Spanners and connectivity problems in temporal graphs

Arnaud Casteigts

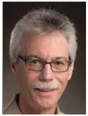
LaBRI, Université de Bordeaux

November 10, 2022

(Liverpool CS seminar)

Based on two joint works with:

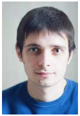
Jason Schoeters
(Le Havre)



Joseph Peters
(Vancouver)

Michael Raskin
(Munich)

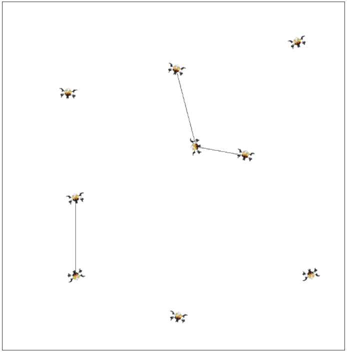
Malte Renken
(Berlin)



Viktor Zamaraev
(Liverpool)

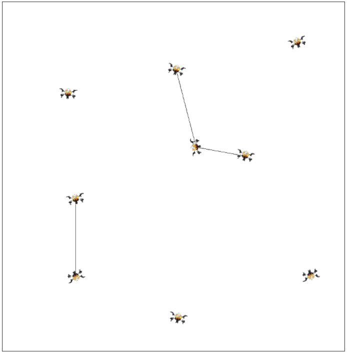
(Highly) dynamic networks?

Example of scenario

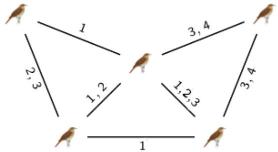


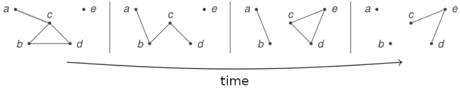
(Highly) dynamic networks?

Example of scenario



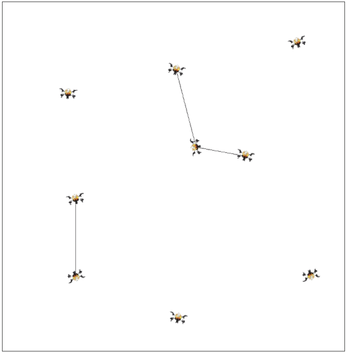
Modeling



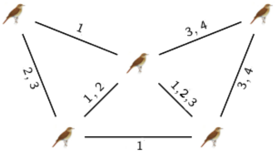


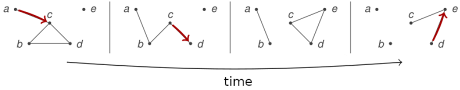
(Highly) dynamic networks?

Example of scenario



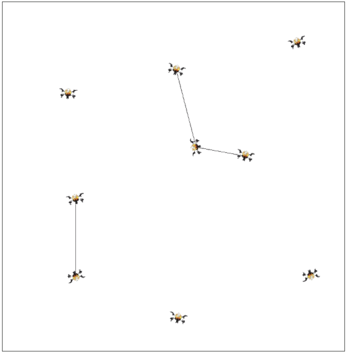
Modeling



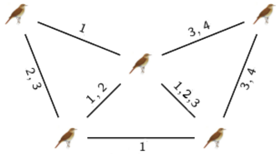


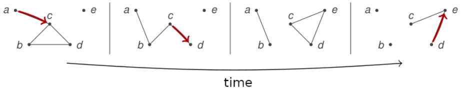
(Highly) dynamic networks?

Example of scenario



Modeling



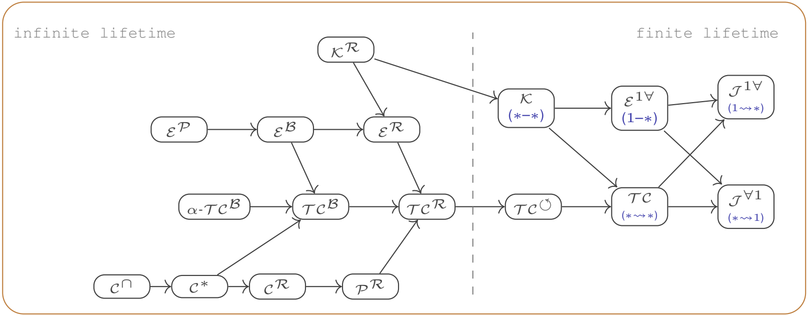


Properties:

