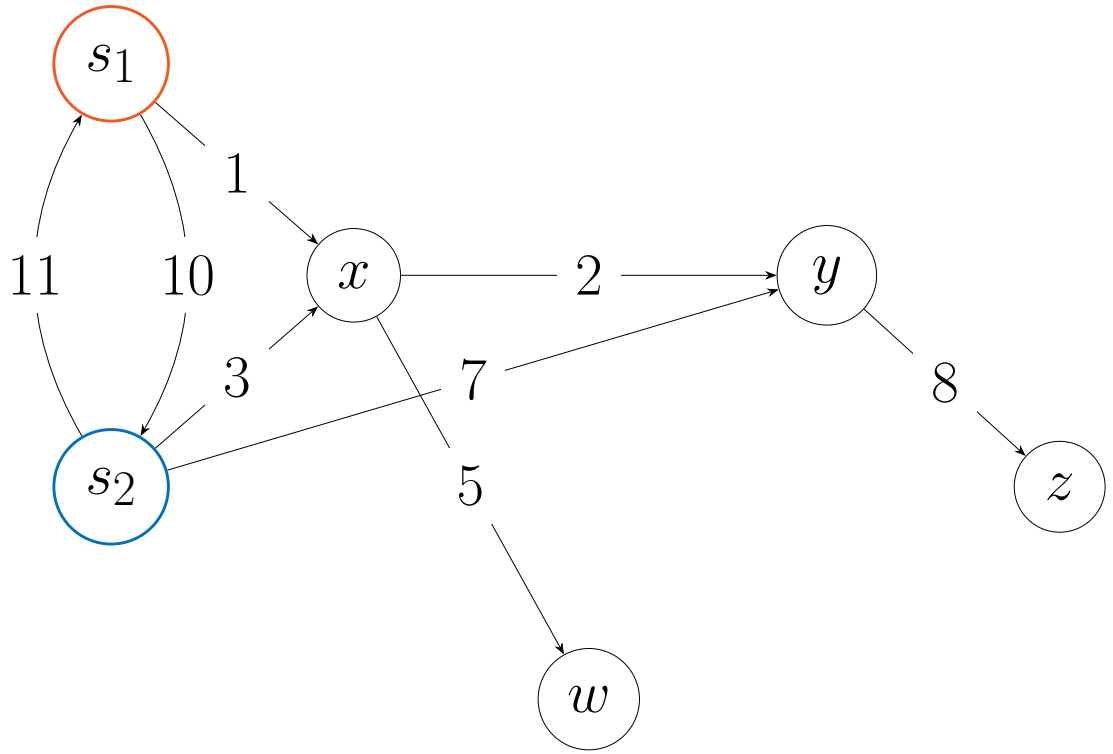


Temporal spanners

Given:

- a directed temporal graph $\mathcal{G} = (G, \lambda)$;
- a subset $S = \{s_1, \ldots, s_k\} \subseteq V(G)$ of *sources*;

we say that $E' \subseteq E(G)$ is a *temporal spanner* for S if it is a **minimal** set of edges such that every source can reach every vertex in the graph using only edges in E'.

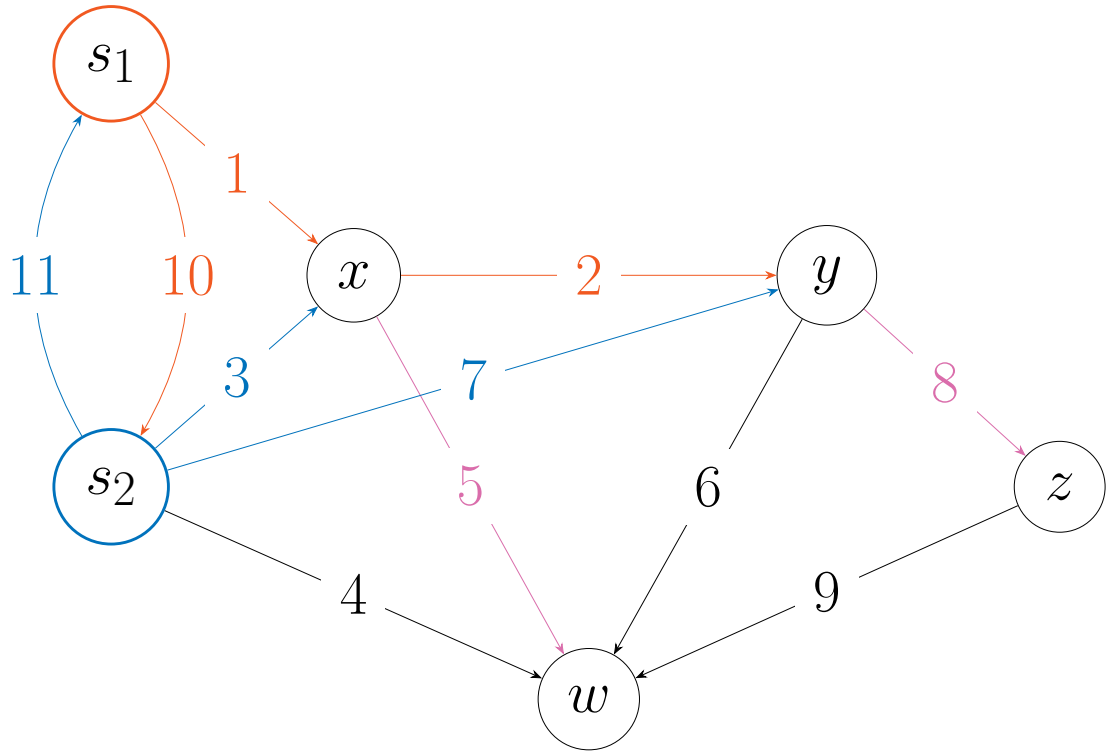


Temporal spanners

Given:

- a directed temporal graph $\mathcal{G} = (G, \lambda)$;
- a subset $S = \{s_1, \ldots, s_k\} \subseteq V(G)$ of *sources*;

we say that $E' \subseteq E(G)$ is a *temporal spanner* for S if it is a **minimal** set of edges such that every source can reach every vertex in the graph using only edges in E'.



(the colors represent which source uses that edge for reachability)

Temporal spanners

We are interested in minimal temporal spanners, which are spanners which are minimal with respect of set inclusion.
The name of the spanner depends on the number of sources.

