Selected results in temporal graph theory

Arnaud Casteigts

CS Dept.

University of Geneva

December 4, 2023

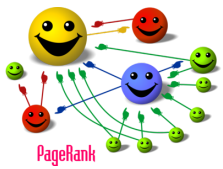
Séminaire “Groupes et Géométrie”,

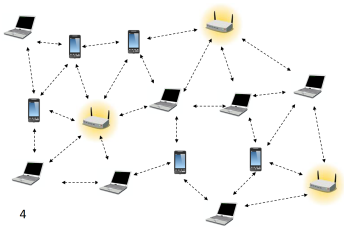
Section de mathématiques (UNIGE)

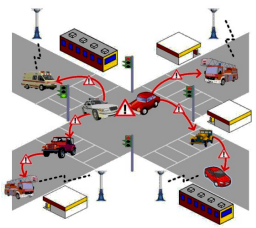
The world is dynamic...

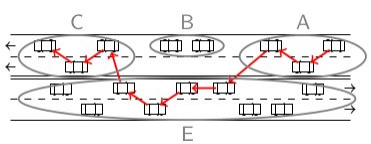
In technologies

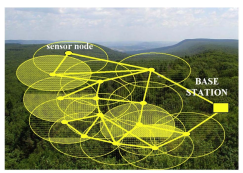


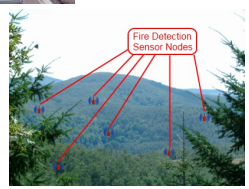


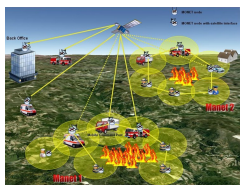


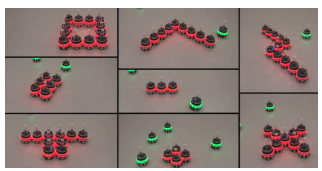








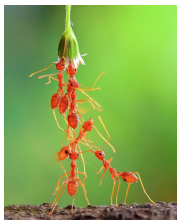


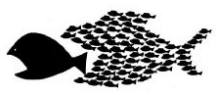


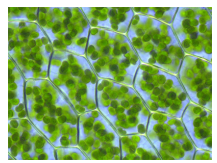
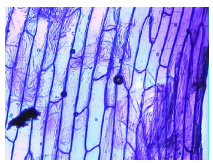

In nature





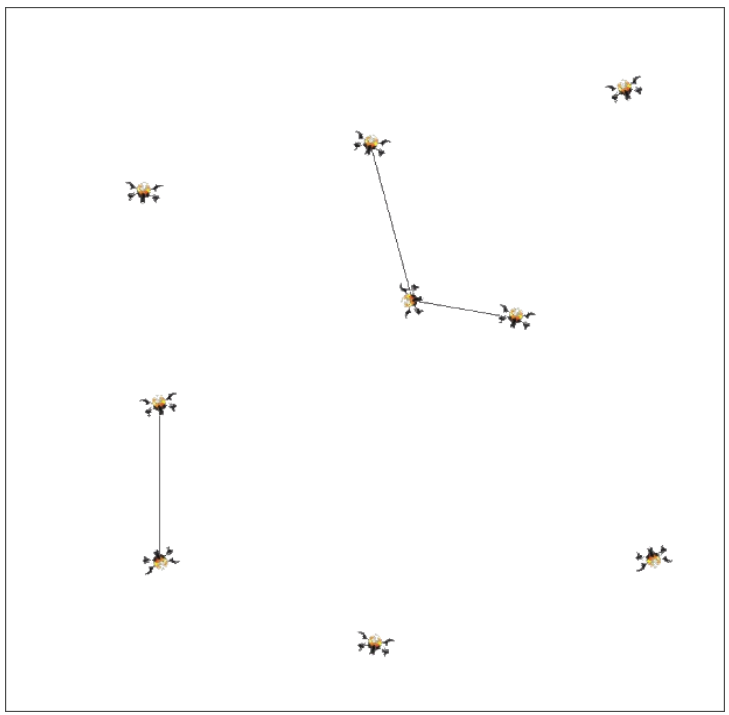



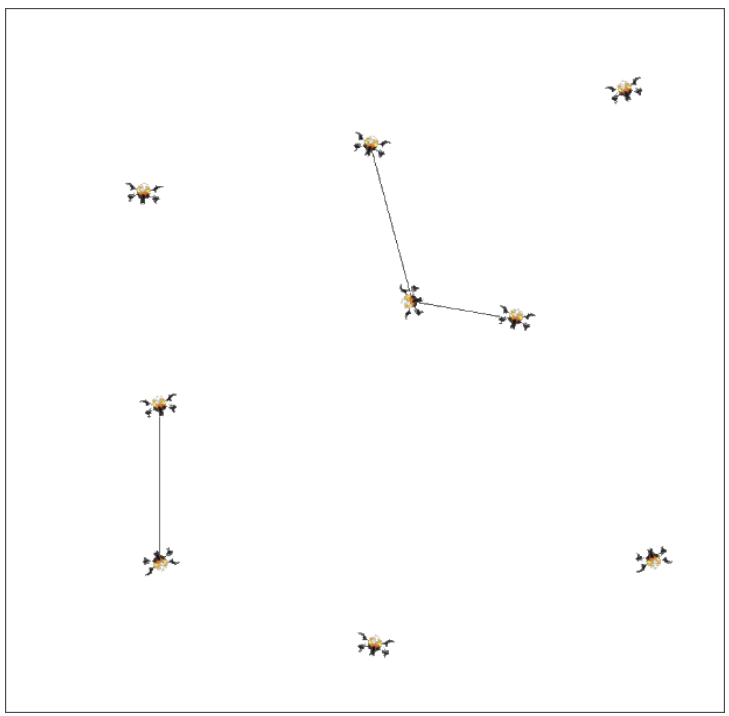

(Highly) dynamic networks?

Example of scenario

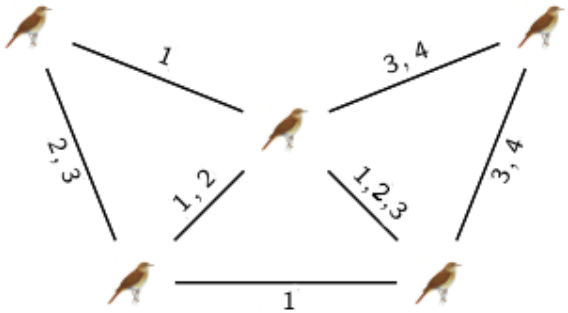


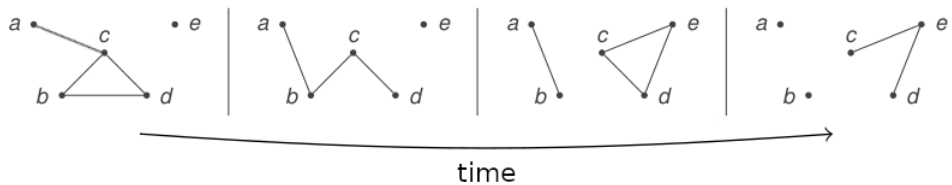
(Highly) dynamic networks?

Example of scenario



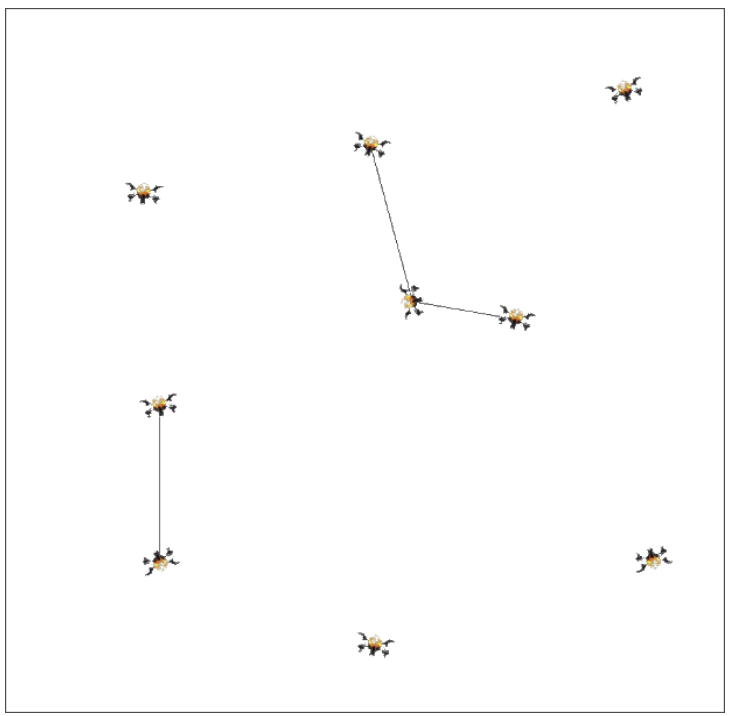
Modeling



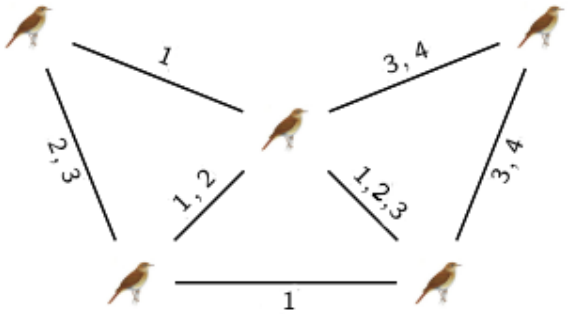


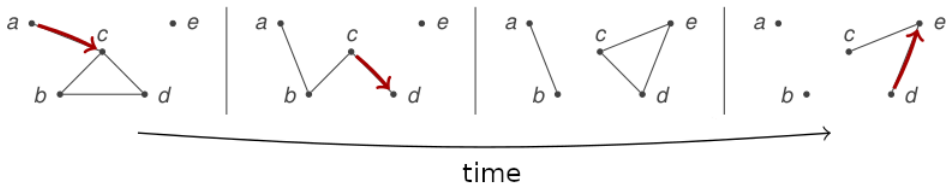
(Highly) dynamic networks?

Example of scenario



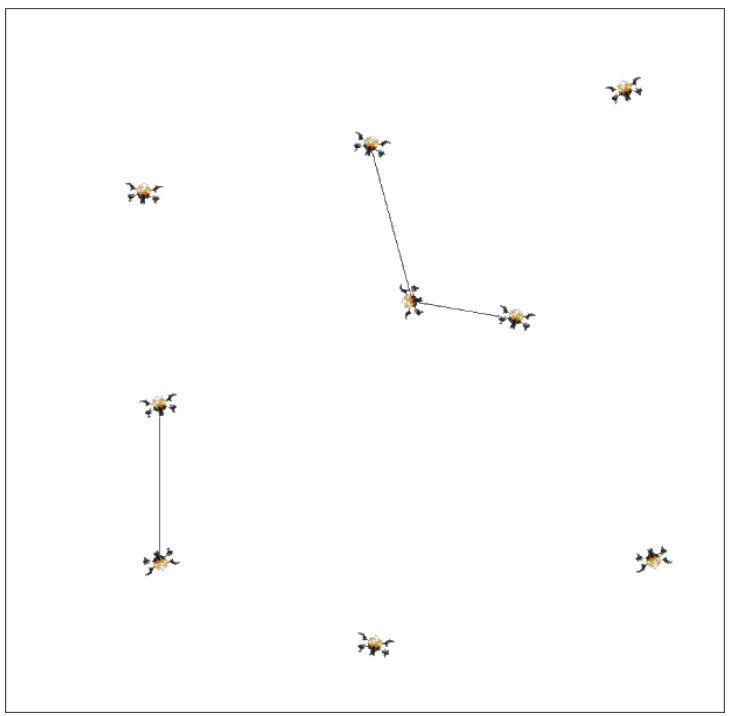
Modeling



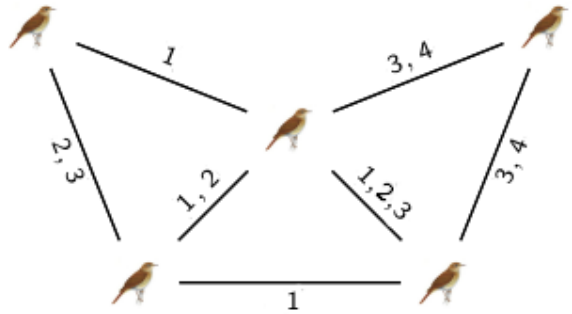


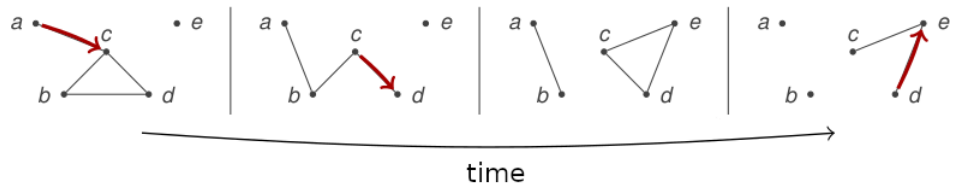
(Highly) dynamic networks?

Example of scenario



Modeling





Properties:

- Temporal connectivity? $\mathcal{TC}$
- Repeatedly? $\mathcal{TC}^{\mathcal{R}}$
- Recurrent links? $\mathcal{E}^{\mathcal{R}}$
- In bounded time? $\mathcal{E}^{\mathcal{B}}$
- ...

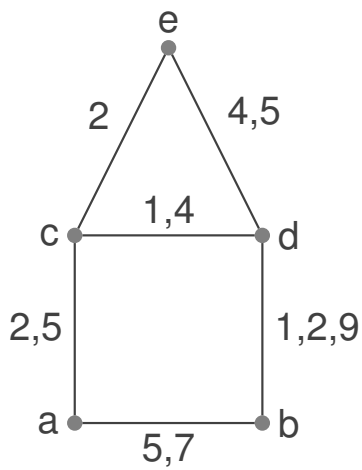
→ Classes of temporal graphs

Temporal graphs

(*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (V, E, \lambda)$, where $\lambda : E \to 2^{\mathbb{N}}$ assigns *time labels* to edges.

Example:

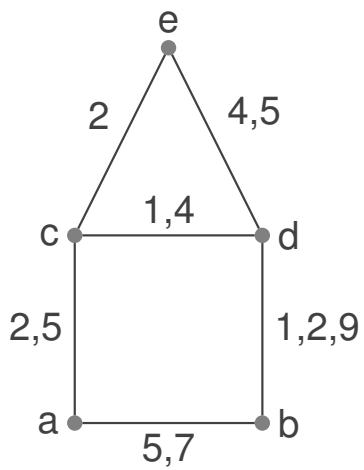


Temporal graphs

(*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (V, E, \lambda)$, where $\lambda : E \to 2^{\mathbb{N}}$ assigns *time labels* to edges.

Example:



Can also be viewed as a sequence of *snapshots* $\{G_i = \{e \in E : i \in \lambda(e)\}\}$

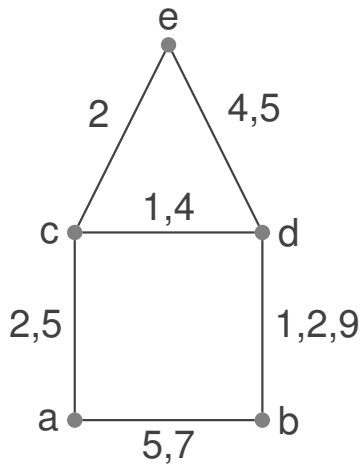
Temporal graphs

(*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (V, E, \lambda)$, where $\lambda : E \to 2^{\mathbb{N}}$ assigns *time labels* to edges.

(V, E): *footprint* of $\mathcal{G}$

Example:



Can also be viewed as a sequence of *snapshots* $\{G_i = \{e \in E : i \in \lambda(e)\}\}$

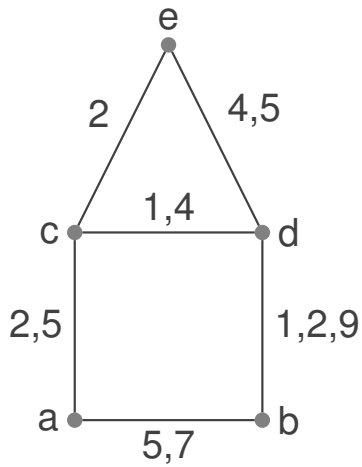
Temporal graphs

(*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (V, E, \lambda)$, where $\lambda : E \to 2^{\mathbb{N}}$ assigns *time labels* to edges.

(V, E): *footprint* of $\mathcal{G}$

Example:



Can also be viewed as a sequence of *snapshots* $\{G_i = \{e \in E : i \in \lambda(e)\}\}$

Restrictions on labeling: *simple* ($\lambda : E \to \mathbb{N}$); *proper* ($\lambda$ locally injective), *happy* (both).

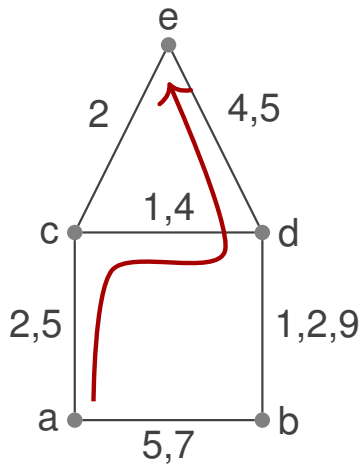
Temporal graphs

(*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (\underline{V, E}, \lambda)$, where $\lambda : E \to 2^{\mathbb{N}}$ assigns *time labels* to edges.

footprint of $\mathcal{G}$

Example:



Can also be viewed as a sequence of *snapshots* $\{G_i = \{e \in E : i \in \lambda(e)\}\}$

Restrictions on labeling: *simple* ($\lambda : E \to \mathbb{N}$); *proper* ($\lambda$ locally injective), *happy* (both).

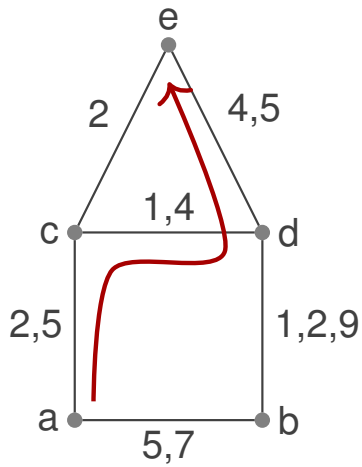
Temporal graphs

(*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (V, E, \lambda)$, where $\lambda : E \to 2^{\mathbb{N}}$ assigns *time labels* to edges.

(V, E): *footprint* of $\mathcal{G}$

Example:



Can also be viewed as a sequence of *snapshots* $\{G_i = \{e \in E : i \in \lambda(e)\}\}$

Restrictions on labeling: *simple* ($\lambda : E \to \mathbb{N}$); *proper* ($\lambda$ locally injective), *happy* (both).

Temporal paths

- e.g. $\langle (a, c, 2), (c, d, 4), (d, e, 5) \rangle$ (*strict*)
- e.g. $\langle (a, c, 2), (c, d, 4), (d, e, 4) \rangle$ (*non-strict*)

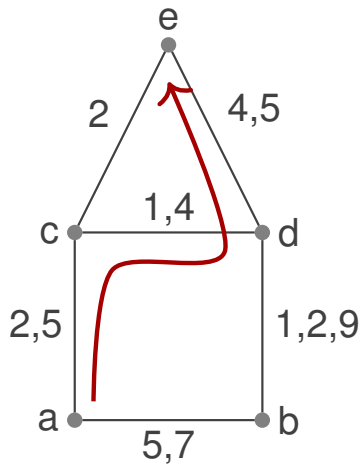
Temporal graphs

(*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (V, E, \lambda)$, where $\lambda : E \to 2^{\mathbb{N}}$ assigns *time labels* to edges.