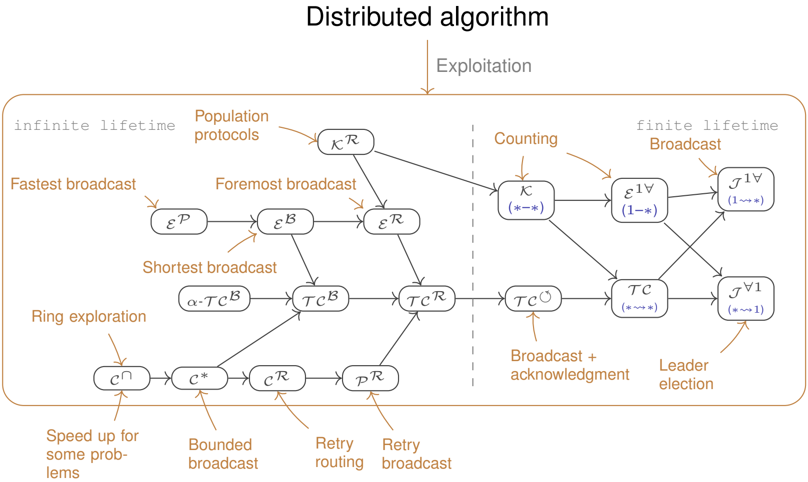
- Temporal connectivity? $\mathcal{TC}$
- Repeatedly? $\mathcal{TC}^{\mathcal{R}}$
- Recurrent links? $\mathcal{E}^{\mathcal{R}}$
- In bounded time? $\mathcal{E}^{\mathcal{B}}$
- ...

$\rightarrow$ Classes of temporal graphs

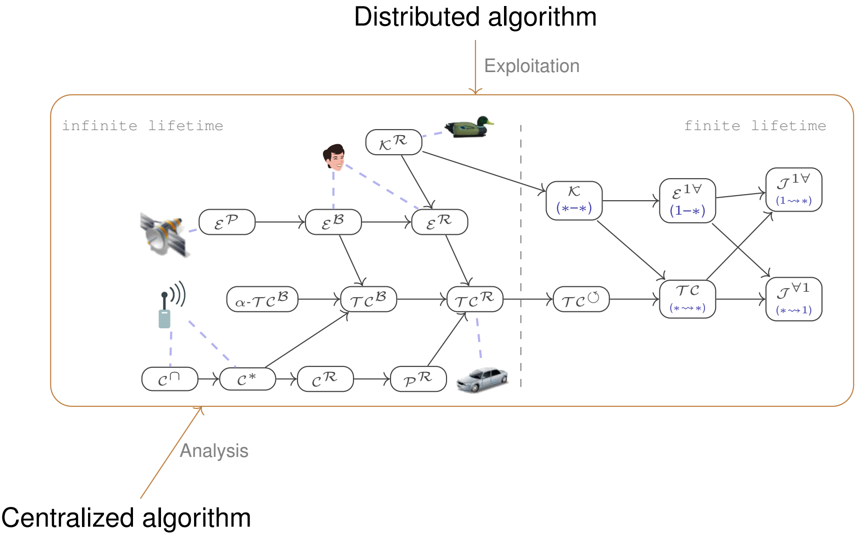
Some classes of temporal graphs



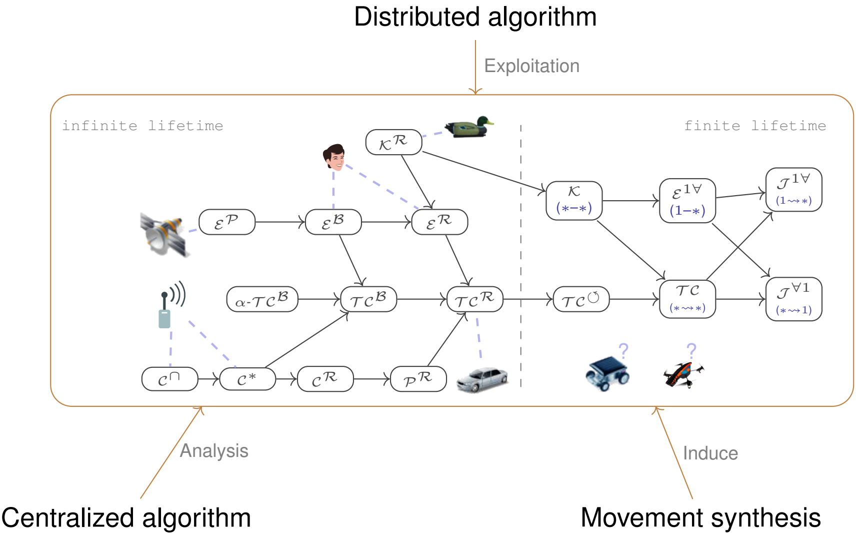
Some classes of temporal graphs



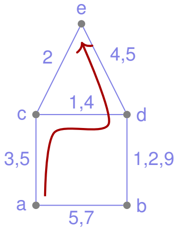
Some classes of temporal graphs



Some classes of temporal graphs



Temporal graphs for their own sake



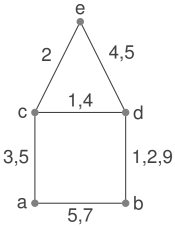
What does make them truly different?