- ONE2ALL: minimal spanner with one single source;
- ALL2ALL: minimal spanner where $S = V$, every vertex is also a source;
- MANY2ALL: minimal spanner with k sources, with $k \geq 2$.

Problems definitions

Remark: a minimal spanner can be found in polynomial time, independently on the amount of sources. What about finding **all** of them?

Problems definitions

Remark: a minimal spanner can be found in polynomial time, independently on the amount of sources. What about finding **all** of them?

For connectivity $X = \text{ONE2ALL}, \text{MANY2ALL}, \text{ALL2ALL}$, we consider two problems:

- Extension Problem: given a set E' of edges, does there exist at least one X temporal spanner of $\mathcal{G}$ containing E'?
- Enumeration: enumerate (i.e. generate exhaustively) all X spanners of $\mathcal{G}$.

If Extension Problem can be solved in polynomial time, then we can use a binary search (*flashlight method*) approach to enumerate all spanners in polynomial delay time.

Results

	Extension	Enumeration
one2all	Poly, even if partial sol. is not connected	Poly-delay
many2all	NP-c, even if $k = 2$, $\ell = 1$, and partial sol. is connected	Poly-delay if $k = 2$, and $\ell = 1$ *
		Dual-hard, even if $k = 3$, and $\ell = 2$
all2all	NP-c $\Leftarrow$	No output poly algorithm unless P = NP, even if $\ell = 1$

- τ: lifetime (maximum timestamp in the input temporal graph);
- k: number of sources in the temporal spanner;
- ℓ: temporality (maximum amount of timestamps on each edge).

All the negative results hold even when lifetime τ is constant.
All the results, except the starred * one, apply to both strict and non-strict models.

ONE2ALL - the easy case

	EXTENSION	ENUMERATION
ONE2ALL	Poly, even if partial sol. is not connected	Poly-delay
MANY2ALL	NP-c, even if $k = 2$, $\ell = 1$, and partial sol. is connected	Poly-delay if $k = 2$, and $\ell = 1$ *
		Dual-hard, even if $k = 3$, and $\ell = 2$
ALL2ALL	NP-c $\Leftarrow$	No output poly algorithm unless P = NP, even if $\ell = 1$

This case is similar to the analogous for undirected temporal spanners (SAND 2025), except that in the directed case the set needs not to be connected, differently from the undirected.
ENUMERATION is then solved by using EXTENSION PROBLEM.

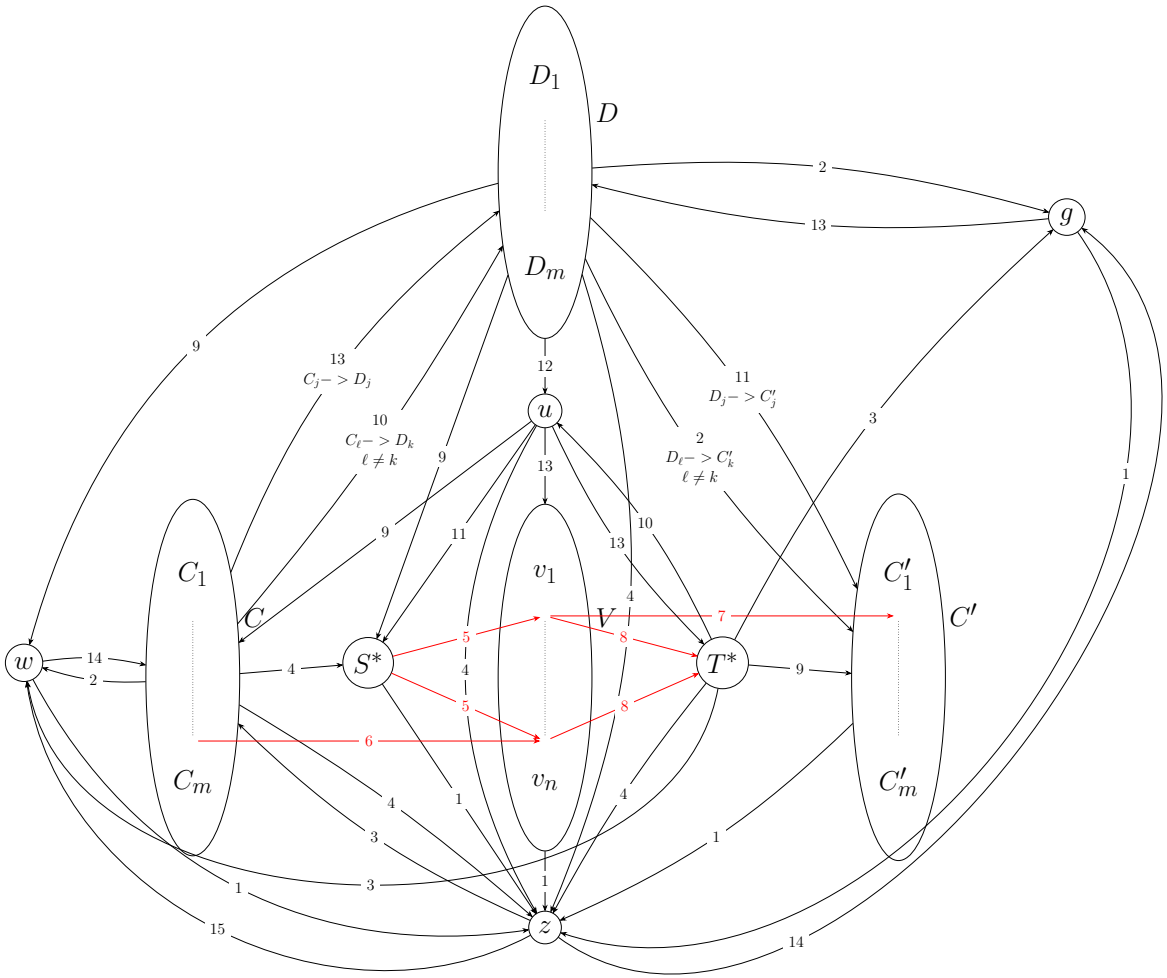
ALL2ALL

	EXTENSION	ENUMERATION
ONE2ALL	Poly, even if partial sol. is not connected	Poly-delay
MANY2ALL	NP-c, even if $k = 2$, $\ell = 1$, and partial sol. is connected	Poly-delay if $k = 2$, and $\ell = 1$
		Dual-hard, even if $k = 3$, and $\ell = 2$
ALL2ALL	NP-c $\Leftarrow$	No output poly algorithm unless P = NP, even if $\ell = 1$

Given a (partial) collection of ALL2ALL temporal directed spanners, it is NP-c to even know if there exists another ALL2ALL spanner.