(V, E): *footprint* of $\mathcal{G}$

Example:



Can also be viewed as a sequence of *snapshots* $\{G_i = \{e \in E : i \in \lambda(e)\}\}$

Restrictions on labeling: *simple* ($\lambda : E \to \mathbb{N}$); *proper* ($\lambda$ locally injective), *happy* (both).

Temporal paths

- e.g. $\langle (a, c, 2), (c, d, 4), (d, e, 5) \rangle$ (*strict*)
- e.g. $\langle (a, c, 2), (c, d, 4), (d, e, 4) \rangle$ (*non-strict*)

Temporal connectivity: All-pairs reachability (class TC).

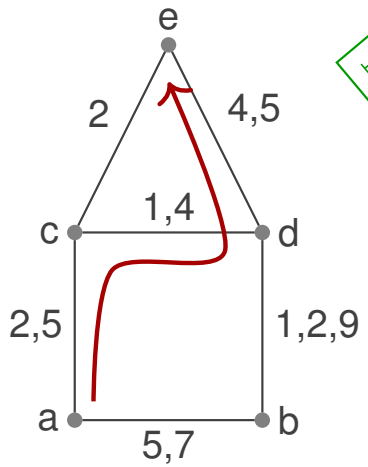
Temporal graphs

(*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (V, E, \lambda)$, where $\lambda : E \to 2^{\mathbb{N}}$ assigns *time labels* to edges.

(V, E): *footprint* of $\mathcal{G}$

Example:



Temporally connected

Can also be viewed as a sequence of *snapshots* $\{G_i = \{e \in E : i \in \lambda(e)\}\}$

Restrictions on labeling: *simple* ($\lambda : E \to \mathbb{N}$); *proper* ($\lambda$ locally injective), *happy* (both).

Temporal paths

- e.g. $\langle (a, c, 2), (c, d, 4), (d, e, 5) \rangle$ (*strict*)
- e.g. $\langle (a, c, 2), (c, d, 4), (d, e, 4) \rangle$ (*non-strict*)

Temporal connectivity: All-pairs reachability (class TC).

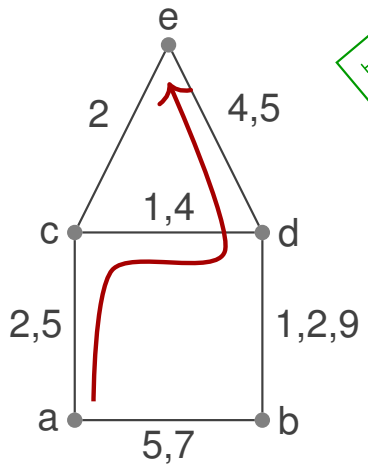
Temporal graphs

(*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (V, E, \lambda)$, where $\lambda : E \to 2^{\mathbb{N}}$ assigns *time labels* to edges.

(V, E): *footprint* of $\mathcal{G}$

Example:



Temporally connected

Can also be viewed as a sequence of *snapshots* $\{G_i = \{e \in E : i \in \lambda(e)\}\}$

Restrictions on labeling: *simple* ($\lambda : E \to \mathbb{N}$); *proper* ($\lambda$ locally injective), *happy* (both).

Temporal paths

- e.g. $\langle (a, c, 2), (c, d, 4), (d, e, 5) \rangle$ (*strict*)
- e.g. $\langle (a, c, 2), (c, d, 4), (d, e, 4) \rangle$ (*non-strict*)

Temporal connectivity: All-pairs reachability (class TC).

$\rightarrow$ Warning: In general, reachability is non-symmetrical...

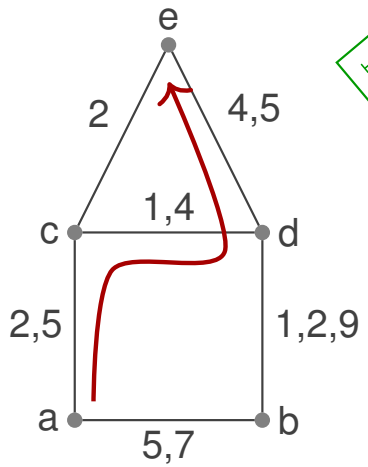
Temporal graphs

(*a.k.a.* time-varying, time-dependent, evolving, dynamic,...)

$\mathcal{G} = (V, E, \lambda)$, where $\lambda : E \to 2^{\mathbb{N}}$ assigns *time labels* to edges.

(V, E): *footprint* of $\mathcal{G}$

Example:



Temporally connected

Can also be viewed as a sequence of *snapshots* $\{G_i = \{e \in E : i \in \lambda(e)\}\}$

Restrictions on labeling: *simple* ($\lambda : E \to \mathbb{N}$); *proper* ($\lambda$ locally injective), *happy* (both).

Temporal paths

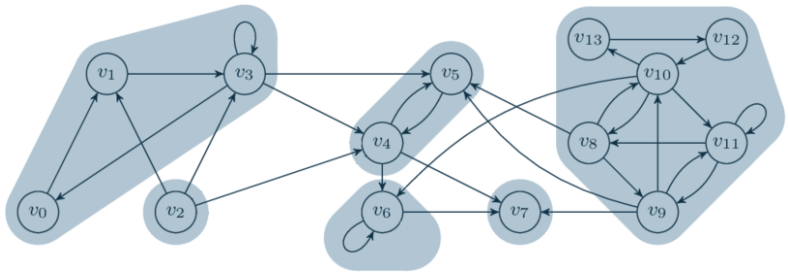
- e.g. $\langle (a, c, 2), (c, d, 4), (d, e, 5) \rangle$ (*strict*)
- e.g. $\langle (a, c, 2), (c, d, 4), (d, e, 4) \rangle$ (*non-strict*)

Temporal connectivity: All-pairs reachability (class TC).

$\rightarrow$ Warning: In general, reachability is non-symmetrical... and non-transitive!

Impact of non-transitivity (Example: CONNECTED COMPONENTS)

In static graphs

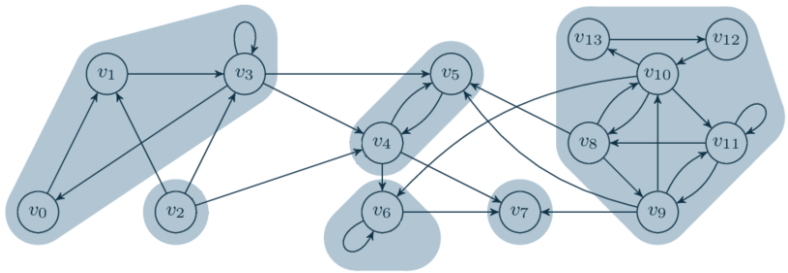


- Components define a partition
- Easy to compute

Impact of non-transitivity

(Example: CONNECTED COMPONENTS)

In static graphs



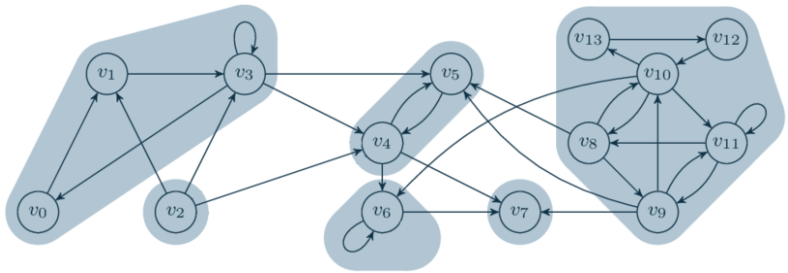
- Components define a partition
- Easy to compute

In temporal graphs

- Maximal components may overlap
- Can be exponentially many

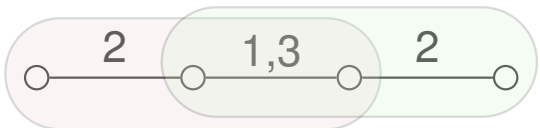
Impact of non-transitivity (Example: CONNECTED COMPONENTS)

In static graphs



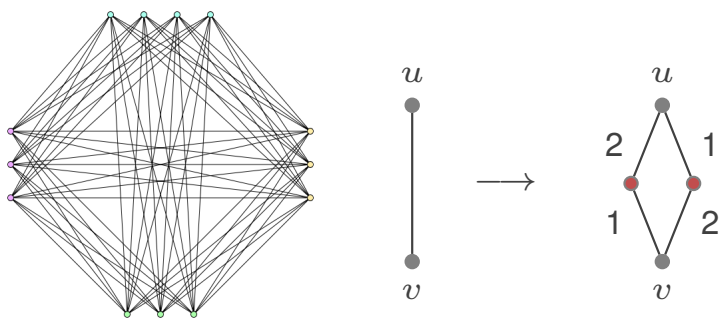
- Components define a partition
- Easy to compute

In temporal graphs



- Maximal components may overlap
- Can be exponentially many

COMPONENT is NP-hard! (from CLIQUE) [Bhadra, Ferreira, 2003]



- Replace edges with semaphore gadgets
- Cliques $\Longleftrightarrow$ temporal components

Spanning trees

In static graphs

- Existence is guaranteed
- Size is always $n - 1$

Spanning trees

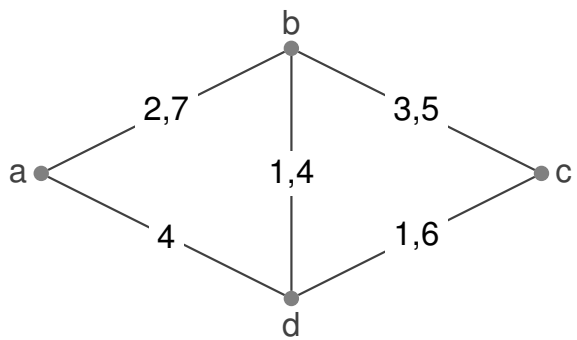
In static graphs

- Existence is guaranteed
- Size is always $n - 1$

Temporal spanning tree ?

Input: A temporal graph $\mathcal{G} \in$ TC.

Goal: Find a spanning tree S of the *footprint*, so that $\mathcal{G}[S] \in$ TC.



Spanning trees

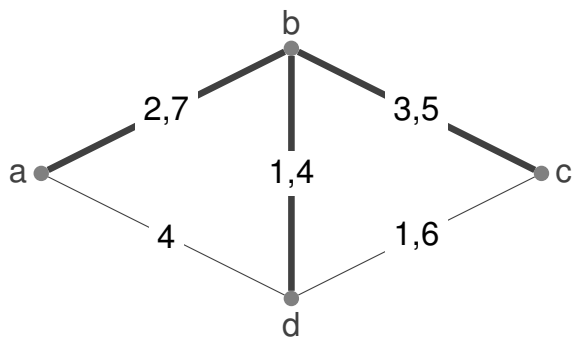
In static graphs

- Existence is guaranteed
- Size is always $n - 1$

Temporal spanning tree ?

Input: A temporal graph $\mathcal{G} \in$ TC.

Goal: Find a spanning tree S of the *footprint*, so that $\mathcal{G}[S] \in$ TC.



Spanning trees

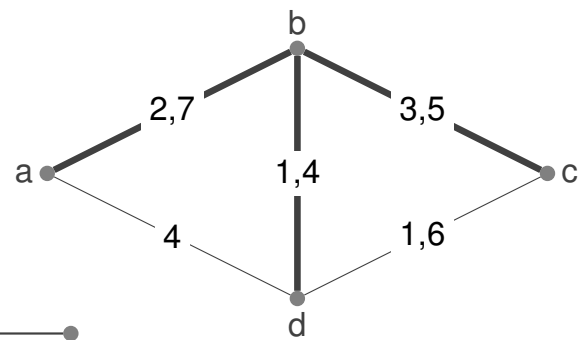
In static graphs

- Existence is guaranteed
- Size is always $n - 1$

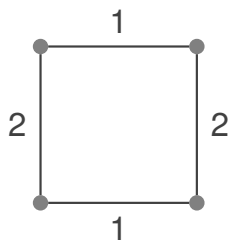
Temporal spanning tree ?

Input: A temporal graph $\mathcal{G} \in$ TC.

Goal: Find a spanning tree S of the *footprint*, so that $\mathcal{G}[S] \in$ TC.



Does not always exist:



Spanning trees

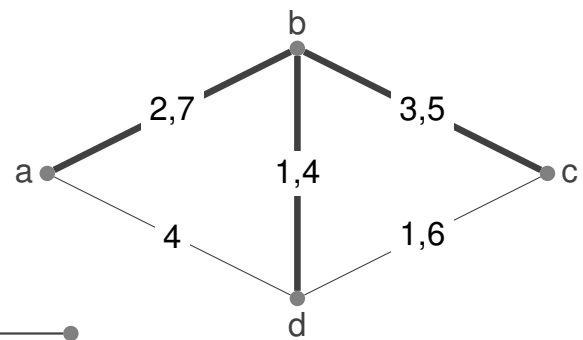
In static graphs

- Existence is guaranteed
- Size is always $n - 1$

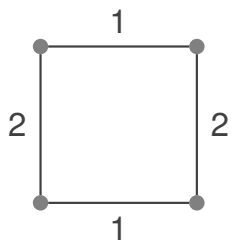
Temporal spanning tree ?

Input: A temporal graph $\mathcal{G} \in$ TC.

Goal: Find a spanning tree S of the *footprint*, so that $\mathcal{G}[S] \in$ TC.



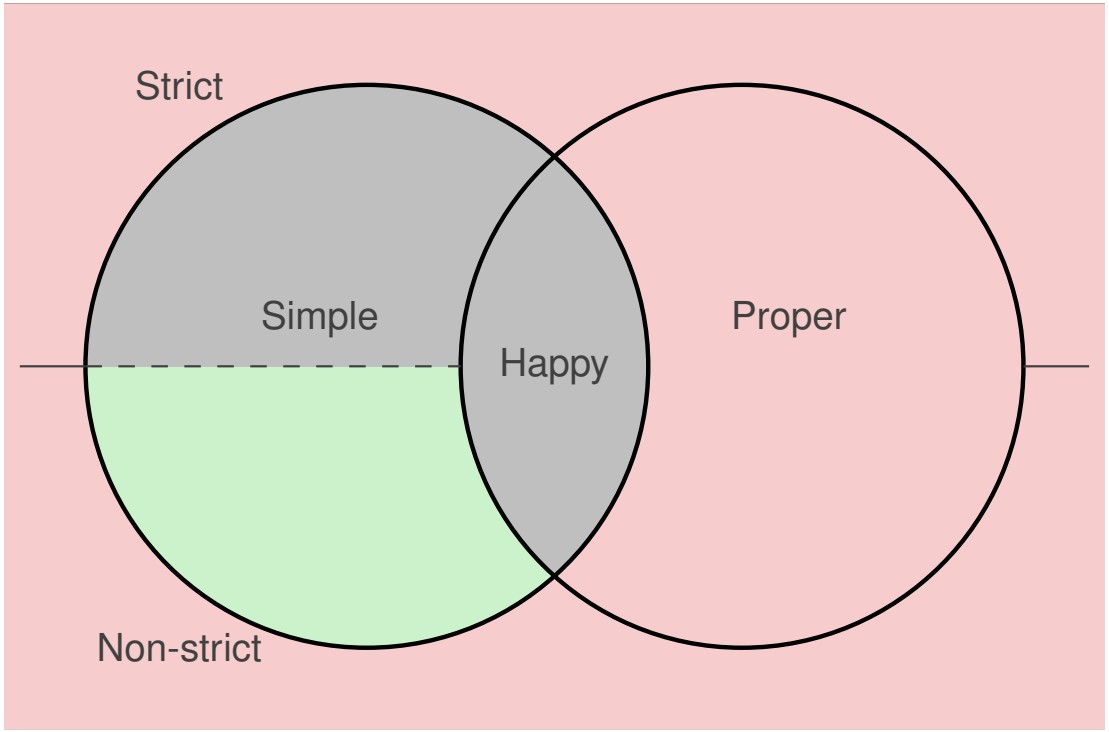
Does not always exist:



In fact, NP-hard to decide!

[C., Corsini, 2024]

Lanscape of hardness for SPANNING TREE



NP-Hard / Polynomial / Always no

Searching for the lost tree

What to replace trees?

Searching for the lost tree

What to replace trees?

$\rightarrow$ Small reachability substructures (*temporal spanners*).

Searching for the lost tree

What to replace trees?

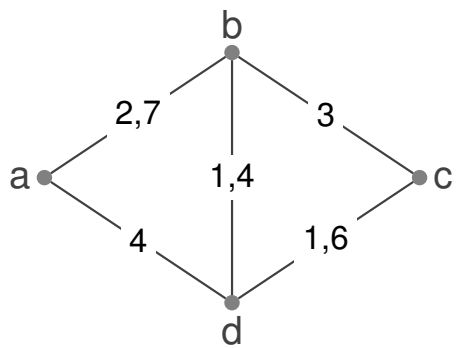
$\rightarrow$ Small reachability substructures (*temporal spanners*).

Temporal spanners

Input: a temporal graph $\mathcal{G} \in \mathrm{TC}$

Output: a temporal subgraph $\mathcal{G}' \subseteq \mathcal{G}$ such that $\mathcal{G}' \in \mathrm{TC}$

Cost measure: # edges or # labels



Searching for the lost tree

What to replace trees?

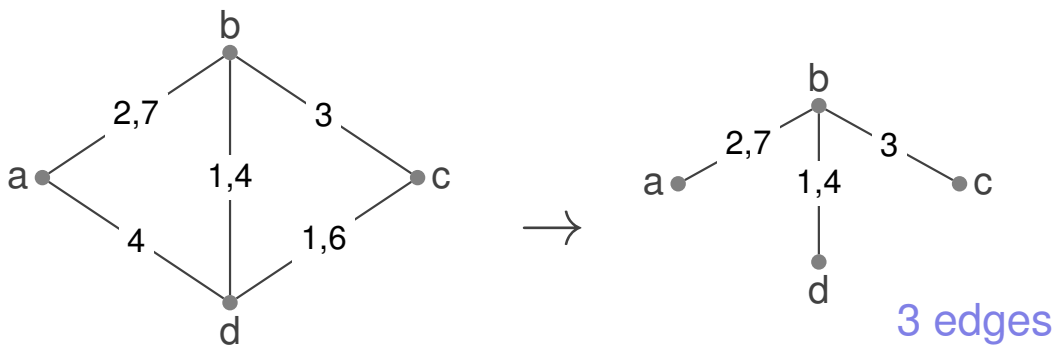
$\rightarrow$ Small reachability substructures (*temporal spanners*).

Temporal spanners

Input: a temporal graph $\mathcal{G} \in$ TC

Output: a temporal subgraph $\mathcal{G}' \subseteq \mathcal{G}$ such that $\mathcal{G}' \in$ TC

Cost measure: # edges or # labels



Searching for the lost tree

What to replace trees?