Basic definitions

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

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

Example:



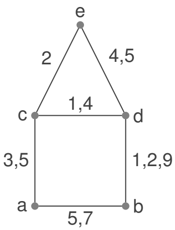
Basic definitions

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

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

footprint of $\mathcal{G}$ (the pair (V, E))

Example:



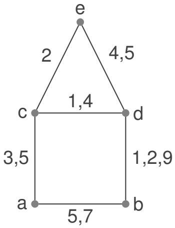
Basic definitions

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 *presence times* to edges.

footprint of $\mathcal{G}$

Example:



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

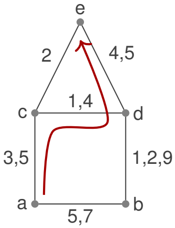
Basic definitions

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

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

footprint of $\mathcal{G}$ (refers to (V, E))

Example:



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

Temporal paths

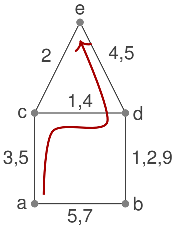
Basic definitions

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

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

footprint of $\mathcal{G}$ (pointing to (V, E))

Example:



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

Temporal paths

- Non-strict - ex: $\langle (a, c, 3), (c, d, 4), (d, e, 4) \rangle$ (*non-decreasing*)
- Strict - ex: $\langle (a, c, 3), (c, d, 4), (d, e, 5) \rangle$ (*increasing*)

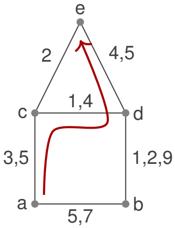
Basic definitions

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

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

footprint of $\mathcal{G}$

Example:



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

Temporal paths

- Non-strict - ex: $\langle (a, c, 3), (c, d, 4), (d, e, 4) \rangle$ (*non-decreasing*)
- Strict - ex: $\langle (a, c, 3), (c, d, 4), (d, e, 5) \rangle$ (*increasing*)

Temporal connectivity: $\exists$ temporal paths between all vertices.

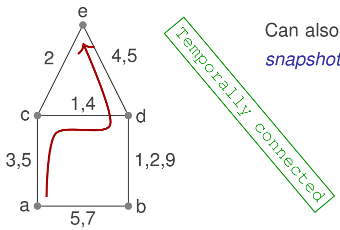
Basic definitions

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

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

footprint of $\mathcal{G}$

Example:



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

Temporal paths

- Non-strict - ex: $\langle (a,c,3),(c,d,4),(d,e,4) \rangle$ (*non-decreasing*)
- Strict - ex: $\langle (a,c,3),(c,d,4),(d,e,5) \rangle$ (*increasing*)

Temporal connectivity: $\exists$ temporal paths between all vertices.

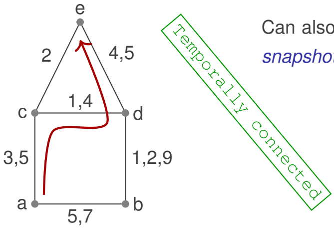
Basic definitions

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

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

footprint of $\mathcal{G}$

Example:



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

Temporal paths

- Non-strict - ex: $\langle (a, c, 3), (c, d, 4), (d, e, 4) \rangle$ (*non-decreasing*)
- Strict - ex: $\langle (a, c, 3), (c, d, 4), (d, e, 5) \rangle$ (*increasing*)

Temporal connectivity: $\exists$ temporal paths between all vertices.

$\to$ Warning: Reachability is non-symmetrical...

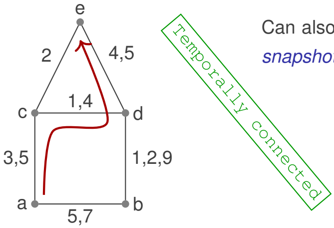
Basic definitions

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

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

footprint of $\mathcal{G}$ (the (V, E) part)

Example:



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

Temporal paths

- Non-strict - ex: $\langle (a, c, 3), (c, d, 4), (d, e, 4) \rangle$ (*non-decreasing*)
- Strict - ex: $\langle (a, c, 3), (c, d, 4), (d, e, 5) \rangle$ (*increasing*)

Temporal connectivity: $\exists$ temporal paths between all vertices.

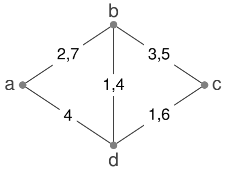
$\rightarrow$ Warning: Reachability is non-symmetrical... and non-transitive!

Temporal spanners

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (say, in number of labels)

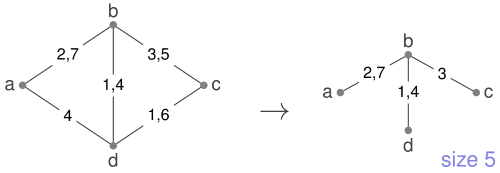


Temporal spanners

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (say, in number of labels)

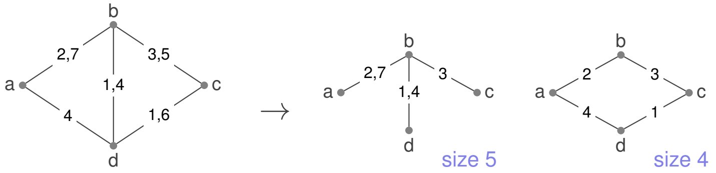


Temporal spanners

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (say, in number of labels)

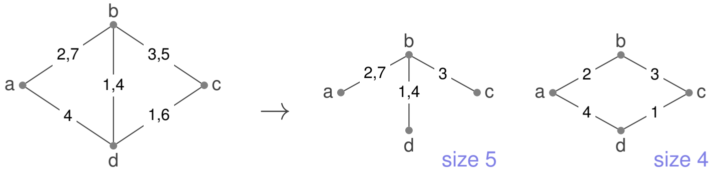


Temporal spanners

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (say, in number of labels)



Can we do better?