ALL2ALL

The main idea is that there are some trivial spanners (exactly n), but anything after those exists if and only if a 3-SAT formula is solvable.
The red edges are part of a "logic" circuit, dependent on the formula.



all2all

The main idea is that there are some trivial spanners (exactly n), but anything after those exists if and only if a 3-SAT formula is solvable.

This implies that there is no output-sensitive algorithm unless P = NP.

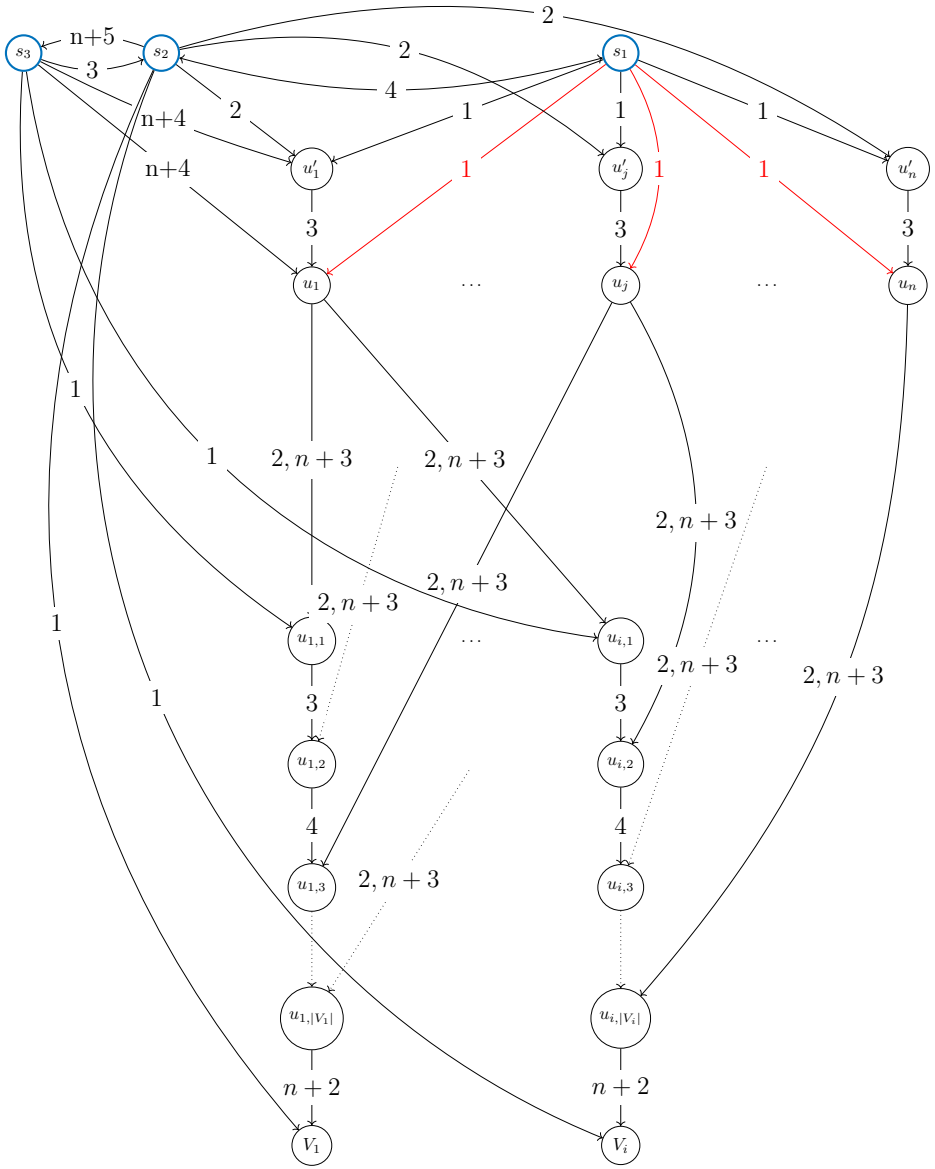
- Suppose there is one with guarantee $O(n^c \alpha^b)$.
- Let it run for $O(n^{c+b})$.
- If it is still running, it means there are at least $n+1$ solutions.
- If it ends, count the number of obtained solutions.

MANY2ALL

	EXTENSION	ENUMERATION
ONE2ALL	Poly, even if partial sol. is not connected	Poly-delay
MANY2ALL	NP-c, even if $k = 2$, $\ell = 1$, and partial sol. is connected	Poly-delay if $k = 2$, and $\ell = 1$
		Dual-hard, even if $k = 3$, and $\ell = 2$
ALL2ALL	NP-c $\Leftarrow$	No output poly algorithm unless P = NP, even if $\ell = 1$

For enumeration, the problem is reduced to HYPERGRAPH DUALIZATION, which is a known difficult enumeration problem. It can be seen as enumeration of minimal hitting sets (given collection of sets, a minimal hitting set is a set of elements of the universe hitting all the sets).

MANY2ALL - $k = 3$ and $\ell = 2$



Given a collection of sets, represent elements of the universe in the upper part and sets in the lower part. Connections in the middle represent membership

Every spanner contains all the black edges, so the set of selected red edges corresponds to a solution of HYPERGRAPH DUALIZATION (and vice versa).

The lifetime is constant in the non-strict model.