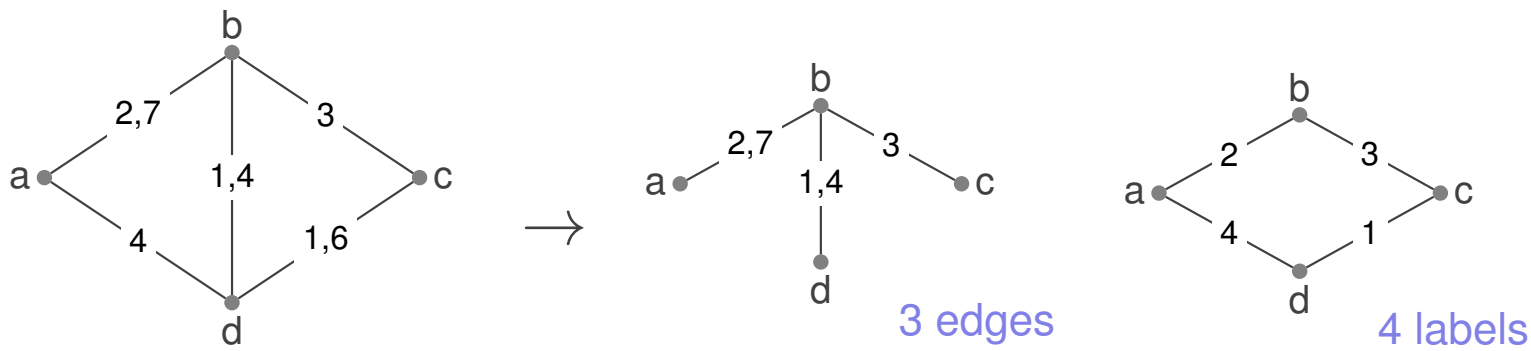
$\rightarrow$ Small reachability substructures (*temporal spanners*).

Temporal spanners

Input: a temporal graph $\mathcal{G} \in$ TC

Output: a temporal subgraph $\mathcal{G}' \subseteq \mathcal{G}$ such that $\mathcal{G}' \in$ TC

Cost measure: # edges or # labels



Searching for the lost tree

What to replace trees?

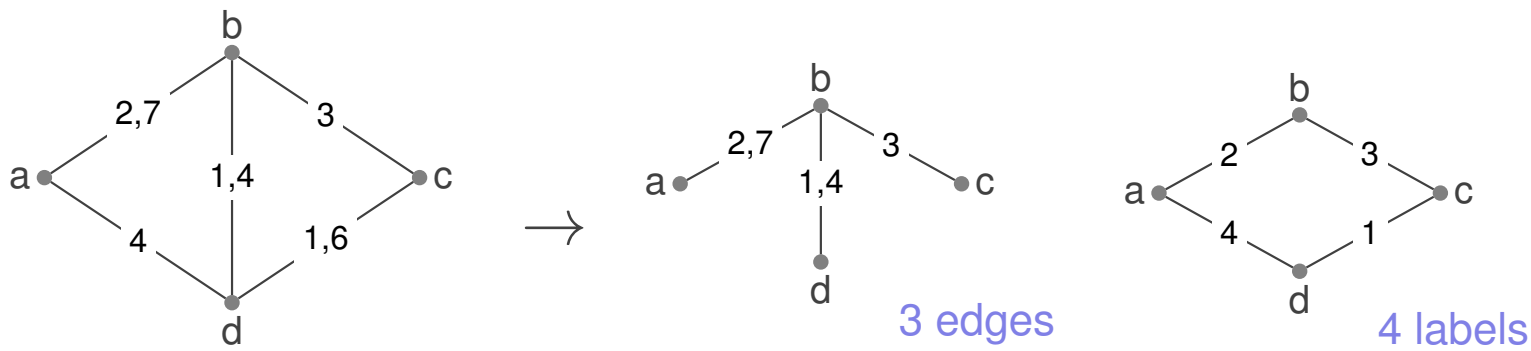
$\rightarrow$ Small reachability substructures (*temporal spanners*).

Temporal spanners

Input: a temporal graph $\mathcal{G} \in$ TC

Output: a temporal subgraph $\mathcal{G}' \subseteq \mathcal{G}$ such that $\mathcal{G}' \in$ TC

Cost measure: # edges or # labels



Complexity:

- MIN-LABEL: APX-hard for non-simple, non-proper, strict [Akrida, Gasieniec, Mertzios, Spirakis, 2017]
- MIN-EDGE (and MIN-LABEL): APX-hard for simple, non-proper, non-strict [Axiotis, Fotakis, 2016]

Searching for the lost tree

What to replace trees?

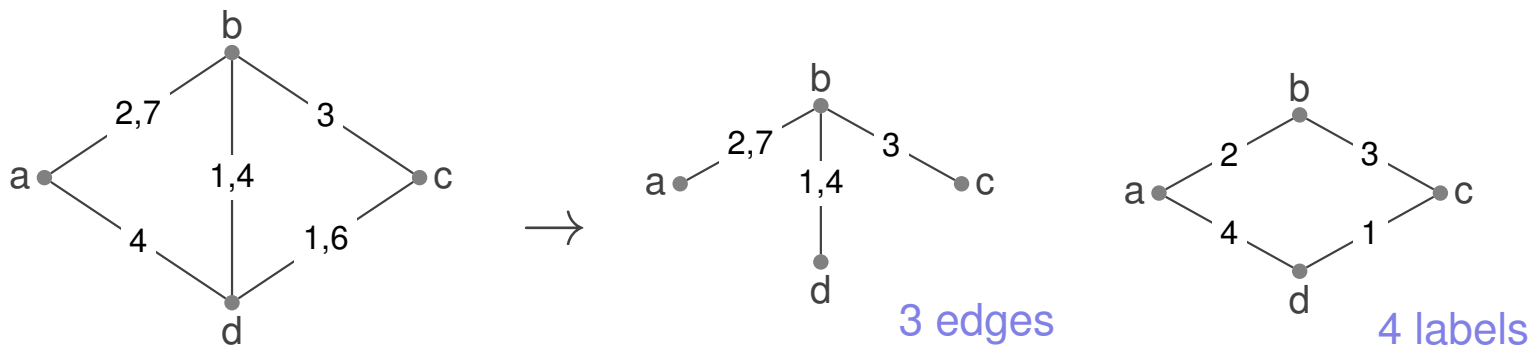
$\rightarrow$ Small reachability substructures (*temporal spanners*).

Temporal spanners

Input: a temporal graph $\mathcal{G} \in$ TC

Output: a temporal subgraph $\mathcal{G}' \subseteq \mathcal{G}$ such that $\mathcal{G}' \in$ TC

Cost measure: # edges or # labels



Complexity:

- MIN-LABEL: APX-hard for non-simple, non-proper, strict [Akrida, Gasieniec, Mertzios, Spirakis, 2017]
- MIN-EDGE (and MIN-LABEL): APX-hard for simple, non-proper, non-strict [Axiotis, Fotakis, 2016]

Open for happy graphs (i.e. both simple and proper), could be polynomial.

Structural results

Given a temporal graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in \mathrm{TC}$),
is there any guarantee on the size of a minimum spanner $\mathcal{G}' \subseteq \mathcal{G}$?

Structural results

Given a temporal graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in$ TC),
is there any guarantee on the size of a minimum spanner $\mathcal{G}' \subseteq \mathcal{G}$?

Note: The absolute minimum is $2n - 4$ [Bumby, 1979 (gossip theory)]

Structural results

Given a temporal graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in$ TC),
is there any guarantee on the size of a minimum spanner $\mathcal{G}' \subseteq \mathcal{G}$?

Note: The absolute minimum is $2n - 4$ [Bumby, 1979 (gossip theory)]

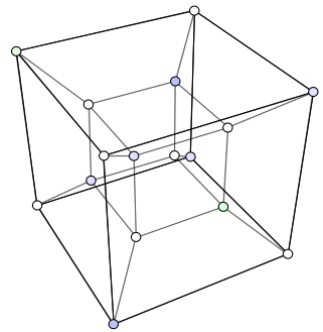
- Are spanners of size $O(n)$ always guaranteed?

Structural results

Given a temporal graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in$ TC), is there any guarantee on the size of a minimum spanner $\mathcal{G}' \subseteq \mathcal{G}$?

Note: The absolute minimum is $2n - 4$ [Bumby, 1979 (gossip theory)]

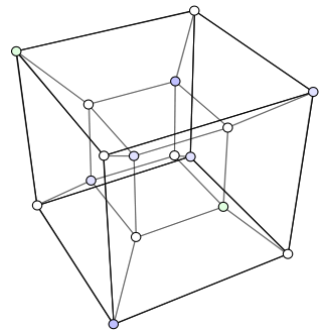
- Are spanners of size $O(n)$ always guaranteed?
  $\rightarrow$ Nope, hypercubes may fail [Kleinberg, Kempe, Kumar, 2000]



Structural results

Given a temporal graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in$ TC), is there any guarantee on the size of a minimum spanner $\mathcal{G}' \subseteq \mathcal{G}$?

Note: The absolute minimum is $2n - 4$ [Bumby, 1979 (gossip theory)]



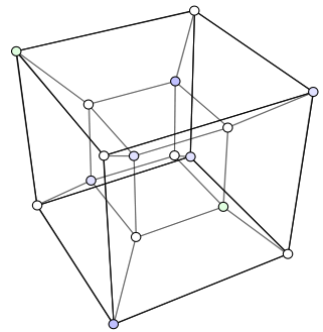
- Are spanners of size $O(n)$ always guaranteed?
  $\rightarrow$ Nope, hypercubes may fail [Kleinberg, Kempe, Kumar, 2000]
- Are spanners of size $o(n^2)$ always guaranteed?

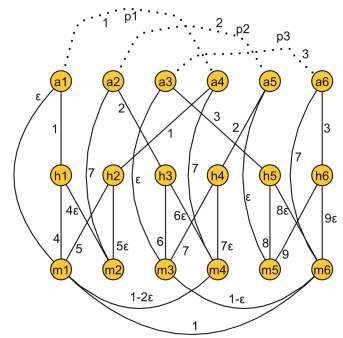
Structural results

Given a temporal graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in$ TC), is there any guarantee on the size of a minimum spanner $\mathcal{G}' \subseteq \mathcal{G}$?

Note: The absolute minimum is $2n - 4$ [Bumby, 1979 (gossip theory)]

- Are spanners of size $O(n)$ always guaranteed?
  $\rightarrow$ Nope, hypercubes may fail [Kleinberg, Kempe, Kumar, 2000]
- Are spanners of size $o(n^2)$ always guaranteed?
  $\rightarrow$ Not even! [Axiotis, Fotakis, 2016]

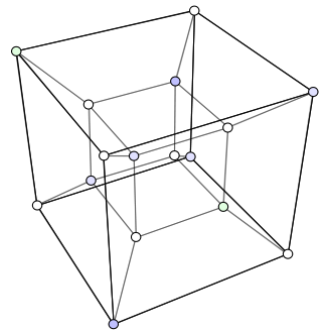




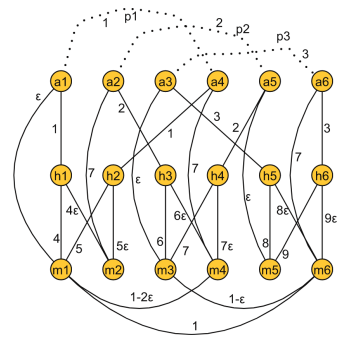
Structural results

Given a temporal graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in$ TC), is there any guarantee on the size of a minimum spanner $\mathcal{G}' \subseteq \mathcal{G}$?

Note: The absolute minimum is $2n - 4$ [Bumby, 1979 (gossip theory)]



- Are spanners of size $O(n)$ always guaranteed?
  $\rightarrow$ Nope, hypercubes may fail [Kleinberg, Kempe, Kumar, 2000]
- Are spanners of size $o(n^2)$ always guaranteed?
  $\rightarrow$ Not even! [Axiotis, Fotakis, 2016]



Any positive results?

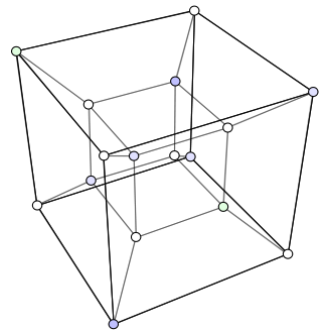
Good news 1 (probabilistic): [C., Raskin, Renken, Zamaraev, FOCS 2021]:

- Nearly optimal spanners (of size $2n + o(n)$) almost surely exist in **random** temporal graphs, and so, **as soon as** the graph becomes TC!

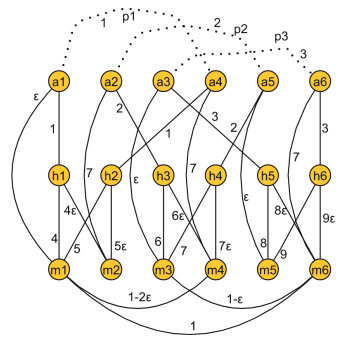
Structural results

Given a temporal graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in$ TC), is there any guarantee on the size of a minimum spanner $\mathcal{G}' \subseteq \mathcal{G}$?

Note: The absolute minimum is $2n - 4$ [Bumby, 1979 (gossip theory)]



- Are spanners of size $O(n)$ always guaranteed?
  $\rightarrow$ Nope, hypercubes may fail [Kleinberg, Kempe, Kumar, 2000]
- Are spanners of size $o(n^2)$ always guaranteed?
  $\rightarrow$ Not even! [Axiotis, Fotakis, 2016]



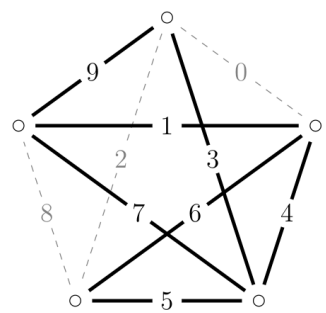
Any positive results?

Good news 1 (probabilistic): [C., Raskin, Renken, Zamaraev, FOCS 2021]:

- Nearly optimal spanners (of size $2n + o(n)$) almost surely exist in **random** temporal graphs, and so, **as soon as** the graph becomes TC!

Good news 2 (deterministic): [C., Peters, Schoeters, ICALP 2019]:

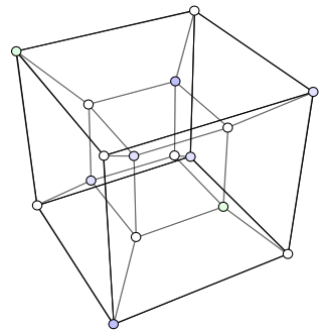
- Spanners of size $O(n \log n)$ always exist in temporal **cliques**



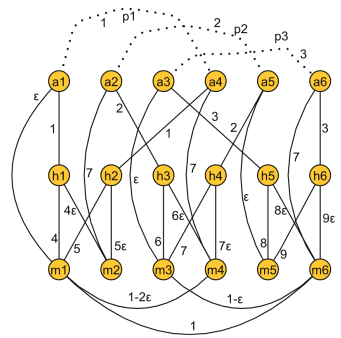
Structural results

Given a temporal graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in$ TC), is there any guarantee on the size of a minimum spanner $\mathcal{G}' \subseteq \mathcal{G}$?

Note: The absolute minimum is $2n - 4$ [Bumby, 1979 (gossip theory)]



- Are spanners of size $O(n)$ always guaranteed?
  $\rightarrow$ Nope, hypercubes may fail [Kleinberg, Kempe, Kumar, 2000]
- Are spanners of size $o(n^2)$ always guaranteed?
  $\rightarrow$ Not even! [Axiotis, Fotakis, 2016]



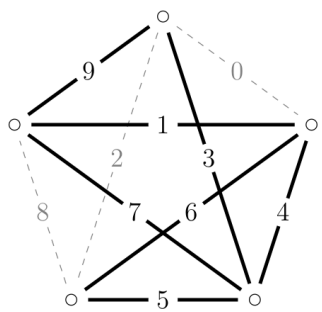
Any positive results?

Good news 1 (probabilistic): [C., Raskin, Renken, Zamaraev, FOCS 2021]:

- Nearly optimal spanners (of size $2n + o(n)$) almost surely exist in **random** temporal graphs, and so, **as soon as** the graph becomes TC!

Good news 2 (deterministic): [C., Peters, Schoeters, ICALP 2019]:

- Spanners of size $O(n \log n)$ always exist in temporal **cliques**



Open: $O(n)$ in temporal cliques?

Good news

Good news

Good news 1: (C., Raskin, Renken, Zamaraev, FOCS 2021):

- Nearly optimal spanners (of size $2n + o(n)$) almost surely exist in **random** temporal graphs, as soon as the graph becomes temporally connected

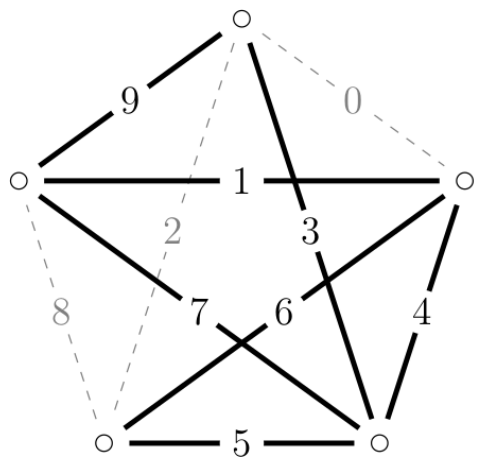
Good news

Good news 1: (C., Raskin, Renken, Zamaraev, FOCS 2021):

- Nearly optimal spanners (of size $2n + o(n)$) almost surely exist in **random** temporal graphs, as soon as the graph becomes temporally connected

Good news 2: (C., Peters, Schoeters, ICALP 2019):

- Spanners of size $O(n \log n)$ always exist in temporal **cliques**



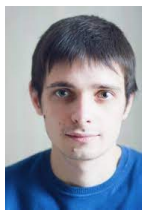
Good news 1:

Spanners of size $2n + o(n)$ almost surely exist

in random temporal graphs

(with)

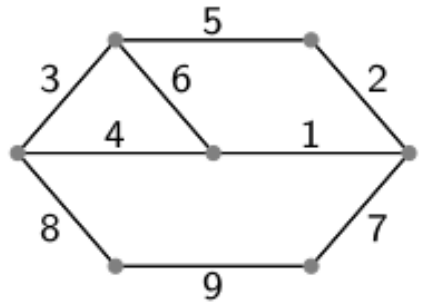




Connectivity in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random happy temporal graphs:

1. Pick an Erdös-Rényi $G \sim G_{n,p}$
2. Permute the edges randomly, interpret as (unique) presence time

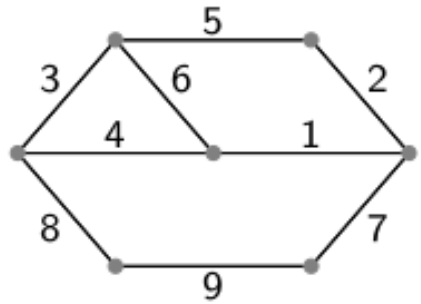


Timeline for p (as $n \to \infty$):

Connectivity in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random happy temporal graphs:

1. Pick an Erdös-Rényi $G \sim G_{n,p}$
2. Permute the edges randomly, interpret as (unique) presence time

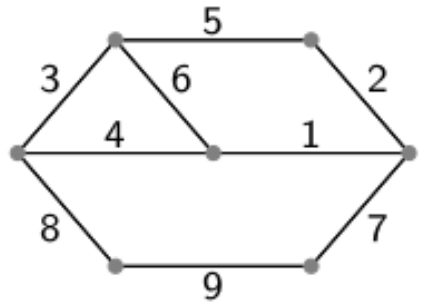


Timeline for p (as $n \to \infty$):