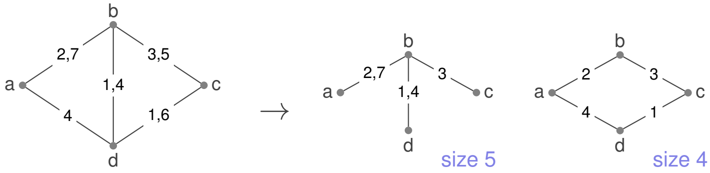
- $2n-4$ labels needed, even if you choose the values! (Bumby'79, gossip theory)

Temporal spanners

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (say, in number of labels)



Can we do better?

- $2n - 4$ labels needed, even if you choose the values! (Bumby'79, gossip theory)

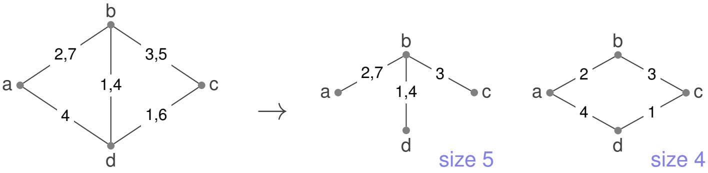
Do spanners of size $2n - 4$ always exist?

Temporal spanners

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (say, in number of labels)



Can we do better?

- $2n - 4$ labels needed, even if you choose the values! (Bumby'79, gossip theory)

Do spanners of size $2n - 4$ always exist?

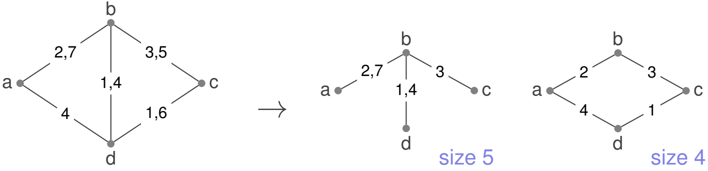
- $\exists$ minimally connected temp. graphs with $\Omega(n \log n)$ labels (Kleinberg, Kempe, Kumar, 2000)

Temporal spanners

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (say, in number of labels)



Can we do better?

- $2n - 4$ labels needed, even if you choose the values! (Bumby'79, gossip theory)

Do spanners of size $2n - 4$ always exist?

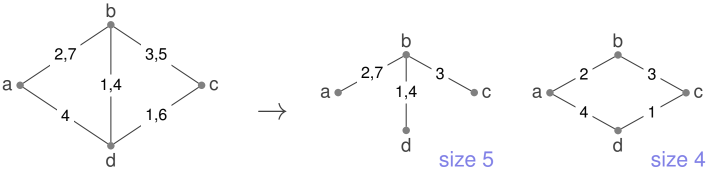
- $\exists$ minimally connected temp. graphs with $\Omega(n \log n)$ labels (Kleinberg, Kempe, Kumar, 2000)
- In fact, $\exists$ some with $\Omega(n^2)$ labels (Axiotis, Fotakis, 2016)

Temporal spanners

Input: a graph $\mathcal{G}$ that is temporally connected ($\mathcal{G} \in TC$)

Output: a graph $\mathcal{G}' \subseteq \mathcal{G}$ that preserves temporal connectivity ($\mathcal{G}' \in TC$)

Cost measure: size of the spanner (say, in number of labels)



Can we do better?

- $2n - 4$ labels needed, even if you choose the values! (Bumby'79, gossip theory)

Do spanners of size $2n - 4$ always exist?

- $\exists$ minimally connected temp. graphs with $\Omega(n \log n)$ labels (Kleinberg, Kempe, Kumar, 2000)
- In fact, $\exists$ some with $\Omega(n^2)$ labels (Axiotis, Fotakis, 2016)

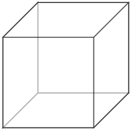
How about complexity?