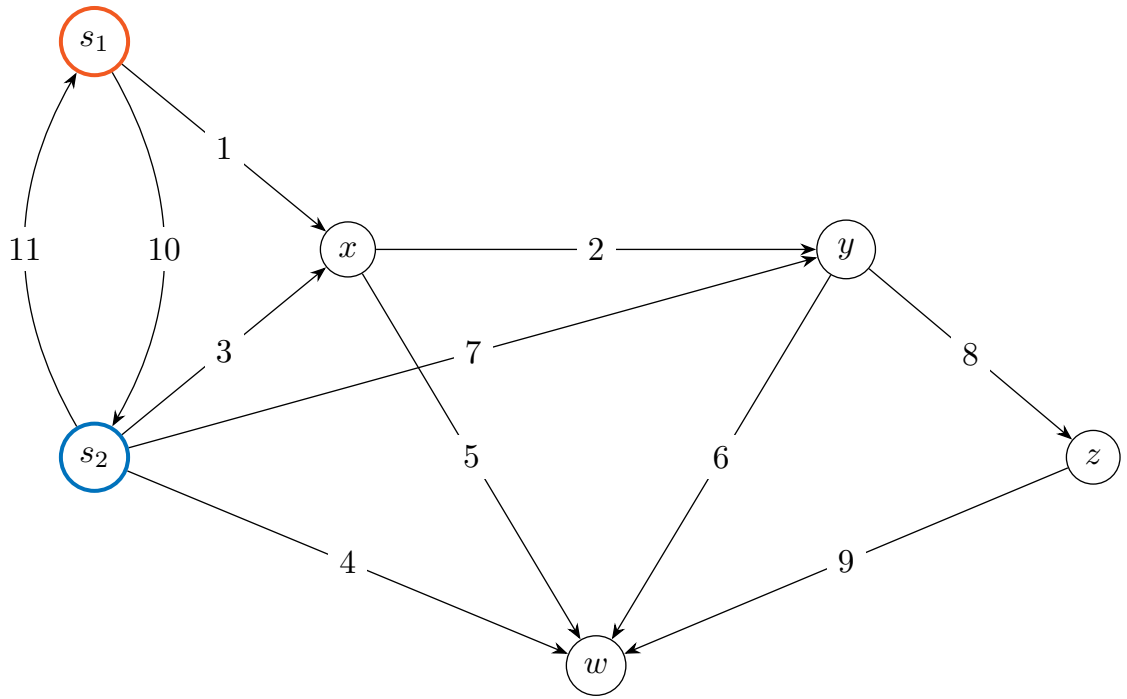
MANY2ALL - Polynomial Delay for $k = 2$ and $\ell = 1$

	EXTENSION	ENUMERATION
ONE2ALL	Poly, even if partial sol. is not connected	Poly-delay
MANY2ALL	NP-c, even if $k = 2$, $\ell = 1$, and partial sol. is connected	Poly-delay if $k = 2$, and $\ell = 1$
		Dual-hard, even if $k = 3$, and $\ell = 2$
ALL2ALL	NP-c $\Leftarrow$	No output poly algorithm unless P = NP, even if $\ell = 1$

For this result, we cannot use flashlight search. Instead, use the approach know as reverse search.
First, we define a parent-child relation between spanners. It is defined via a procedure to compute the parent of any spanner.
Then we reverse this by providing an algorithm that, for any given spanner, produces all its children in polynomial delay.

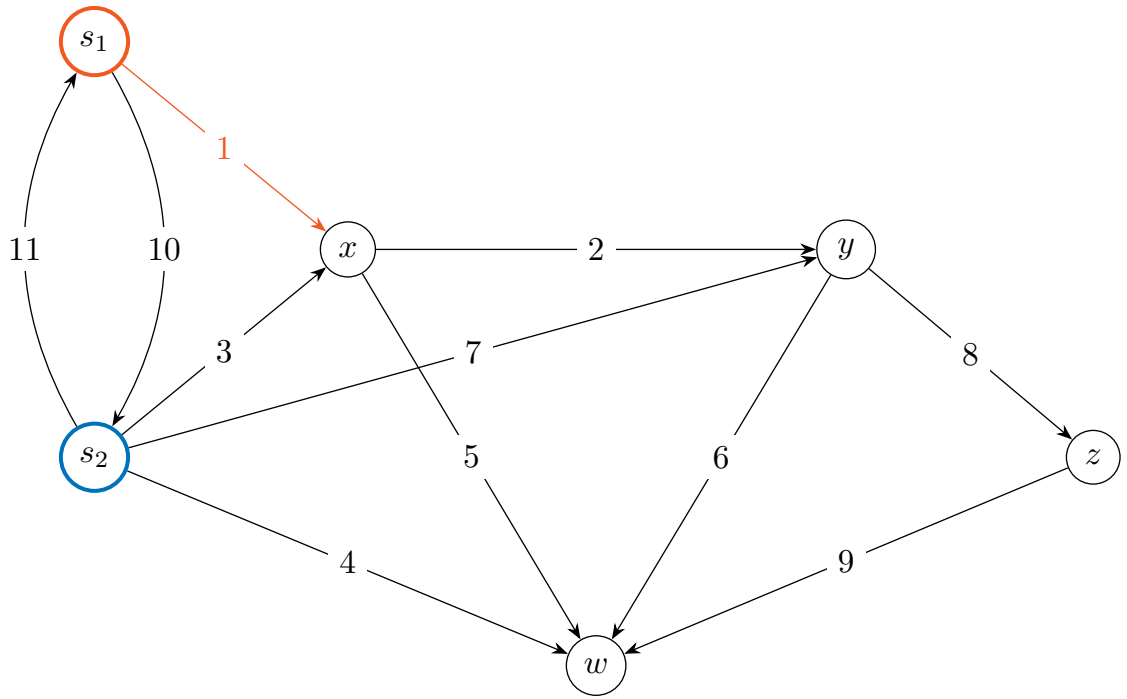
MANY2ALL - Polynomial Delay for $k = 2$ and $\ell = 1$

The parent-child relation induces a tree in the space of the spanners. Here we show how to find the root.