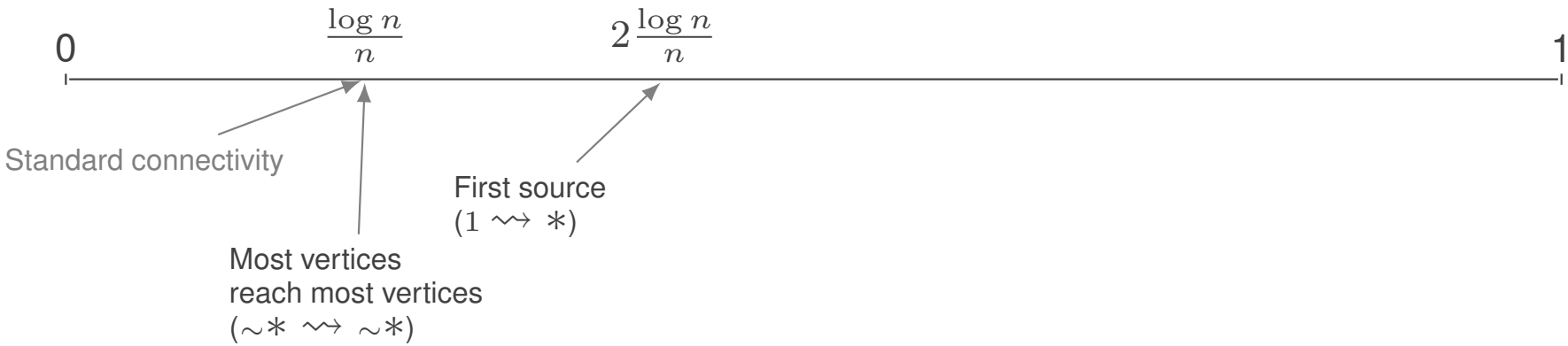
Connectivity in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random happy temporal graphs:

1. Pick an Erdös-Rényi $G \sim G_{n,p}$
2. Permute the edges randomly, interpret as (unique) presence time



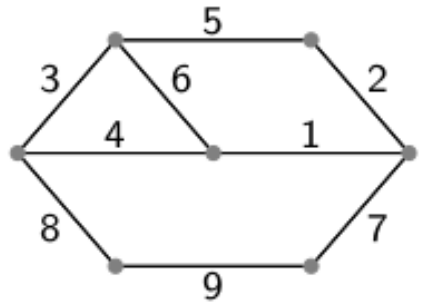
Timeline for p (as $n \to \infty$):



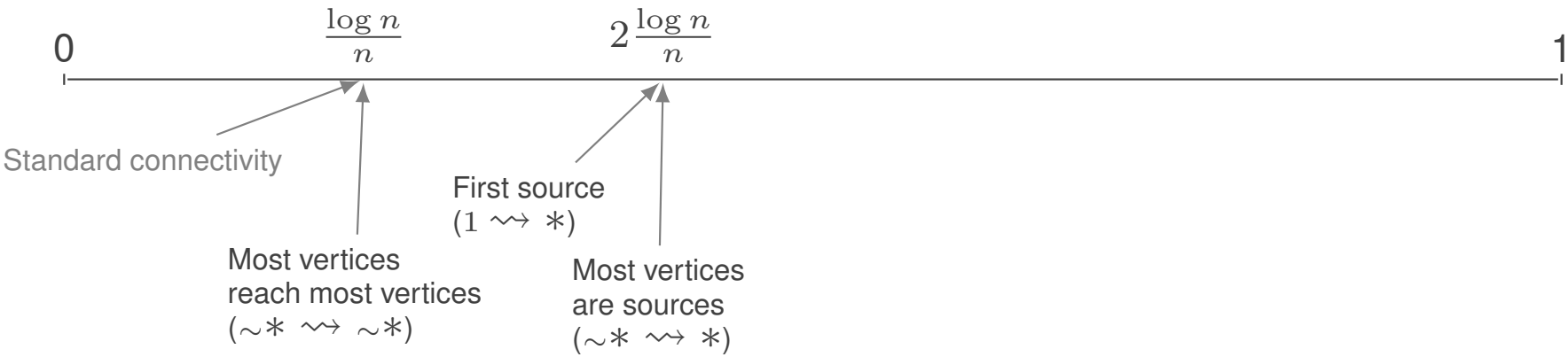
Connectivity in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random happy temporal graphs:

1. Pick an Erdös-Rényi $G \sim G_{n,p}$
2. Permute the edges randomly, interpret as (unique) presence time



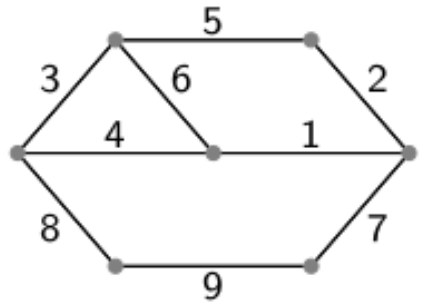
Timeline for p (as $n \to \infty$):



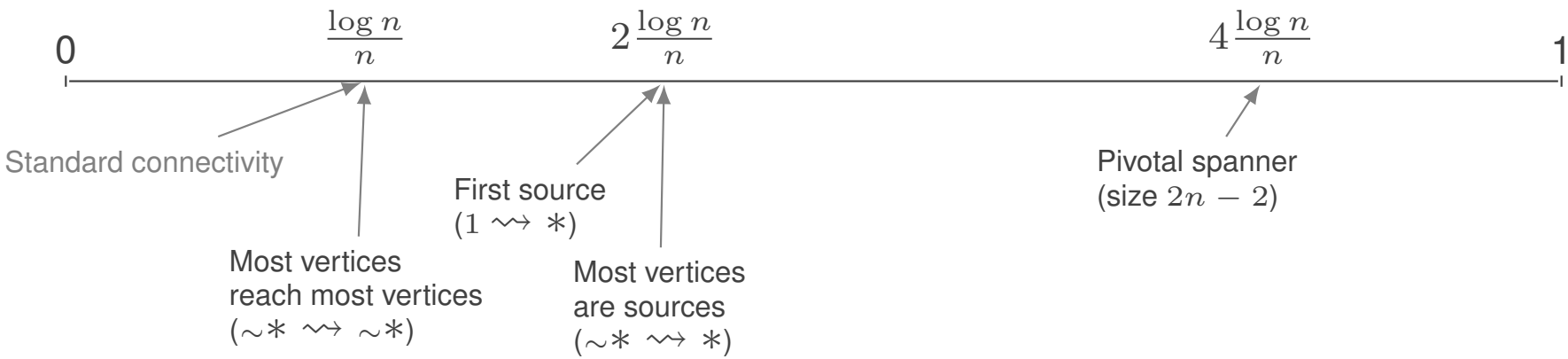
Connectivity in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random happy temporal graphs:

1. Pick an Erdös-Rényi $G \sim G_{n,p}$
2. Permute the edges randomly, interpret as (unique) presence time



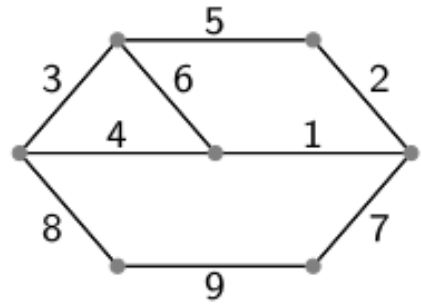
Timeline for p (as $n \to \infty$):



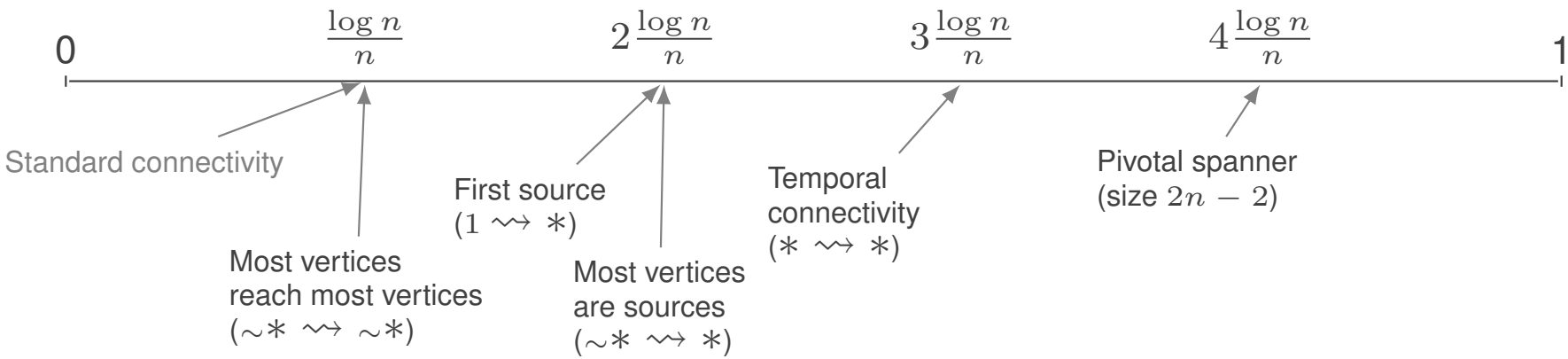
Connectivity in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random happy temporal graphs:

1. Pick an Erdös-Rényi $G \sim G_{n,p}$
2. Permute the edges randomly, interpret as (unique) presence time



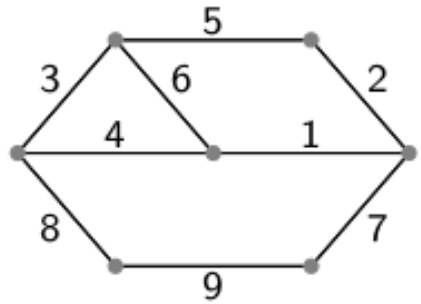
Timeline for p (as $n \to \infty$):



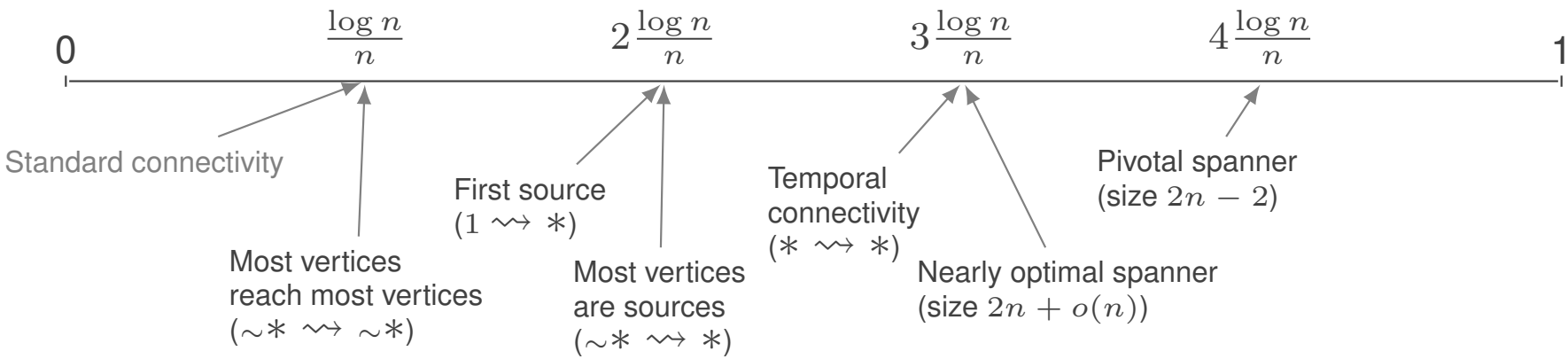
Connectivity in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random happy temporal graphs:

1. Pick an Erdös-Rényi $G \sim G_{n,p}$
2. Permute the edges randomly, interpret as (unique) presence time



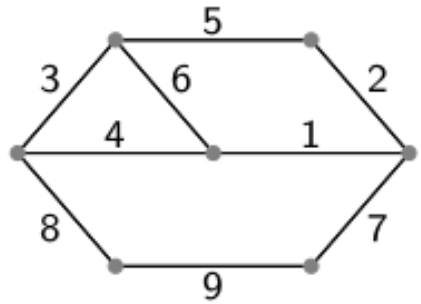
Timeline for p (as $n \to \infty$):



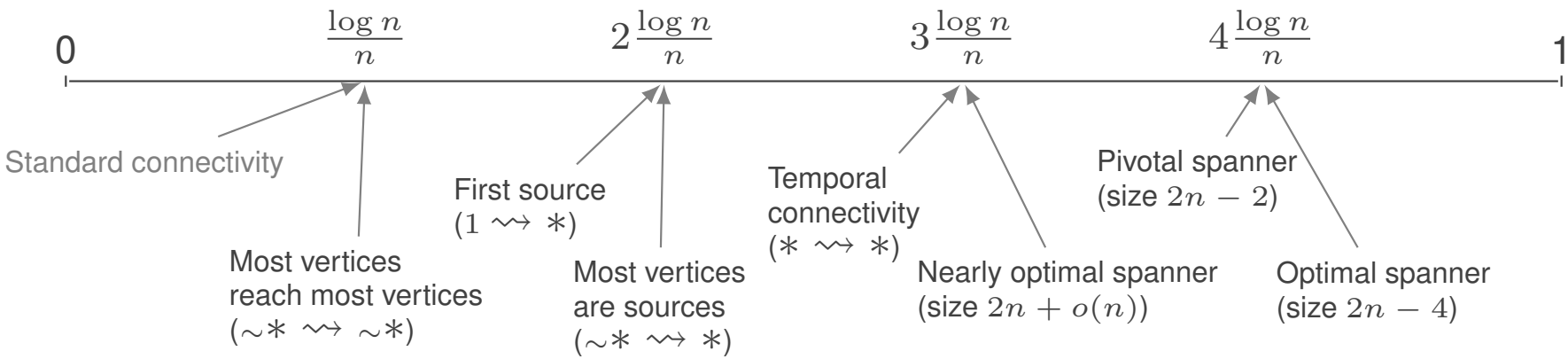
Connectivity in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random happy temporal graphs:

1. Pick an Erdös-Rényi $G \sim G_{n,p}$
2. Permute the edges randomly, interpret as (unique) presence time



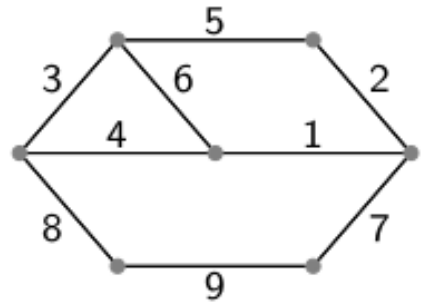
Timeline for p (as $n \to \infty$):



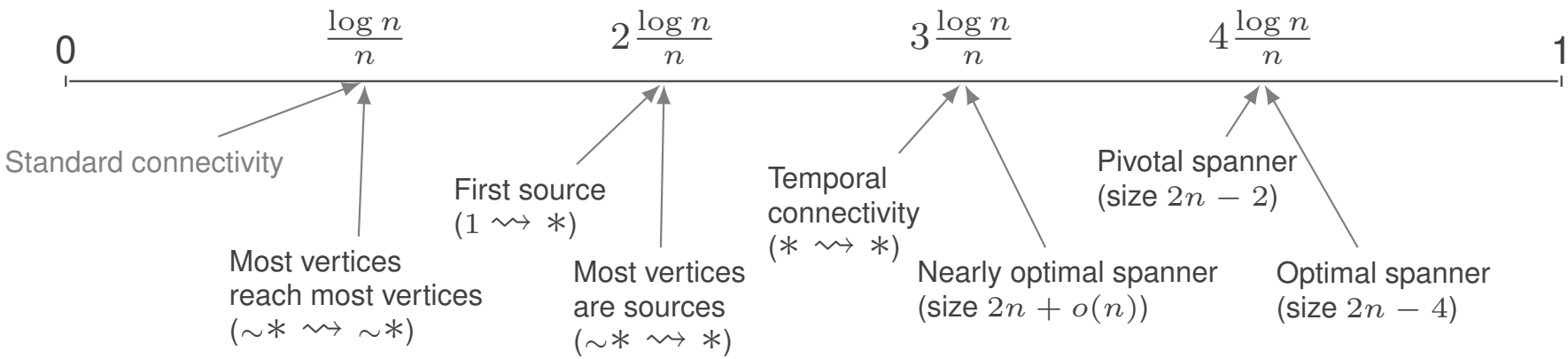
Connectivity in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random happy temporal graphs:

1. Pick an Erdös-Rényi $G \sim G_{n,p}$
2. Permute the edges randomly, interpret as (unique) presence time



Timeline for p (as $n \to \infty$):

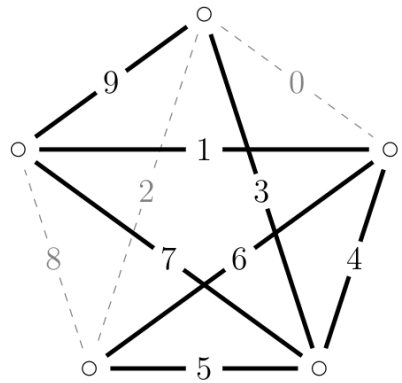


All the thresholds are sharp.

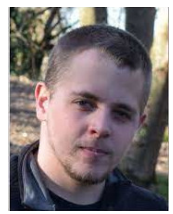
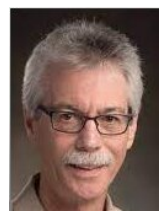
(sharp: $\exists \epsilon(n) = o(1)$, not true at $(1 - \epsilon(n))p$, true at $(1 + \epsilon(n))p$)

Good news 2:

Temporal cliques admit sparse spanners



(with)

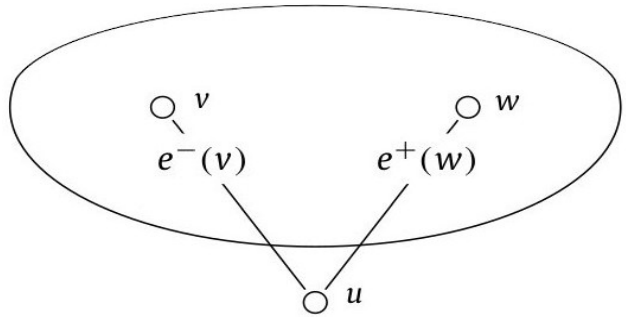
Dismountability

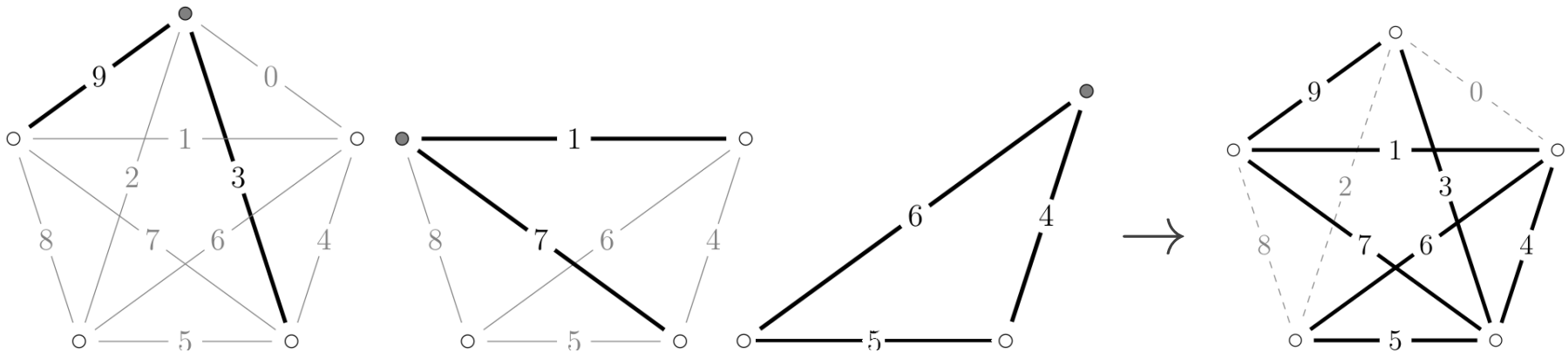
Find a node u s.t. :

- $uv =$ min edge of v
- $uw =$ max edge of w

Then `spanner(`$\mathcal{G}$`)` := `spanner(`$\mathcal{G}[V \setminus u]$`)` + uv + uw

$\rightarrow$ Recurse.





spanner of size $2n - 3$.