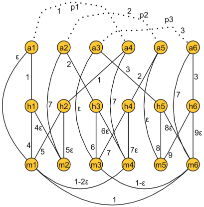
- Minimum-size spanner is APX-hard (Akrida, Gasieniec, Mertzios, Spirakis, 2017)

Bad news and good news

Recall the bad news:

- $\Omega(n \log n)$ - easy
- $\Omega(n^2)$ - rather unexpected

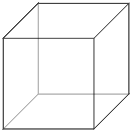


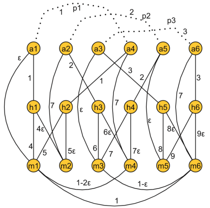


Bad news and good news

Recall the bad news:

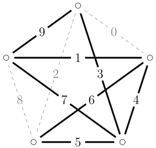
- $\Omega(n \log n)$ - easy
- $\Omega(n^2)$ - rather unexpected





Good news 1: (C., Peters, Schoeters, ICALP 2019):

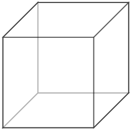
- Spanners of size $O(n \log n)$ always exist in **complete** temporal graphs

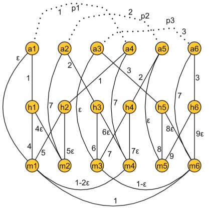


Bad news and good news

Recall the bad news:

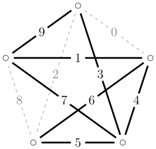
- $\Omega(n \log n)$ - easy
- $\Omega(n^2)$ - rather unexpected





Good news 1: (C., Peters, Schoeters, ICALP 2019):

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



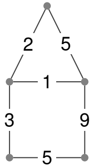
Good news 2: (C., Raskin, Renken, Zamaraev, FOCS 2021):

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

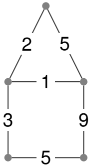
Before we start... an easier model

Simple Temporal Graphs (STGs):

1. A single presence time per edge ($\lambda : E \to \mathbb{N}$)
2. Adjacent edges have different times (λ is locally injective)



Before we start... an easier model



Simple Temporal Graphs (STGs):

1. A single presence time per edge ($\lambda : E \to \mathbb{N}$)
2. Adjacent edges have different times (λ is locally injective)

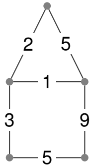
Generality for spanners:

- Most negative results still apply
- Positive results extend to general case
- No distinction between strict and non-strict temporal paths

Before we start... an easier model

Simple Temporal Graphs (STGs):

1. A single presence time per edge ($\lambda : E \to \mathbb{N}$)
2. Adjacent edges have different times (λ is locally injective)



Generality for spanners:

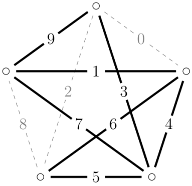
- Most negative results still apply
- Positive results extend to general case
- No distinction between strict and non-strict temporal paths

Further motivations:

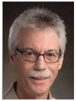
- Distributed models by pairwise interactions, e.g. population protocols or gossip models (without repetition)
- Close model to edge-ordered graphs (Chvátal, Komlós, 1971)

Good news 1:

Temporal cliques admit sparse spanners

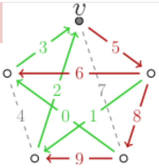


(with)



Two promising techniques...

Pivotability



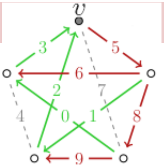
Pivot node v and time t such that:

- all nodes can reach v before t
- v can reach all nodes after t

Then in-tree $\cup$ out-tree $=$ spanner of size $2n-2$ (in fact $2n-3$...)

Two promising techniques...

Pivotability

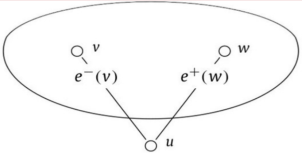


Pivot node v and time t such that:

- all nodes can reach v before t
- v can reach all nodes after t

Then in-tree $\cup$ out-tree $=$ spanner of size $2n-2$ (in fact $2n-3$...)

Dismountability

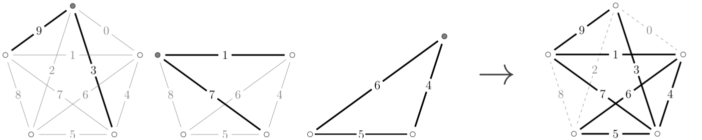


Three nodes u, v, w such that:

- uv =`min-edge`(v)
- uw =`max-edge`(w)

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

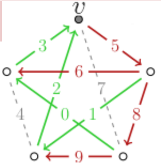
Recursively,



spanner of size $2n-3$.

Two promising techniques...

Pivotability

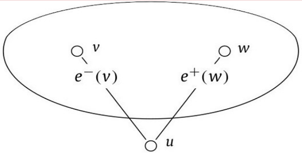


Pivot node v and time t such that:

- all nodes can reach v before t
- v can reach all nodes after t

Then in-tree $\cup$ out-tree $=$ spanner of size $2n-2$ (in fact $2n-3$...)

Dismountability

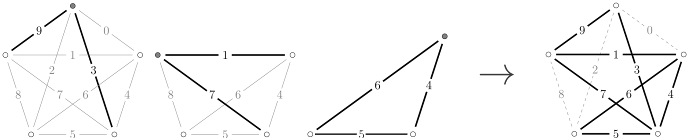


Three nodes u, v, w such that:

- uv =`min-edge`(v)
- uw =`max-edge`(w)

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

Recursively,



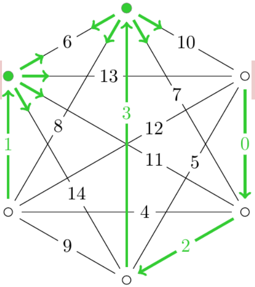
spanner of size $2n-3$.