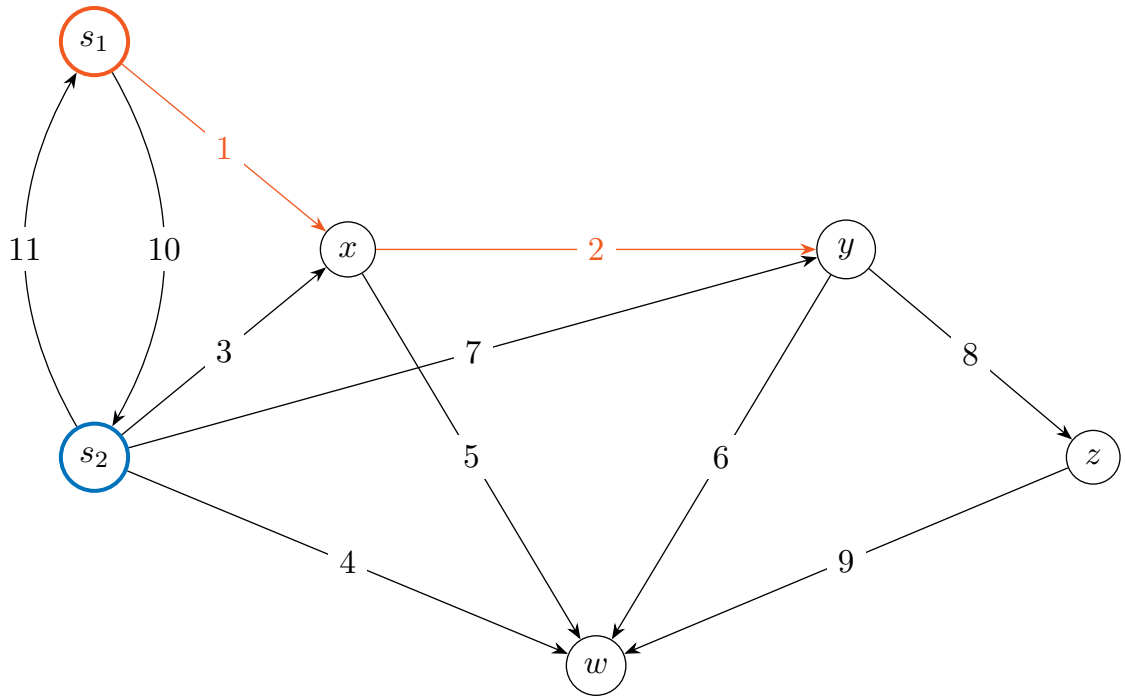
MANY2ALL - Polynomial Delay for $k = 2$ and $\ell = 1$

The parent-child relation induces a tree in the space of the spanners. Here we show how to find the root.



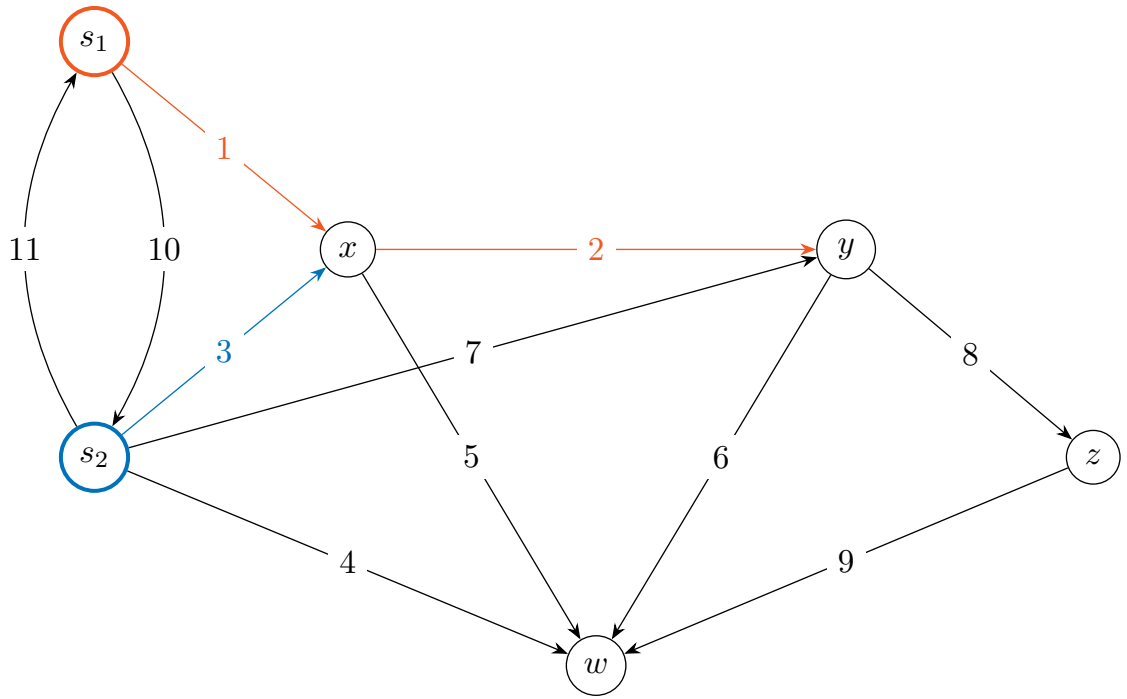
MANY2ALL - Polynomial Delay for $k = 2$ and $\ell = 1$

The parent-child relation induces a tree in the space of the spanners. Here we show how to find the root.



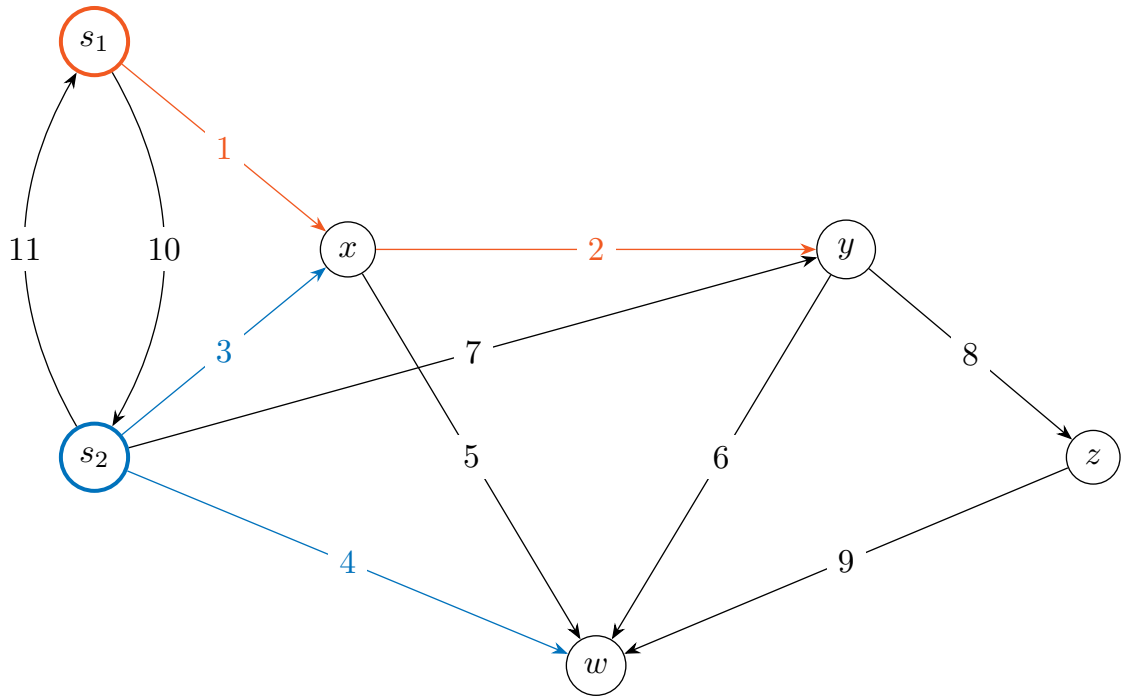
MANY2ALL - Polynomial Delay for $k = 2$ and $\ell = 1$