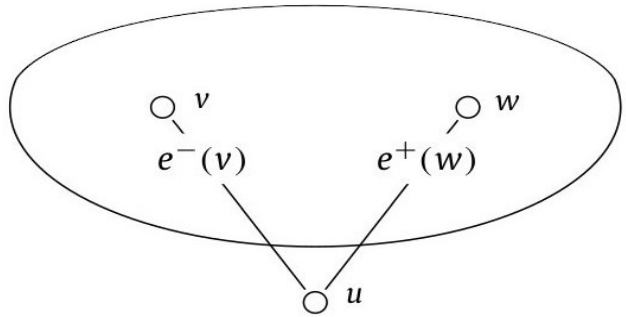
Dismountability

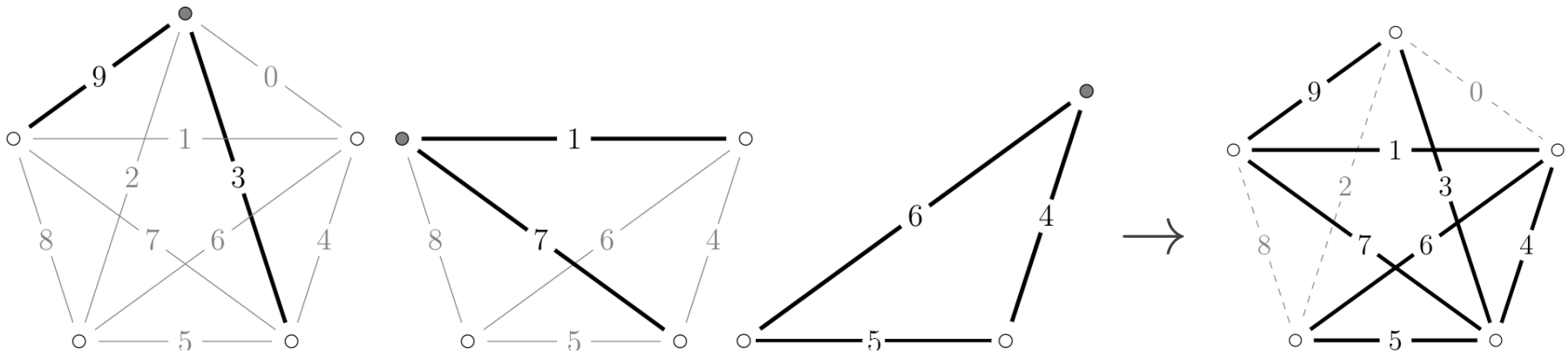
Find a node u s.t. :

- uv = min edge of v
- uw = max edge of w

Then `spanner(`$\mathcal{G}$`)` := `spanner(`$\mathcal{G}[V \setminus u]$`)` + uv + uw

$\rightarrow$ Recurse.





Not always feasible.

spanner of size $2n - 3$.

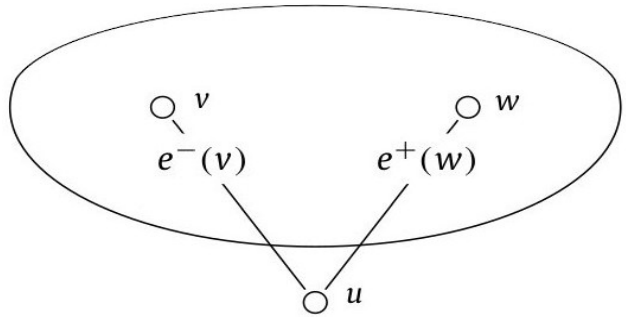
Dismountability

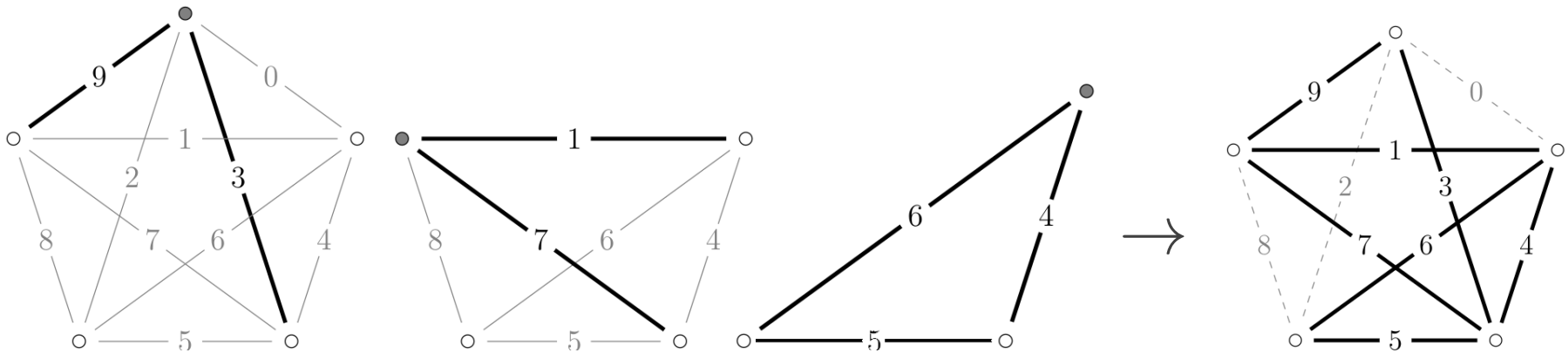
Find a node u s.t. :

- uv = min edge of v
- uw = max edge of w

Then `spanner(`$\mathcal{G}$`)` := `spanner(`$\mathcal{G}[V \setminus u]$`)` + uv + uw

$\rightarrow$ Recurse.





Not always feasible.

spanner of size $2n - 3$.

Relaxed version: k-hop dismountability

- Temporal paths $u \rightsquigarrow v$ ending at $e^-(v)$ and $w \rightsquigarrow u$ starting at $e^+(w)$

Select these $2k$ edges, then recurse $\rightarrow$ $O(n)$-spanner if k constant.

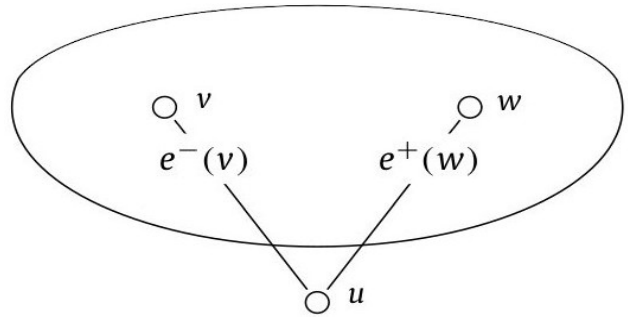
Dismountability

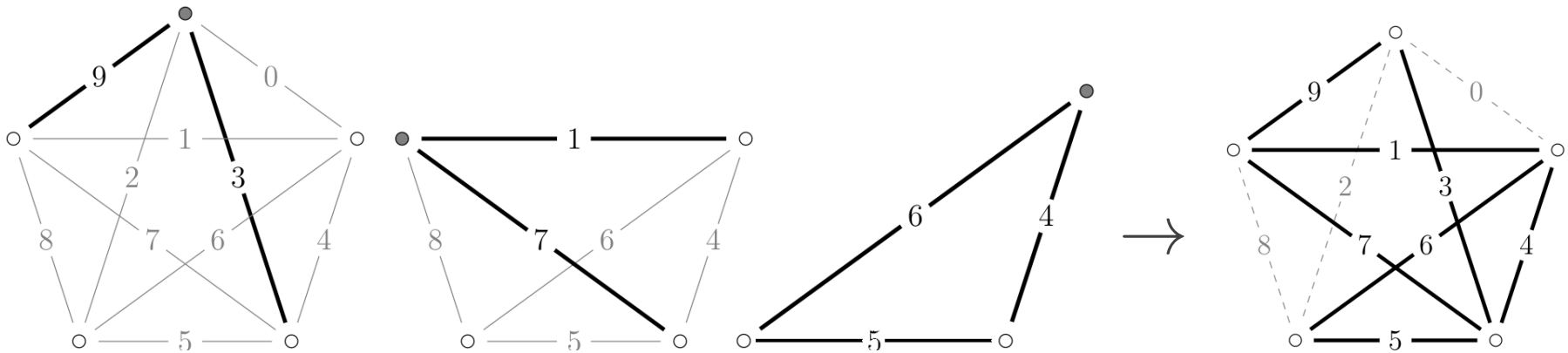
Find a node u s.t. :

- uv = min edge of v
- uw = max edge of w

Then `spanner(`$\mathcal{G}$`)` := `spanner(`$\mathcal{G}[V \setminus u]$`)` + uv + uw

$\rightarrow$ Recurse.





Not always feasible.

spanner of size $2n - 3$.

Relaxed version: k-hop dismountability

- Temporal paths $u \rightsquigarrow v$ ending at $e^-(v)$ and $w \rightsquigarrow u$ starting at $e^+(w)$

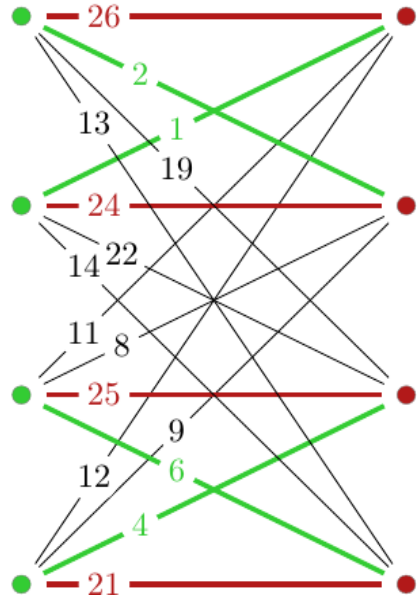
Select these $2k$ edges, then recurse $\rightarrow$ $O(n)$-spanner if k constant.

Not always feasible, but...

What if dismountability fails?

If $\mathcal{G}$ is neither 1-hop nor 2-hop dismountable, then the following is guaranteed:

- Complete bipartite graph $\mathcal{H} \subseteq \mathcal{G}$ ($n/2$ vertices in each part)
- Min edges of green nodes form a matching
- Max edges of red nodes form a matching
- Both matchings are disjoint
- **A spanner of $\mathcal{H}$ is a spanner of $\mathcal{G}$**

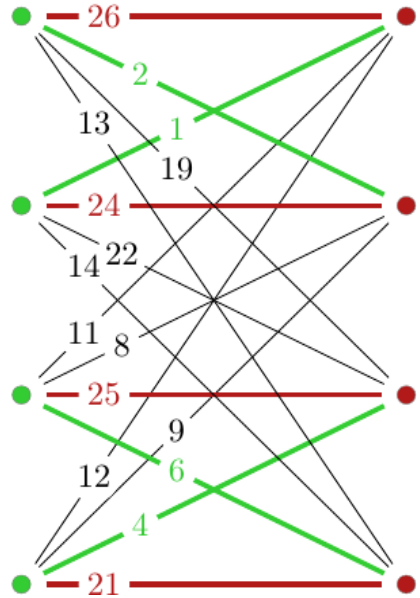


What if dismountability fails?

If $\mathcal{G}$ is neither 1-hop nor 2-hop dismountable, then the following is guaranteed:

- Complete bipartite graph $\mathcal{H} \subseteq \mathcal{G}$ ($n/2$ vertices in each part)
- Min edges of green nodes form a matching
- Max edges of red nodes form a matching
- Both matchings are disjoint
- **A spanner of $\mathcal{H}$ is a spanner of $\mathcal{G}$**

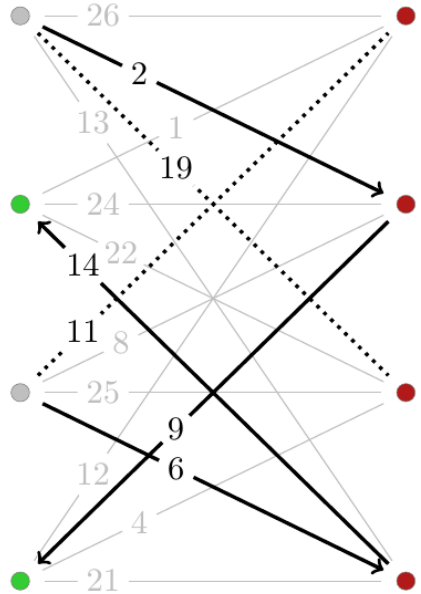
New goal: $\rightarrow$ Sparsify $\mathcal{H}$.



What if dismountability fails?

If $\mathcal{G}$ is neither 1-hop nor 2-hop dismountable, then the following is guaranteed:

- Complete bipartite graph $\mathcal{H} \subseteq \mathcal{G}$ ($n/2$ vertices in each part)
- Min edges of green nodes form a matching
- Max edges of red nodes form a matching
- Both matchings are disjoint
- **A spanner of $\mathcal{H}$ is a spanner of $\mathcal{G}$**



New goal: $\rightarrow$ Sparsify $\mathcal{H}$.

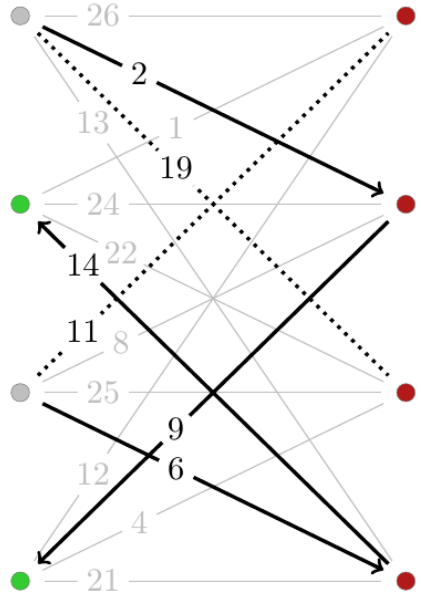
Main lemma:

Half of the green nodes can be iteratively removed, at doubling cost.
Repeat $\log n$ times.

What if dismountability fails?

If $\mathcal{G}$ is neither 1-hop nor 2-hop dismountable, then the following is guaranteed:

- Complete bipartite graph $\mathcal{H} \subseteq \mathcal{G}$ ($n/2$ vertices in each part)
- Min edges of green nodes form a matching
- Max edges of red nodes form a matching
- Both matchings are disjoint
- **A spanner of $\mathcal{H}$ is a spanner of $\mathcal{G}$**



New goal: $\rightarrow$ Sparsify $\mathcal{H}$.

Main lemma:

Half of the green nodes can be iteratively removed, at doubling cost.
Repeat $\log n$ times.

$\rightarrow$ Spanners of size $O(n \log n)$ always exist.

Summary and open questions

Algorithmic

- Complexity of MIN-SPANNER in happy graphs?

Summary and open questions

Algorithmic

- Complexity of MIN-SPANNER in happy graphs?

Structural

- Do cliques admit spanners of size $O(n)$?
- Do cliques admit spanners of size $\boldsymbol{2n-3}$?
- What else than cliques?

Summary and open questions

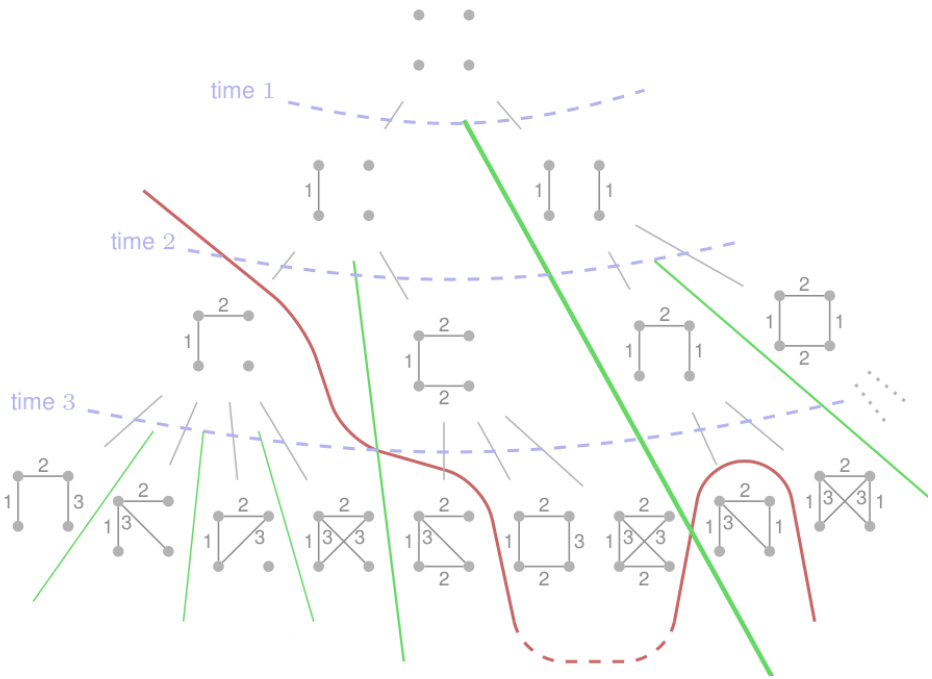
Algorithmic

- Complexity of MIN-SPANNER in happy graphs?

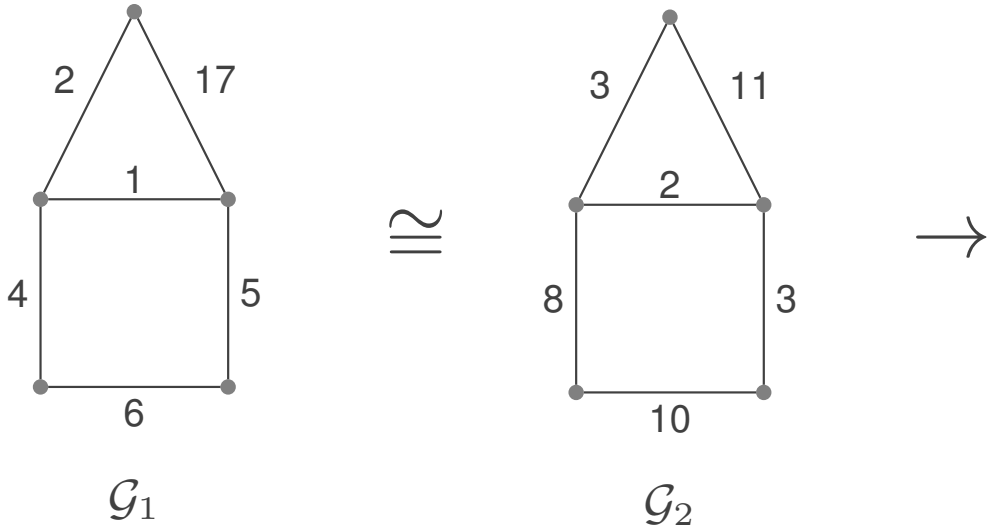
Structural

- Do cliques admit spanners of size $O(n)$?
- **Do cliques admit spanners of size $2n - 3$?**
- What else than cliques?