... unfortunately

Both techniques fail in some cases.

Transitive delegations ("fireworks")

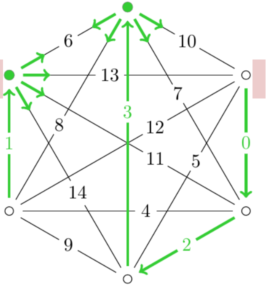
Transitive delegations (“fireworks”)



Principle:

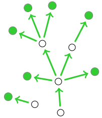
- Transitive delegations towards emitters
- Spanner = forest of min edges + edges of emitters

Transitive delegations (“fireworks”)



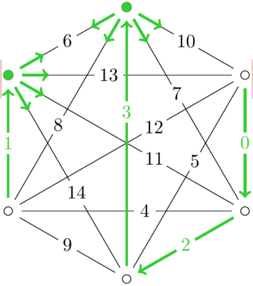
Principle:

- Transitive delegations towards emitters
- Spanner = forest of min edges + edges of emitters



Wait a minute... could be lot of emitters!

Transitive delegations ("fireworks")

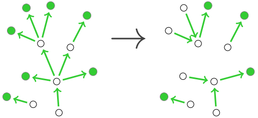


Principle:

- Transitive delegations towards emitters
- Spanner = forest of min edges + edges of emitters

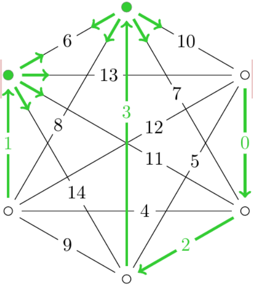
Wait a minute... could be lot of emitters!

$\rightarrow$ *Local transformation of the forest*:



- At most $n/2$ emitters

Transitive delegations ("fireworks")

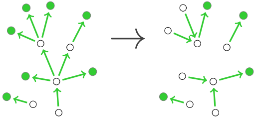


Principle:

- Transitive delegations towards emitters
- Spanner = forest of min edges + edges of emitters

Wait a minute... could be lot of emitters!

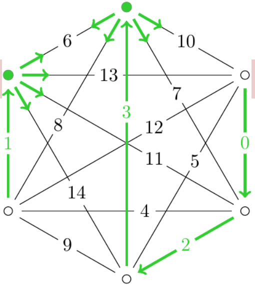
$\rightarrow$ *Local transformation of the forest*:



- At most $n/2$ emitters

Theorem: $\exists$ spanners of size $\frac{3}{4}\binom{n}{2} + O(n)$

Transitive delegations (“fireworks”)

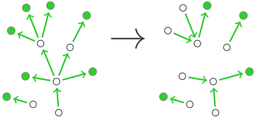


Principle:

- Transitive delegations towards emitters
- Spanner = forest of min edges + edges of emitters

Wait a minute... *could be lot of emitters!*

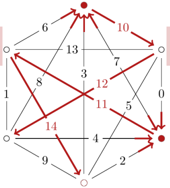
$\rightarrow$ *Local transformation of the forest*:



- At most $n/2$ emitters

Theorem: $\exists$ spanners of size $\frac{3}{4}\binom{n}{2} + O(n)$

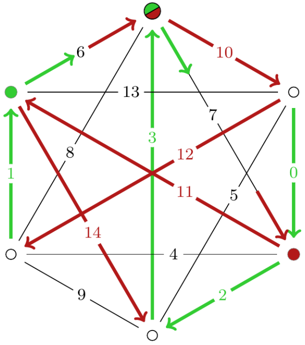
Backward spanner also possible



$\rightarrow$ Spanner = max edges + all edges of collectors

Combining both directions

- Each vertex can reach at least one emitter u through u's min edge
- Each vertex can be reached by a collector v through v's max edge
- Each emitter can reach all collectors through direct edges



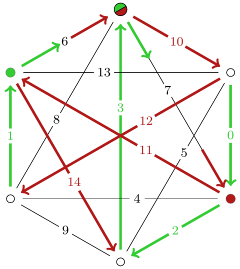
$\rightarrow$ Spanner = min edges + max edges
+ edges between emitters and collectors

Theorem:

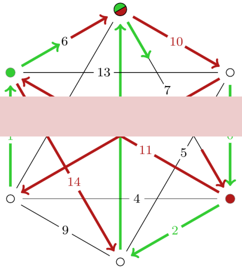
At most $n/2$ emitters and $n/2$ collectors $\Rightarrow$ $\exists$ Spanners of size $\binom{n}{2}/2 + O(n)$

$\approx$ half of the edges

Recurse or sparsify?



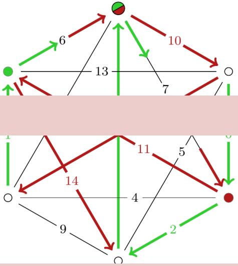
Recurse or sparsify?



Case 1: emitters $\cup$ collectors $\neq V$

Lemma: There exists a "2-*hop dismountable*" vertex v
$\rightarrow$ select 4 edges, recurse on $\mathcal{G}[V - v]$

Recurse or sparsify?