The parent-child relation induces a tree in the space of the spanners. Here we show how to find the root.



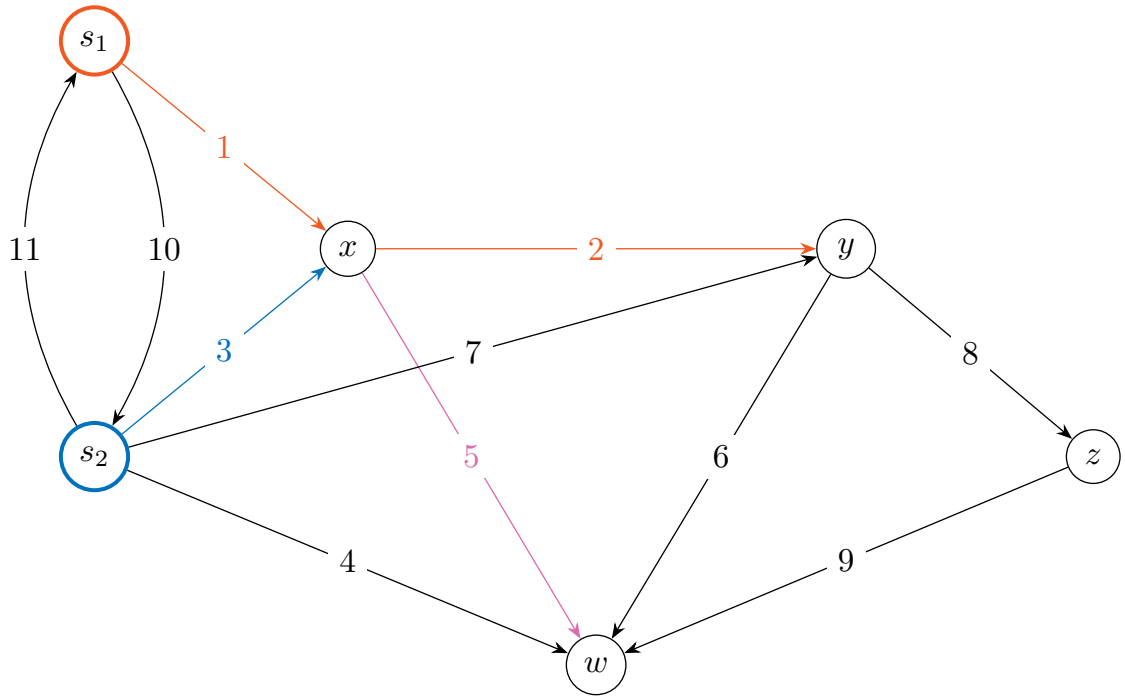
MANY2ALL - Polynomial Delay for $k = 2$ and $\ell = 1$

The parent-child relation induces a tree in the space of the spanners. Here we show how to find the root.



MANY2ALL - Polynomial Delay for $k = 2$ and $\ell = 1$

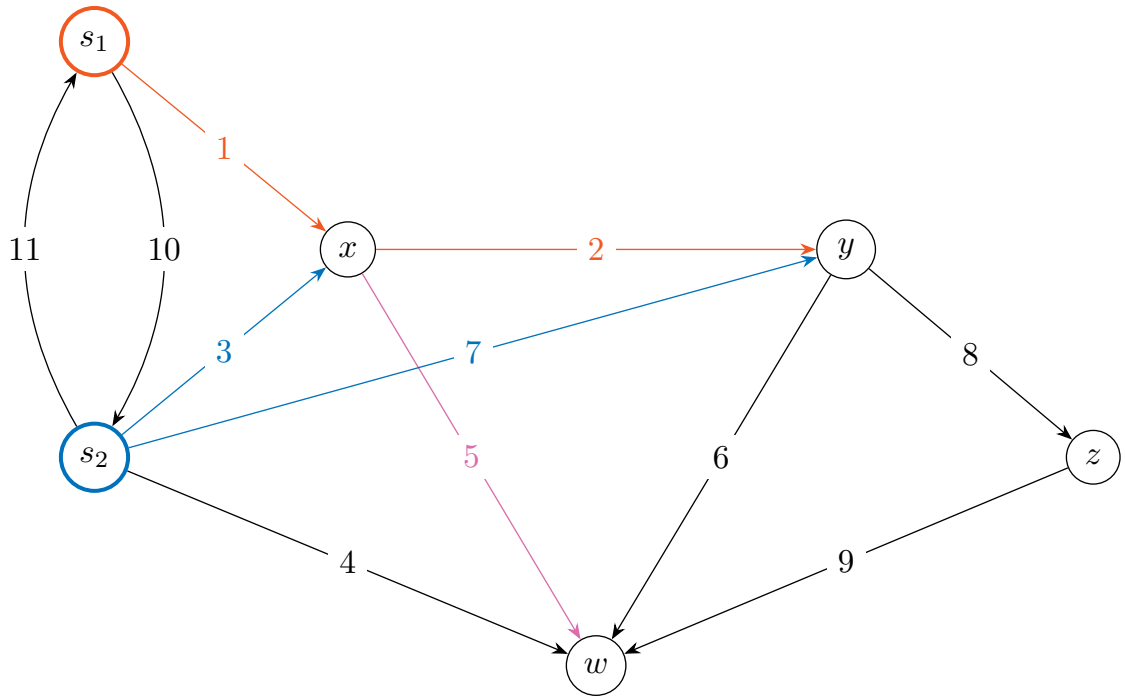
The parent-child relation induces a tree in the space of the spanners. Here we show how to find the root.