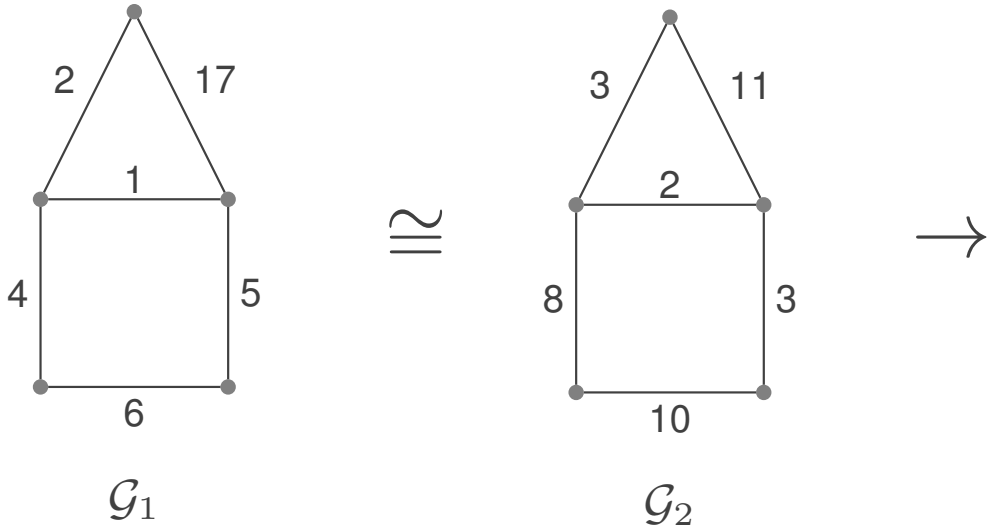
Enumeration of happy temporal graphs



Equivalence based on reachability

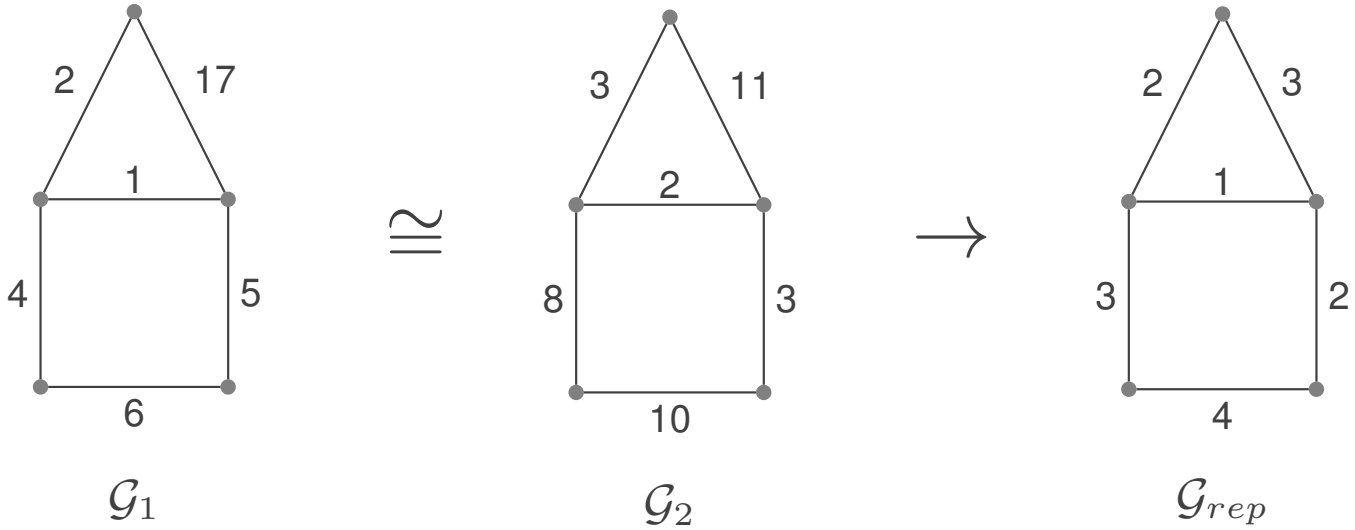


Equivalence based on reachability



How to capture this equivalence?

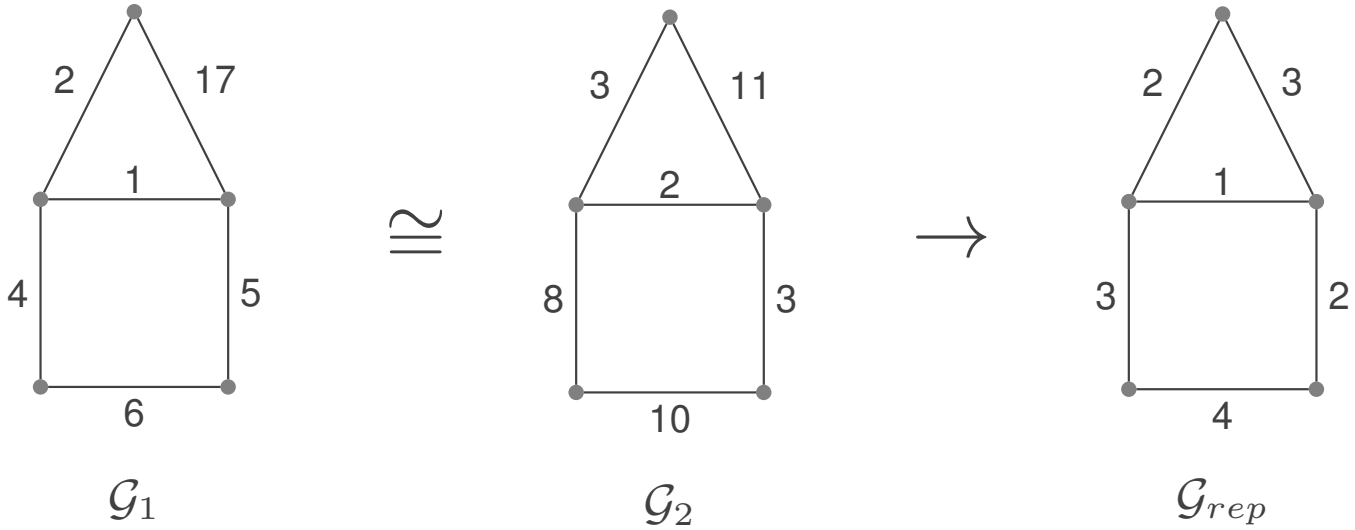
Equivalence based on reachability



How to capture this equivalence?

- Canonical representatives

Equivalence based on reachability



How to capture this equivalence?

- Canonical representatives

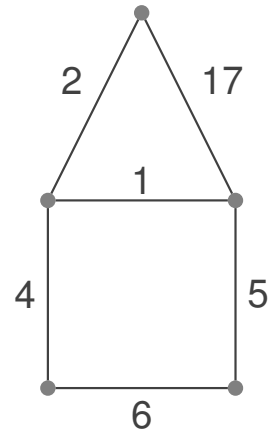
Good news:

- Finite number
- Canonization, isomorphism testing, and generators of the automorphism group all computable in *polynomial time*.

Properties of representatives

Canonization

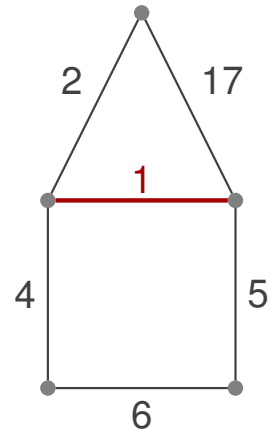
1. Find minimal edges
2. Give smallest available time
3. Repeat



Properties of representatives

Canonization

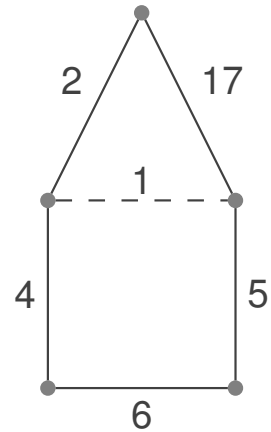
1. Find minimal edges
2. Give smallest available time
3. Repeat



Properties of representatives

Canonization

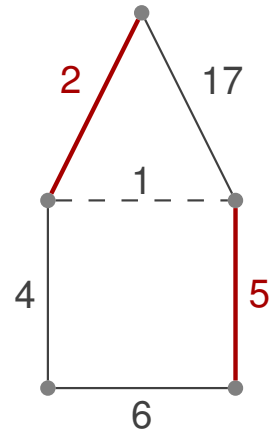
1. Find minimal edges
2. Give smallest available time
3. Repeat



Properties of representatives

Canonization

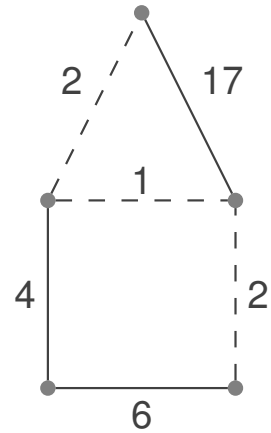
1. Find minimal edges
2. Give smallest available time
3. Repeat



Properties of representatives

Canonization

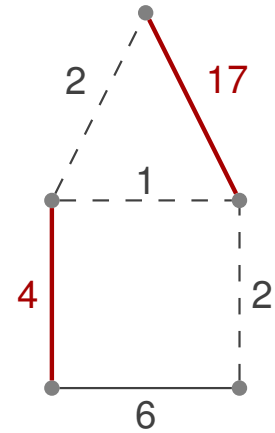
1. Find minimal edges
2. Give smallest available time
3. Repeat



Properties of representatives

Canonization

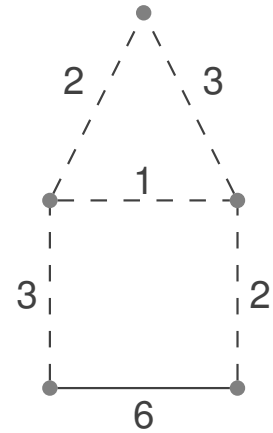
1. Find minimal edges
2. Give smallest available time
3. Repeat



Properties of representatives

Canonization

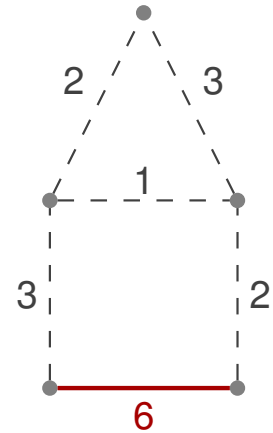
1. Find minimal edges
2. Give smallest available time
3. Repeat



Properties of representatives

Canonization

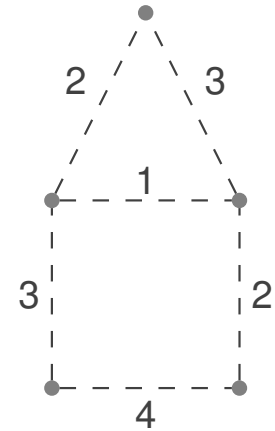
1. Find minimal edges
2. Give smallest available time
3. Repeat



Properties of representatives

Canonization

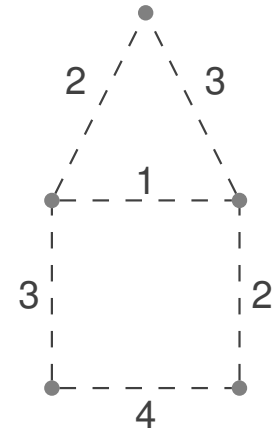
1. Find minimal edges
2. Give smallest available time
3. Repeat



Properties of representatives

Canonization

1. Find minimal edges
2. Give smallest available time
3. Repeat



Contiguity Lemma: If an edge is labeled $t > 1$, then an adjacent edge is labeled $t - 1$.

Isomorphism testing

Input: Two happy graphs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

Isomorphism testing

Input: Two happy graphs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

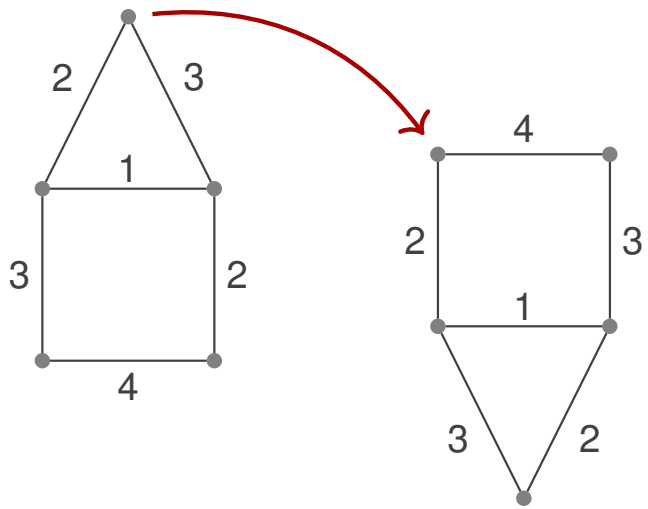
1. Canonize them
2. Test for edge-colored (classical) isomorphism

Isomorphism testing

Input: Two happy graphs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

1. Canonize them
2. Test for edge-colored (classical) isomorphism



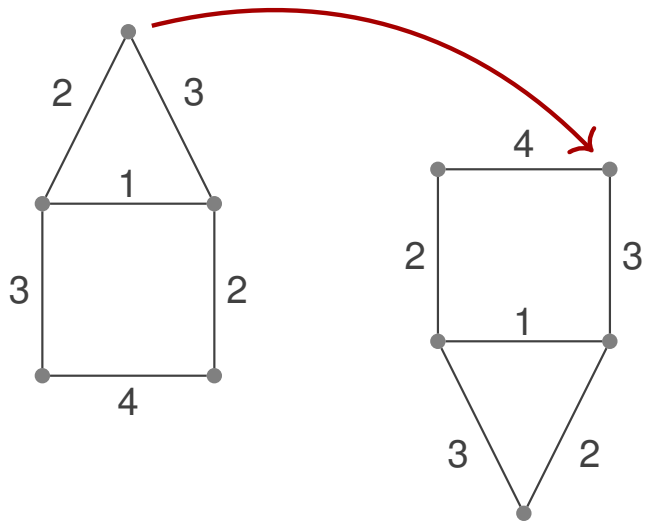
1. Fix an arbitrary vertex v_1 of G_1
2. Try to send it to a vertex v_2 of G_2
3. If OK, answer `YES`
4. If not, try the next vertex of G_2 (or answer `NO` if none remain)

Isomorphism testing

Input: Two happy graphs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

1. Canonize them
2. Test for edge-colored (classical) isomorphism



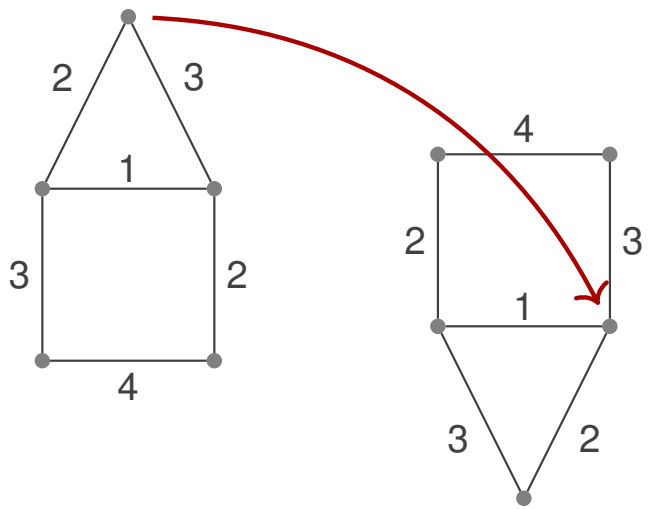
1. Fix an arbitrary vertex v_1 of G_1
2. Try to send it to a vertex v_2 of G_2
3. If OK, answer `YES`
4. If not, try the next vertex of G_2 (or answer `NO` if none remain)

Isomorphism testing

Input: Two happy graphs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

1. Canonize them
2. Test for edge-colored (classical) isomorphism



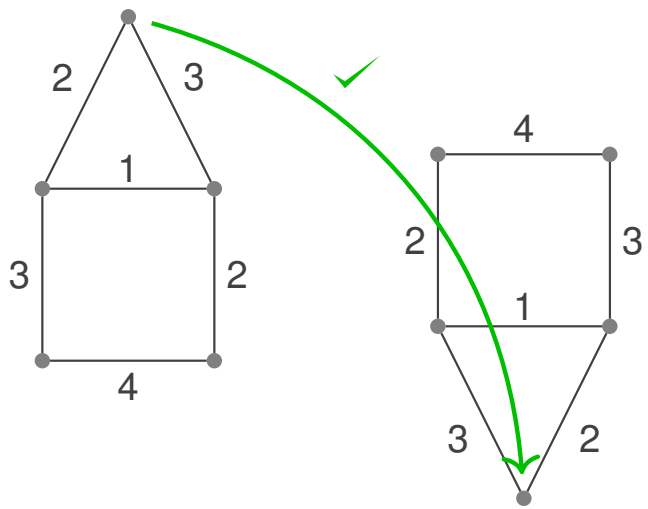
1. Fix an arbitrary vertex v_1 of G_1
2. Try to send it to a vertex v_2 of G_2
3. If OK, answer `YES`
4. If not, try the next vertex of G_2 (or answer `NO` if none remain)

Isomorphism testing

Input: Two happy graphs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

1. Canonize them
2. Test for edge-colored (classical) isomorphism



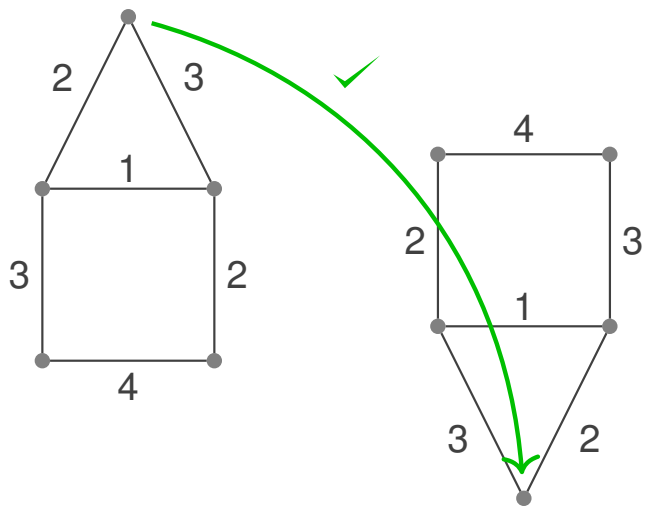
1. Fix an arbitrary vertex v_1 of G_1
2. Try to send it to a vertex v_2 of G_2
3. If OK, answer `YES`
4. If not, try the next vertex of G_2 (or answer `NO` if none remain)

Isomorphism testing

Input: Two happy graphs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

1. Canonize them
2. Test for edge-colored (classical) isomorphism



1. Fix an arbitrary vertex v_1 of G_1
2. Try to send it to a vertex v_2 of G_2
3. If OK, answer `YES`
4. If not, try the next vertex of G_2 (or answer `NO` if none remain)

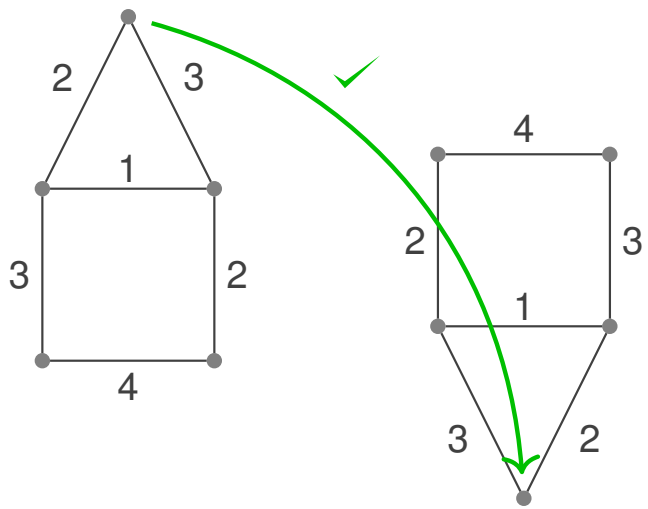
Key observation: when trying to send v_1 to v_2, the mapping unfolds recursively without any choice (due to *properness*) $\rightarrow$ passes or fails in polynomial time.