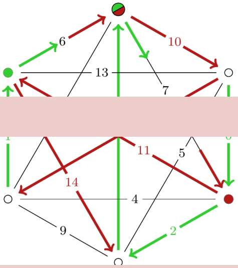
Case 1: emitters $\cup$ collectors $\neq V$

Lemma: There exists a "2-*hop dismountable*" vertex v
$\rightarrow$ select 4 edges, recurse on $\mathcal{G}[V - v]$

Case 2: emitters $\cup$ collectors $= V$

$\rightarrow$ All vertices are either emitters or collectors (not both)!

Recurse or sparsify?

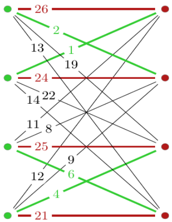


Case 1: emitters $\cup$ collectors $\neq V$

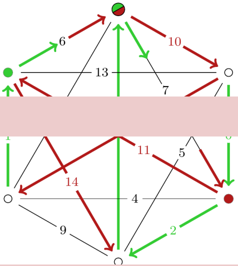
Lemma: There exists a "2-*hop dismountable*" vertex v
$\rightarrow$ select 4 edges, recurse on $\mathcal{G}[V - v]$

Case 2: emitters $\cup$ collectors $= V$

$\rightarrow$ All vertices are <u>either</u> emitters or collectors (not both)!



Recurse or sparsify?

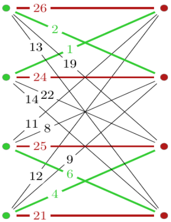


Case 1: emitters $\cup$ collectors $\neq V$

Lemma: There exists a "2-*hop dismountable*" vertex v
$\rightarrow$ select 4 edges, recurse on $\mathcal{G}[V - v]$

Case 2: emitters $\cup$ collectors $= V$

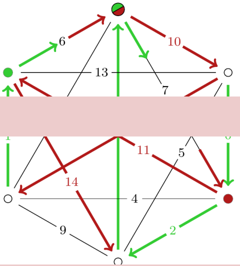
$\rightarrow$ All vertices are <u>either</u> emitters or collectors (not both)!



A lot of structure to work with:

- Complete bipartite graph $\mathcal{H}$ between emitters and collectors
- Min edges and max edges form two perfect matchings
- W.l.o.g. min edges (max edges) are *reciprocal* in $\mathcal{H}$

Recurse or sparsify?

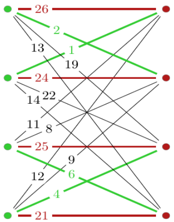


Case 1: emitters $\cup$ collectors $\neq V$

Lemma: There exists a "2-*hop dismountable*" vertex v
$\rightarrow$ select 4 edges, recurse on $\mathcal{G}[V - v]$

Case 2: emitters $\cup$ collectors $= V$

$\rightarrow$ All vertices are either emitters or collectors (not both)!



A lot of structure to work with:

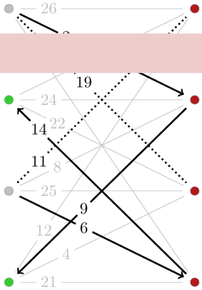
- Complete bipartite graph $\mathcal{H}$ between emitters and collectors
- Min edges and max edges form two perfect matchings
- W.l.o.g. min edges (max edges) are *reciprocal* in $\mathcal{H}$

New objective:
$\rightarrow$ Sparsify $\mathcal{H}$ while preserving journeys from each emitter to all collectors

Sparsification of the bipartite graph

New objective:

$\rightarrow$ Sparsify $\mathcal{H}$ while preserving journeys from each emitter to all collectors



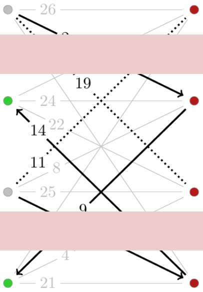
Technique: Partial delegations among emitters

- Find a 2-hop journey from one emitter to another, arriving through a "locally small" edge
- Pay extra edges (penalty) to reach missed collectors

Sparsification of the bipartite graph

New objective:

$\rightarrow$ Sparsify $\mathcal{H}$ while preserving journeys from each emitter to all collectors



Technique: Partial delegations among emitters

- Find a 2-hop journey from one emitter to another, arriving through a "locally small" edge
- Pay extra edges (penalty) to reach missed collectors

Iterative procedure:

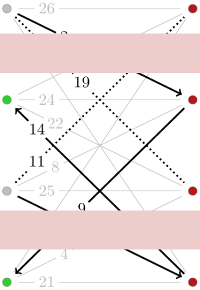
In each step i:

- Half of the emitters partially delegate to other half
- Penalty doubles in each step, but #emitters halves
- $O(n)$ edges over $O(\log n)$ iterations $\rightarrow$ $\boldsymbol{O(n \log n)}$ edges.