Note here we deleted the 4, because of cleaning.

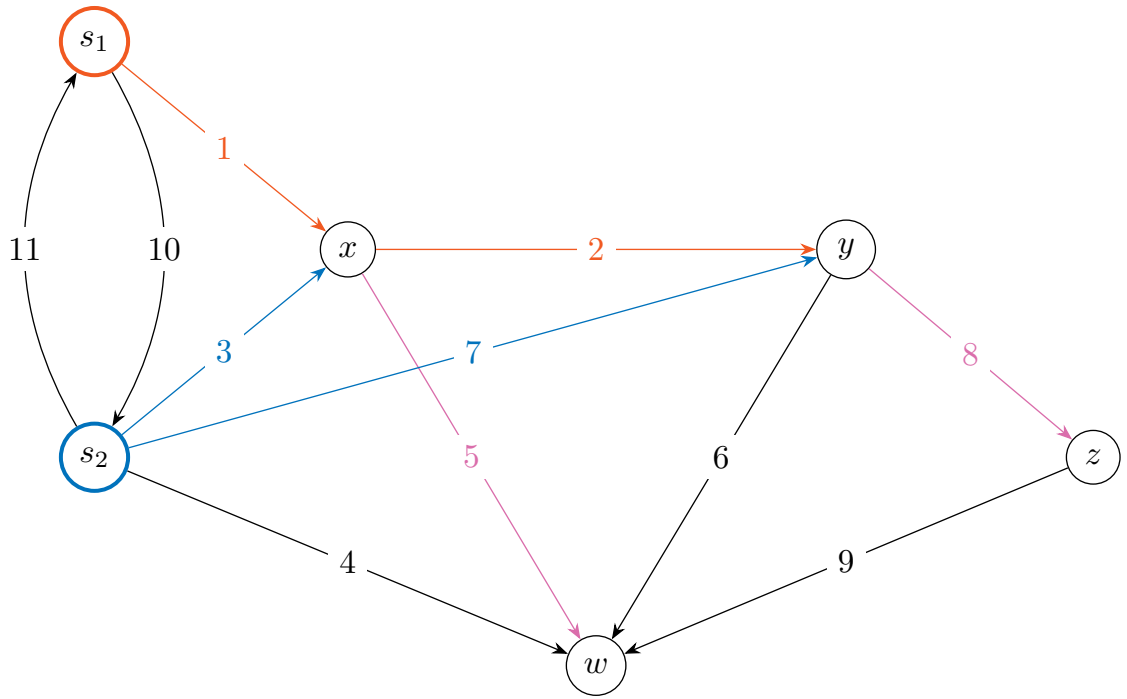
MANY2ALL - Polynomial Delay for $k = 2$ and $\ell = 1$

The parent-child relation induces a tree in the space of the spanners. Here we show how to find the root.



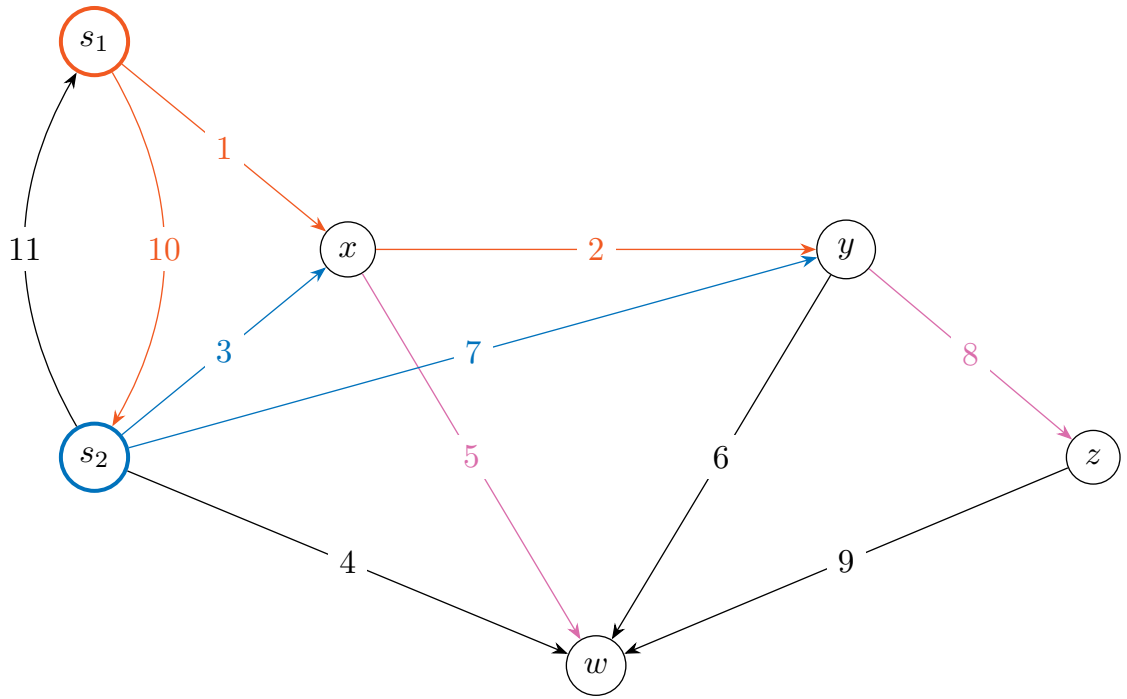
MANY2ALL - Polynomial Delay for $k = 2$ and $\ell = 1$

The parent-child relation induces a tree in the space of the spanners. Here we show how to find the root.