Isomorphism testing

Input: Two happy graphs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

1. Canonize them
2. Test for edge-colored (classical) isomorphism



1. Fix an arbitrary vertex v_1 of G_1
2. Try to send it to a vertex v_2 of G_2
3. If OK, answer `YES`
4. If not, try the next vertex of G_2 (or answer `NO` if none remain)

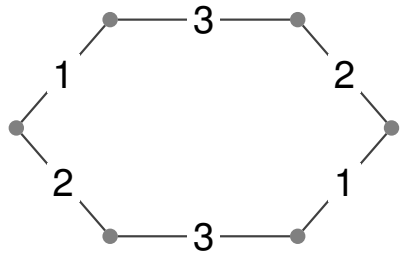
Key observation: when trying to send v_1 to v_2, the mapping unfolds recursively without any choice (due to *properness*) $\rightarrow$ passes or fails in polynomial time.

Remark: Also feasible using Babai & Luks machinery (1983)

Automorphisms

Case 1: The footprint is connected.

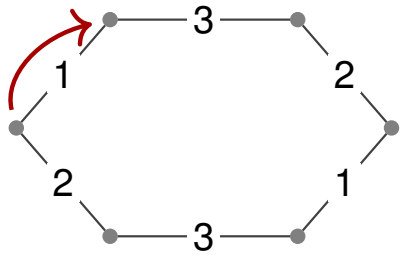
$\rightarrow$ Same strategy as for isomorphism.



Automorphisms

Case 1: The footprint is connected.

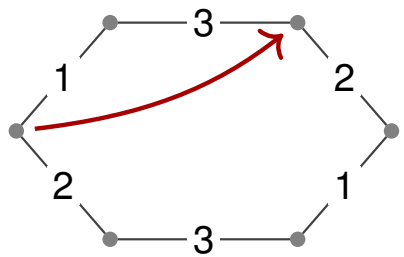
→ Same strategy as for isomorphism.



Automorphisms

Case 1: The footprint is connected.

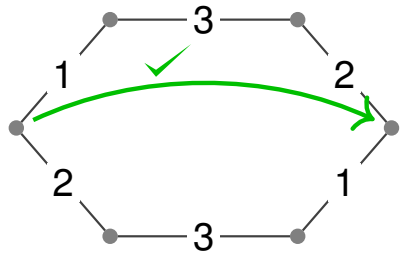
$\rightarrow$ Same strategy as for isomorphism.



Automorphisms

Case 1: The footprint is connected.

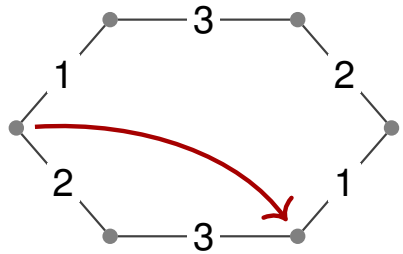
$\rightarrow$ Same strategy as for isomorphism.



Automorphisms

Case 1: The footprint is connected.

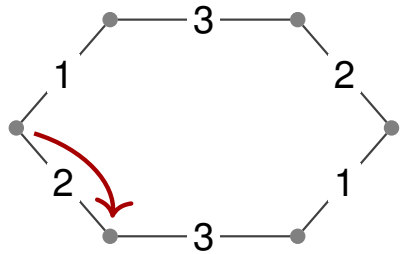
$\rightarrow$ Same strategy as for isomorphism.



Automorphisms

Case 1: The footprint is connected.

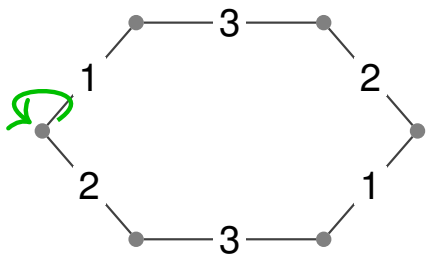
→ Same strategy as for isomorphism.



Automorphisms

Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.

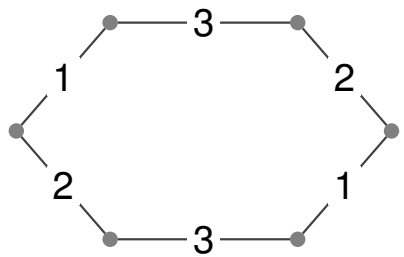


At most n automorphisms!

Automorphisms

Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.



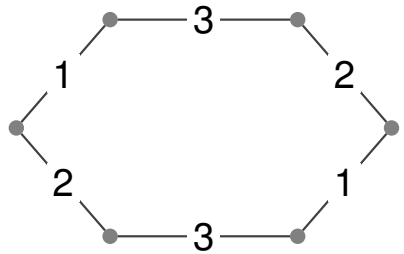
Case 2: It is not.

(complement trick does not works...)

Automorphisms

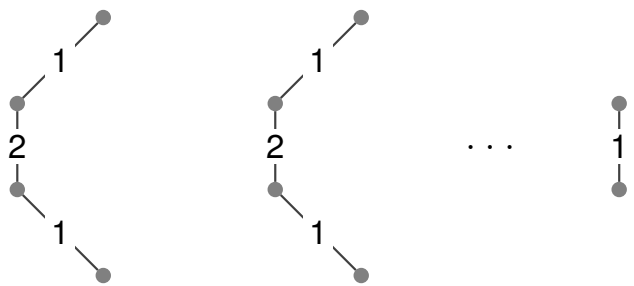
Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: It is not.

(complement trick does not works...)

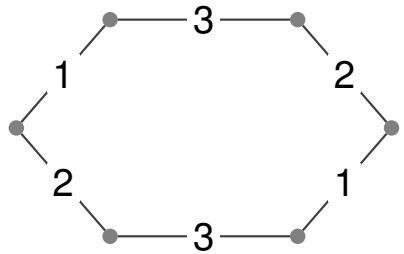


1. Find underlying components
2. Find isomorphisms among them
3. Find automorphisms within each (extended to $\mathcal{G}$)

Automorphisms

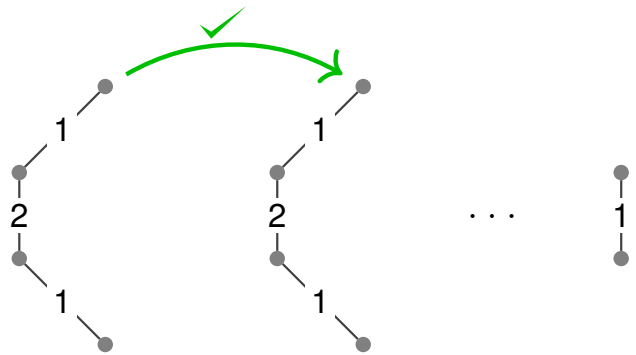
Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: It is not.

(complement trick does not works...)

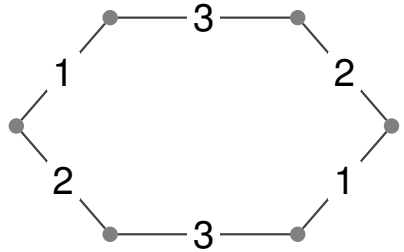


1. Find underlying components
2. Find isomorphisms among them
3. Find automorphisms within each (extended to $\mathcal{G}$)

Automorphisms

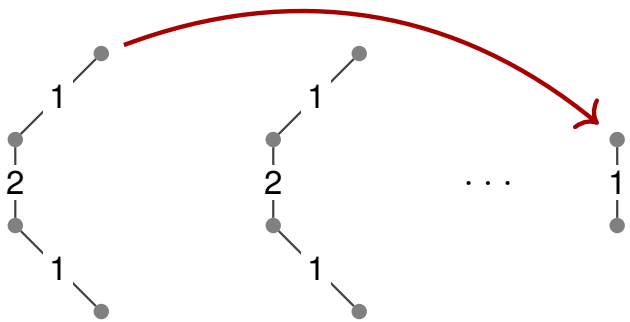
Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: It is not.

(complement trick does not works...)

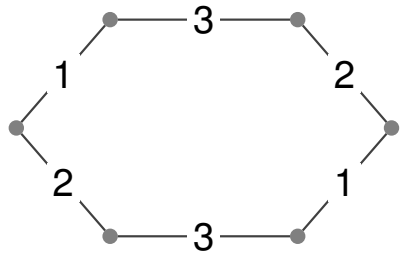


1. Find underlying components
2. Find isomorphisms among them
3. Find automorphisms within each (extended to $\mathcal{G}$)

Automorphisms

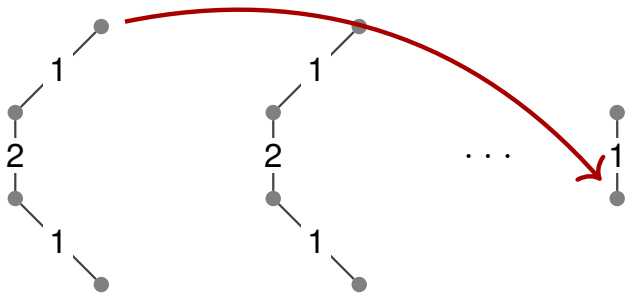
Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: It is not.

(complement trick does not works...)

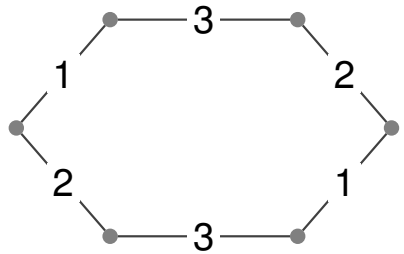


1. Find underlying components
2. Find isomorphisms among them
3. Find automorphisms within each (extended to $\mathcal{G}$)

Automorphisms

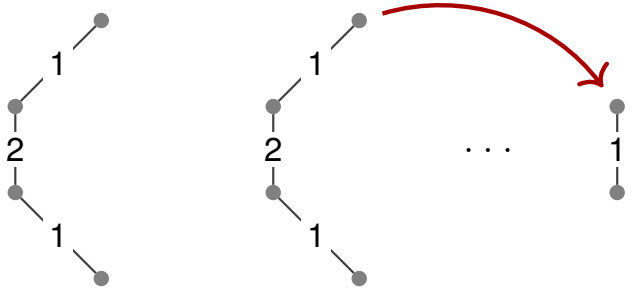
Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: It is not.

(complement trick does not works...)

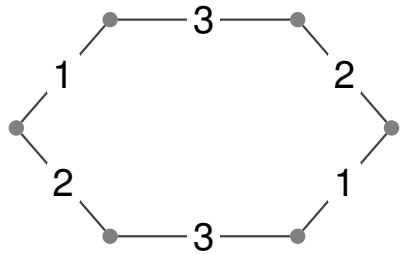


1. Find underlying components
2. Find isomorphisms among them
3. Find automorphisms within each (extended to $\mathcal{G}$)

Automorphisms

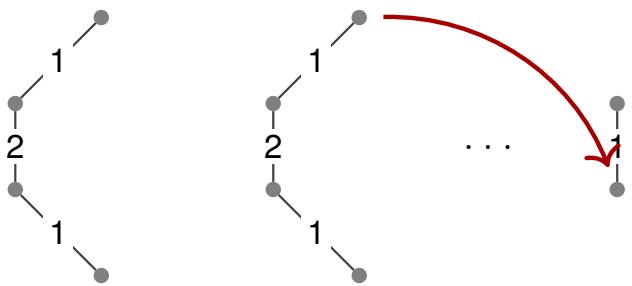
Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: It is not.

(complement trick does not works...)

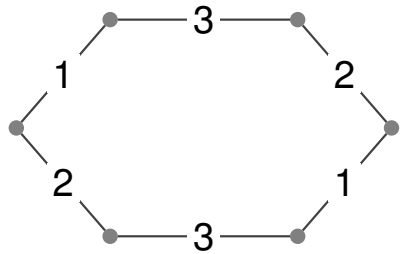


1. Find underlying components
2. Find isomorphisms among them
3. Find automorphisms within each (extended to $\mathcal{G}$)

Automorphisms

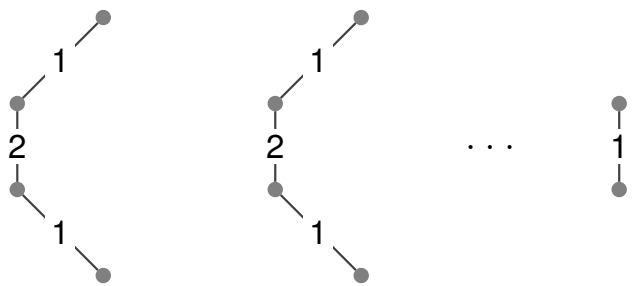
Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: It is not.

(complement trick does not works...)

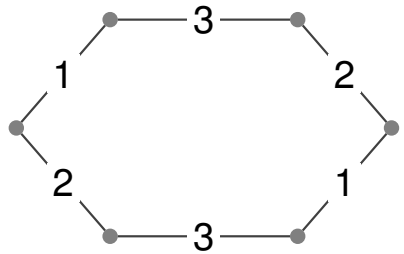


1. Find underlying components
2. Find isomorphisms among them
3. Find automorphisms within each (extended to $\mathcal{G}$)

Automorphisms

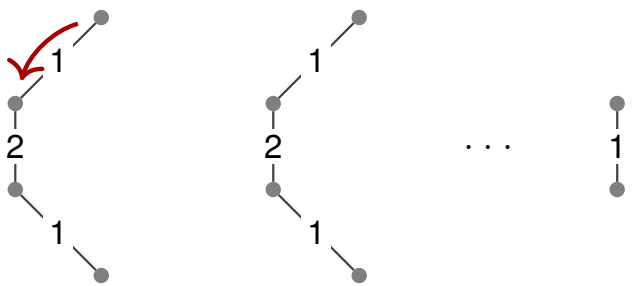
Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: It is not.

(complement trick does not works...)

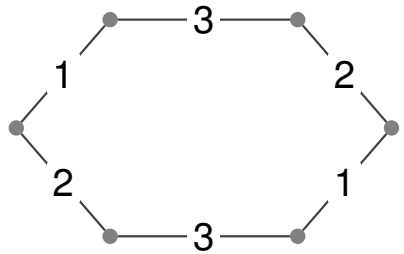


1. Find underlying components
2. Find isomorphisms among them
3. Find automorphisms within each (extended to $\mathcal{G}$)

Automorphisms

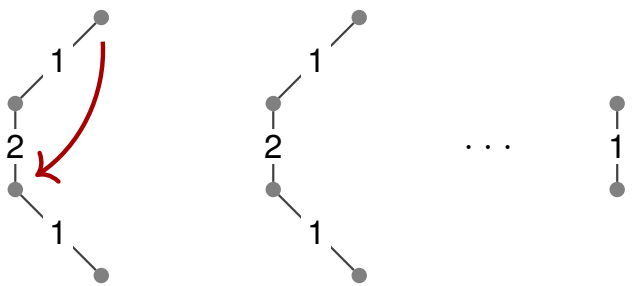
Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: It is not.

(complement trick does not works...)

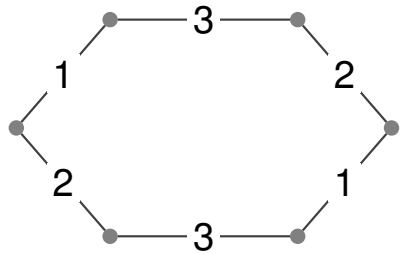


1. Find underlying components
2. Find isomorphisms among them
3. Find automorphisms within each (extended to $\mathcal{G}$)

Automorphisms

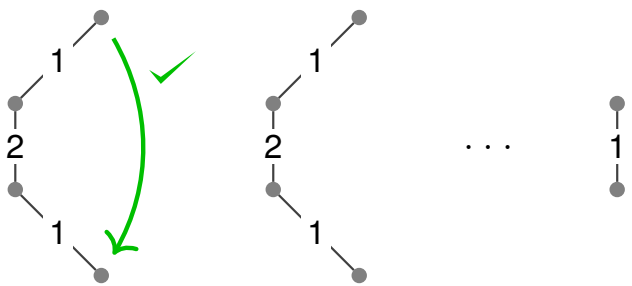
Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: It is not.

(complement trick does not works...)

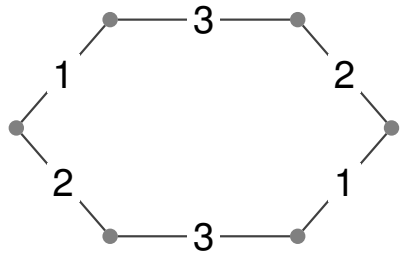


1. Find underlying components
2. Find isomorphisms among them
3. Find automorphisms within each (extended to $\mathcal{G}$)

Automorphisms

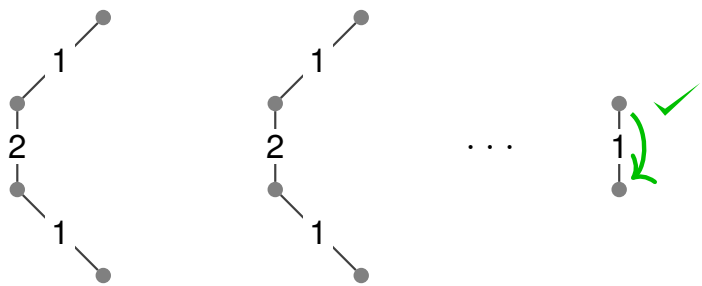
Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: It is not.

(complement trick does not works...)

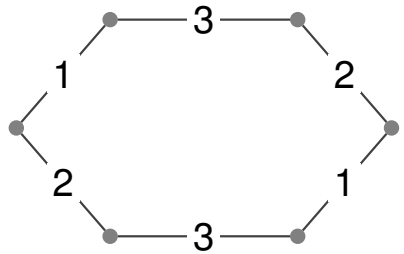


1. Find underlying components
2. Find isomorphisms among them
3. Find automorphisms within each (extended to $\mathcal{G}$)

Automorphisms

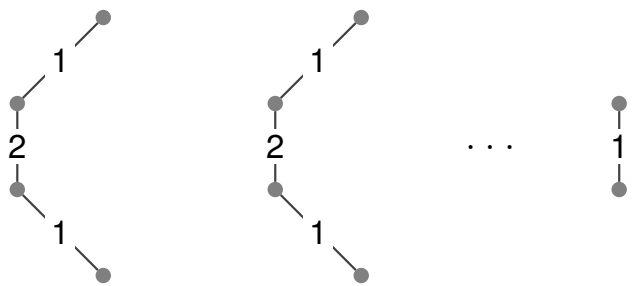
Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: It is not.