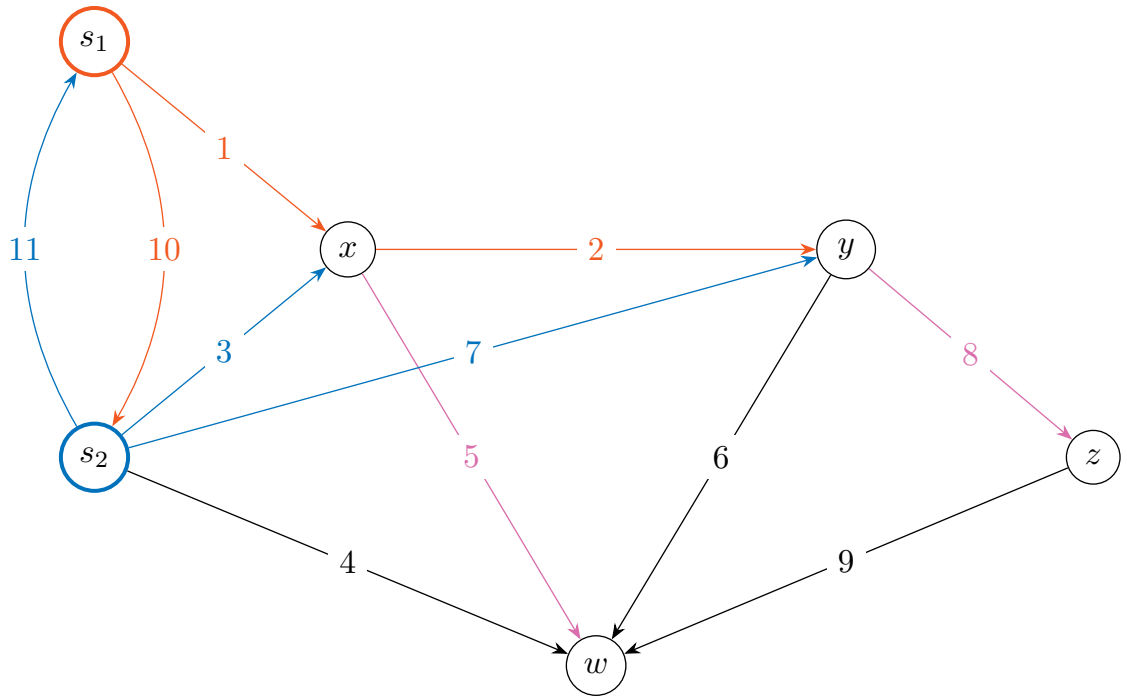
MANY2ALL - Polynomial Delay for $k = 2$ and $\ell = 1$

The parent-child relation induces a tree in the space of the spanners. Here we show how to find the root.



MANY2ALL - Polynomial Delay for $k = 2$ and $\ell = 1$

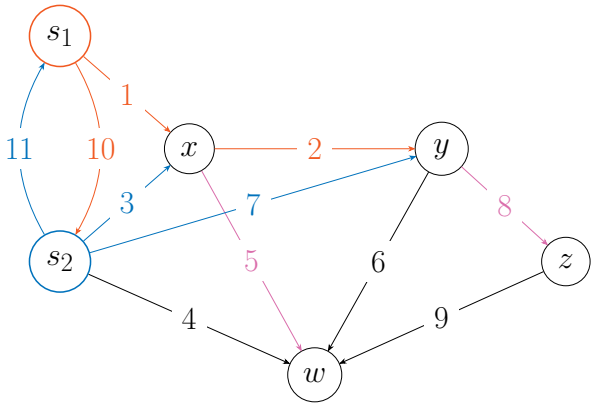
The parent-child relation induces a tree in the space of the spanners. Here we show how to find the root.



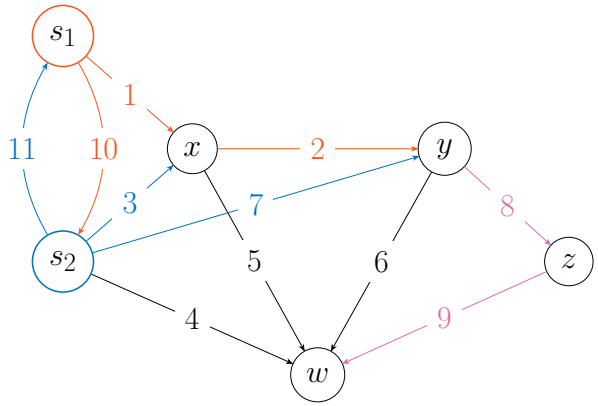
MANY2ALL - Polynomial Delay for $k = 2$ and $\ell = 1$

- Similar reasoning can be done starting from some given arcs.
- This intuitively corresponds to a **completion** of a partial solution, which in the case of the root is starting from an empty set of arcs.
- However, **this is not an extension** as it may delete some of the given arcs.
  - That's what happened to the edge labeled with 4.

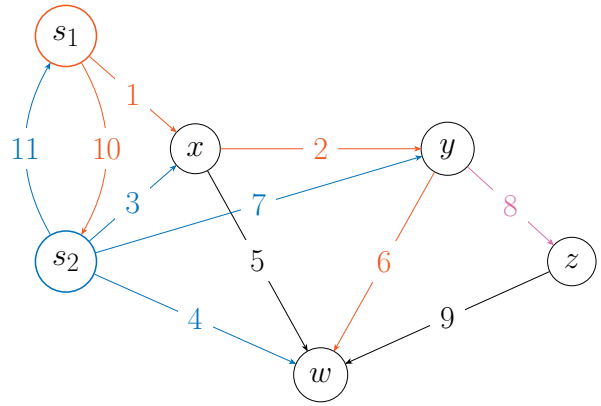
MANY2ALL - Polynomial Delay for $k = 2$ and $\ell = 1$



$P = \{1, 2, 3, 5, 7, 8, 10, 11\}$



$R = \{1, 2, 3, 7, 8, 9, 10, 11\}$.



$S = \{1, 2, 3, 4, 6, 8, 10, 11\}$.

- The parent of R and S is P because their maximum prefix such that completion is different resp. from R and S gives P.
- Computing the children of a parent is non-trivial. We need to define two kinds of children, called regular and special. In this example, R is a regular children, S is a special children.

Further work

We are working to adapt the algorithm for Enumeration many2all, 2 sources temporality 1 to:

- 2 sources, any temporality;
- any sources, temporality 1.

Thanks for your attention!