(complement trick does not works...)



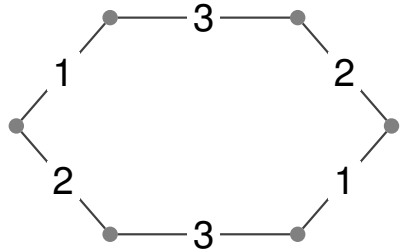
1. Find underlying components
2. Find isomorphisms among them
3. Find automorphisms within each (extended to $\mathcal{G}$)

Claim: $Aut(\mathcal{G}) = \langle$ isomorphisms + automorphisms $\rangle$

Automorphisms

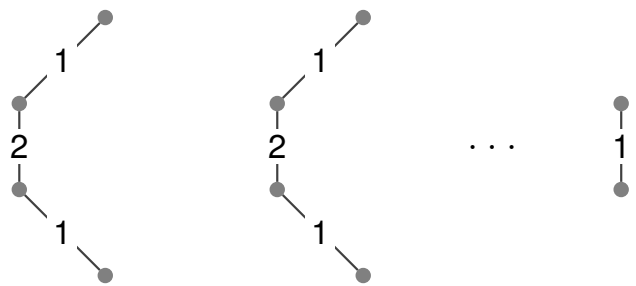
Case 1: The footprint is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: It is not.

(complement trick does not works...)



1. Find underlying components
2. Find isomorphisms among them
3. Find automorphisms within each (extended to $\mathcal{G}$)

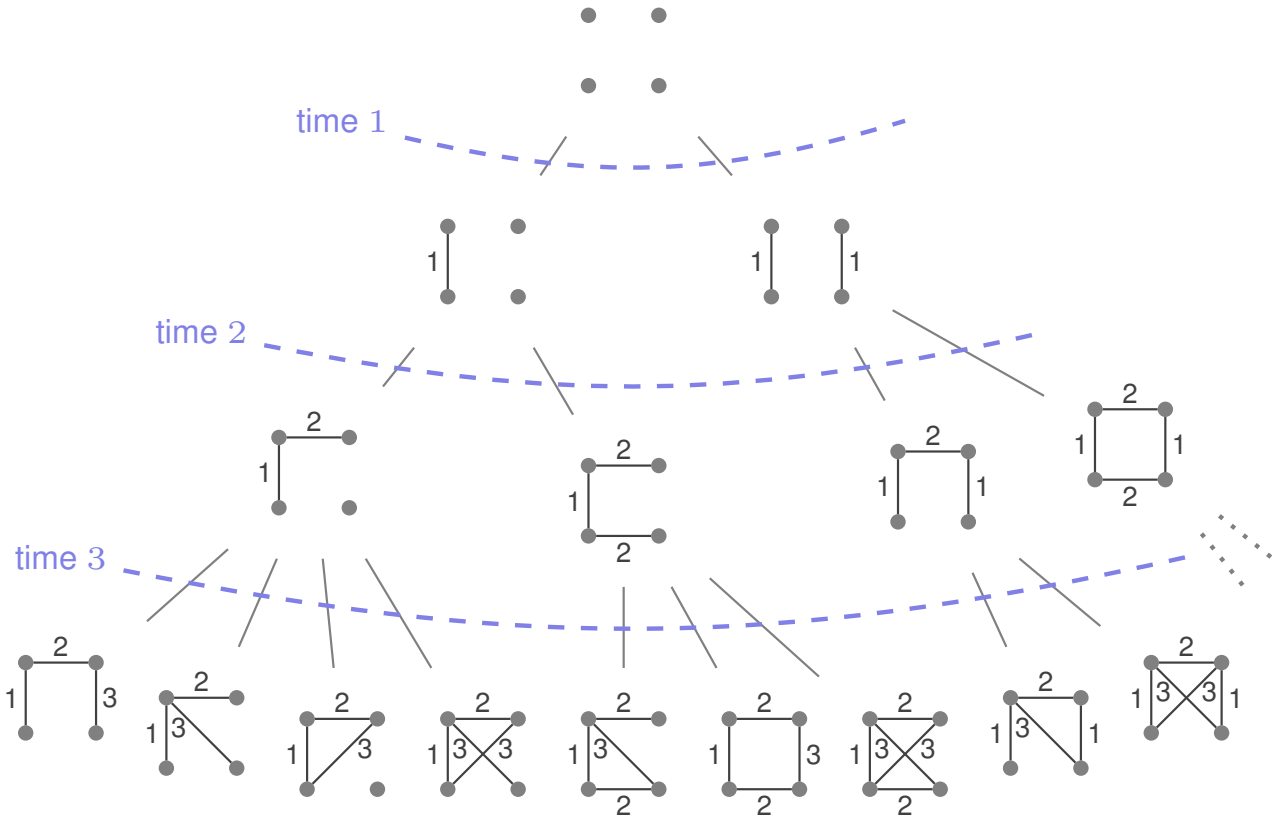
Claim: $Aut(\mathcal{G}) = \langle$ isomorphisms + automorphisms $\rangle$

$\rightarrow$ *Generators in polynomial time!*

Generation tree

Principle: One level = one time

$\rightarrow$ successors $=$ all possible ways to add the *next time*



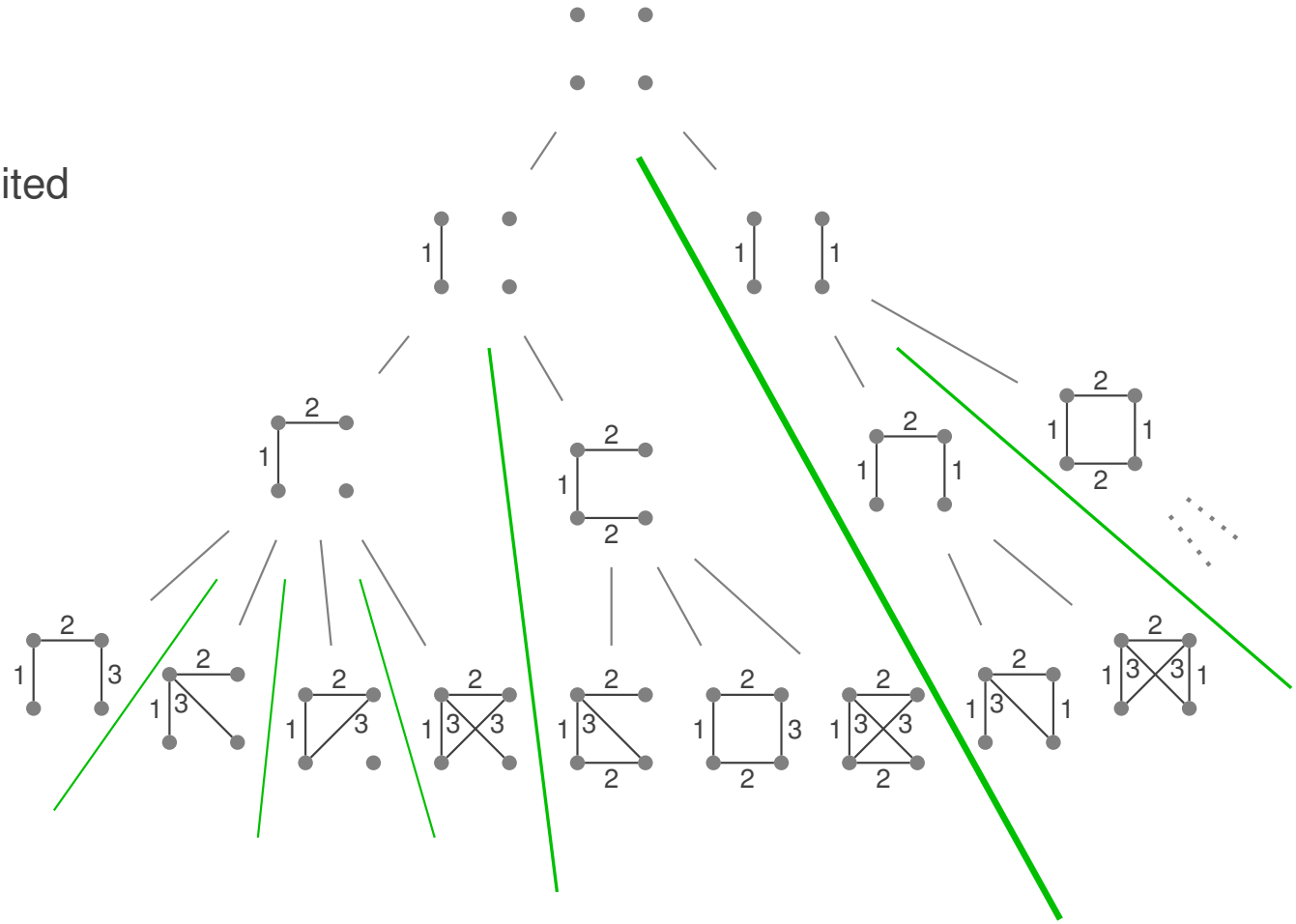
Generation tree

Principle: One level = one time

$\rightarrow$ successors $=$ all possible ways to add the *next time*

Key properties

1. Dissimilarity is inherited



$\downarrow$ Isomorphism types separated (forever)

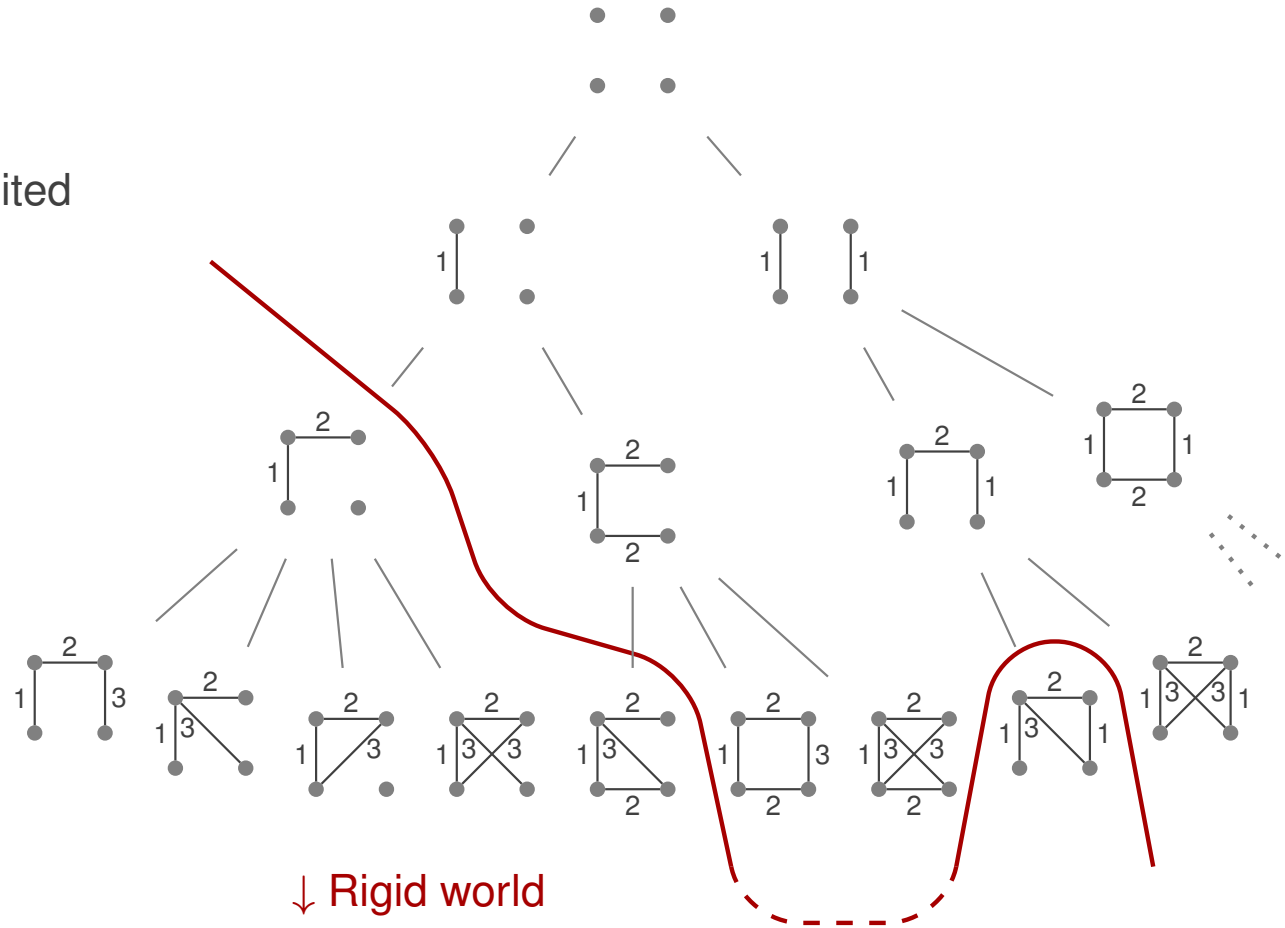
Generation tree

Principle: One level = one time

$\rightarrow$ successors $=$ all possible ways to add the *next time*

Key properties

1. Dissimilarity is inherited
2. Rigidity is inherited



How to compute successors?

Input: A happy representative $\mathcal{G}$, max time t

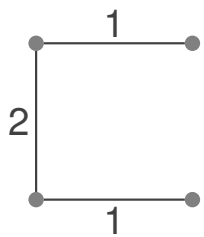
Output: Happy representatives extending $\mathcal{G}$ with time $t + 1$.

How to compute successors?

Input: A happy representative $\mathcal{G}$, max time t

Output: Happy representatives extending $\mathcal{G}$ with time $t + 1$.

First, find eligible *non-edges* (E.g. here, for time 3)

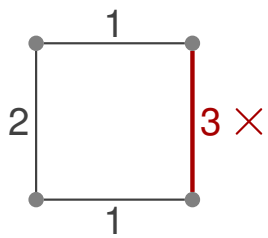


How to compute successors?

Input: A happy representative $\mathcal{G}$, max time t

Output: Happy representatives extending $\mathcal{G}$ with time $t + 1$.

First, find eligible *non-edges* (E.g. here, for time 3)



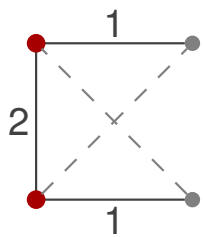
Contiguity lemma: t+1 must be adjacent to t

How to compute successors?

Input: A happy representative $\mathcal{G}$, max time t

Output: Happy representatives extending $\mathcal{G}$ with time $t + 1$.

First, find eligible *non-edges* (E.g. here, for time 3)



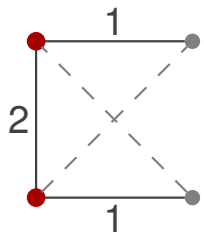
Contiguity lemma: t+1 must be adjacent to t

How to compute successors?

Input: A happy representative $\mathcal{G}$, max time t

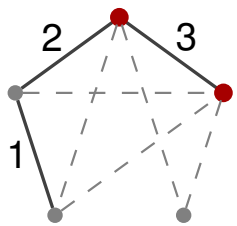
Output: Happy representatives extending $\mathcal{G}$ with time $t+1$.

First, find eligible *non-edges* (E.g. here, for time 3)



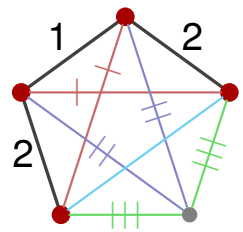
Contiguity lemma: t+1 must be adjacent to t

$\mathcal{G}$ is rigid



Two cases

$\mathcal{G}$ has symmetries

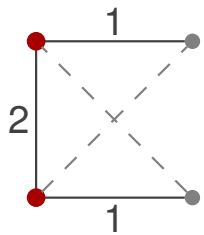


How to compute successors?

Input: A happy representative $\mathcal{G}$, max time t

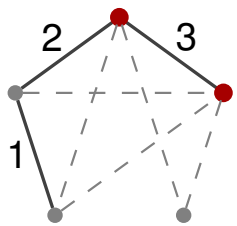
Output: Happy representatives extending $\mathcal{G}$ with time $t+1$.

First, find eligible *non-edges* (E.g. here, for time 3)



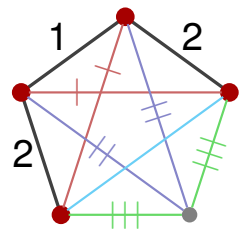
Contiguity lemma: t+1 must be adjacent to t

$\mathcal{G}$ is rigid



Two cases

$\mathcal{G}$ has symmetries



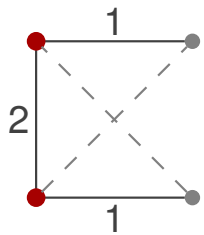
$\rightarrow$ Enumerate matchings among eligible *non-edges*.

How to compute successors?

Input: A happy representative $\mathcal{G}$, max time t

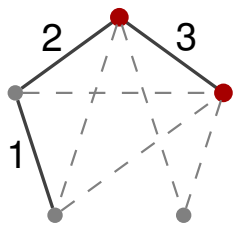
Output: Happy representatives extending $\mathcal{G}$ with time $t+1$.

First, find eligible *non-edges* (E.g. here, for time 3)



Contiguity lemma: t+1 must be adjacent to t

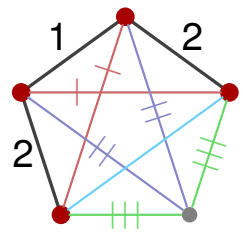
$\mathcal{G}$ is rigid



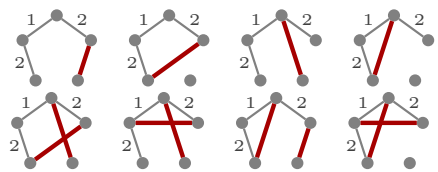
→ Enumerate matchings among eligible *non-edges*.

Two cases

$\mathcal{G}$ has symmetries



→ Enumerate matchings of eligible *non-edges* whose *multisets of orbits* are distinct



Back to the 2n-3 conjecture

Various implementations of the generator (`Python`, `Java`, `Julia`, `Rust`, **C++**)
`https://arnaudcasteigts.net/blog/stgen.html`

Do spanners of size $2n - 3$ always exist in happy cliques?

Back to the 2n-3 conjecture

Various implementations of the generator (`Python`, `Java`, `Julia`, `Rust`, **C++**)
`https://arnaudcasteigts.net/blog/stgen.html`

Do spanners of size $2n - 3$ always exist in happy cliques?

$\rightarrow$ True for $n \leq 8$.

Back to the 2n-3 conjecture

Various implementations of the generator (`Python`, `Java`, `Julia`, `Rust`, **C++**)
`https://arnaudcasteigts.net/blog/stgen.html`

Do spanners of size $2n - 3$ always exist in happy cliques?

$\rightarrow$ True for $n \leq 8$.

Some numbers

n	# Happy graphs (representatives)	# Happy cliques (representatives)
3	4	1
4	62	20
5	15378	4524
6	89769096	23218501
7	13725757879376	3106952711040

Thanks!

(Battelle, Dec 3, 2023)