Sparsification of the bipartite graph

New objective:

$\rightarrow$ Sparsify $\mathcal{H}$ while preserving journeys from each emitter to all collectors



Technique: Partial delegations among emitters

- Find a 2-hop journey from one emitter to another, arriving through a “locally small” edge
- Pay extra edges (penalty) to reach missed collectors

Iterative procedure:

In each step i:

- Half of the emitters partially delegate to other half
- Penalty doubles in each step, but #emitters halves
- $O(n)$ edges over $O(\log n)$ iterations $\rightarrow$ $\boldsymbol{O(n \log n)}$ edges.

Conclusion:

$\exists$ spanner of size $O(n \log n)$ $\square$

Open questions (deterministic)

Better spanners for temporal cliques?

- Is $O(n \log n)$ optimal for cliques? Is $O(n)$ possible?
- Even better, does $2n - 4 \leq OPT \leq 2n - 3$?
  (so far, no counter-example found)

Open questions (deterministic)

Better spanners for temporal cliques?

- Is $O(n \log n)$ optimal for cliques? Is $O(n)$ possible?
- Even better, does $2n - 4 \leq OPT \leq 2n - 3$?
  (so far, no counter-example found)

Relaxing the complete graph assumption

- Can more general classes of dense graphs be sparsified?

  $\rightarrow$ Recall that $\exists$ unsparsifiable graphs of density $\Theta(n^2)$

  $\rightarrow$ Is there a family of graphs of density < 1 which admits sparse spanners?

Open questions (deterministic)

Better spanners for temporal cliques?

- Is $O(n \log n)$ optimal for cliques? Is $O(n)$ possible?
- Even better, does $2n-4 \leq OPT \leq 2n-3$?
  (so far, no counter-example found)

Relaxing the complete graph assumption

- Can more general classes of dense graphs be sparsified?

  $\rightarrow$ Recall that $\exists$ unsparsifiable graphs of density $\Theta(n^2)$

  $\rightarrow$ Is there a family of graphs of density < 1 which admits sparse spanners?

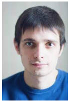
What about random temporal graphs?

Good news 2:

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

in random temporal graphs

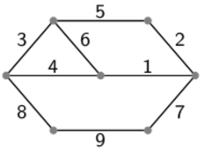
(with)

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

Random simple temporal graphs:

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

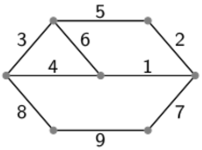


Timeline for p (as $n \to \infty$):

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

Random simple 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 $n \to \infty$):

0 $\frac{\log n}{n}$ 1

Standard connectivity

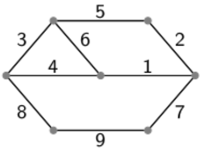
Most vertex pairs reach each other ($\sim * \rightsquigarrow \sim *$)

 17/1

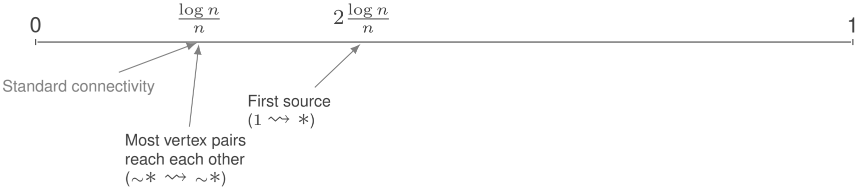
Sharp thresholds in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random simple temporal graphs:

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



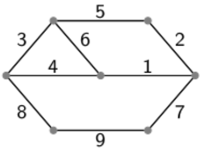
Timeline for p (as $n \to \infty$):



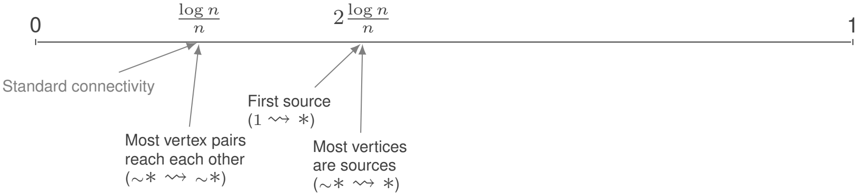
Sharp thresholds in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random simple temporal graphs:

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



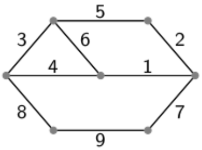
Timeline for p (as $n \to \infty$):



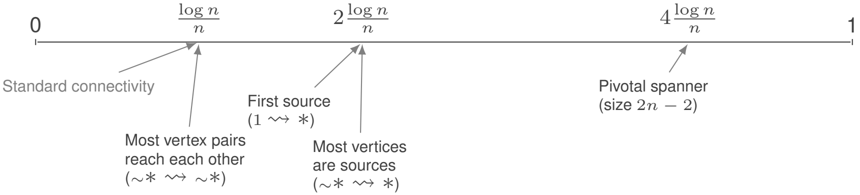
Sharp thresholds in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random simple temporal graphs:

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



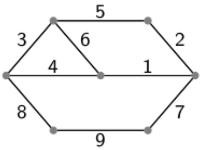
Timeline for p (as $n \to \infty$):



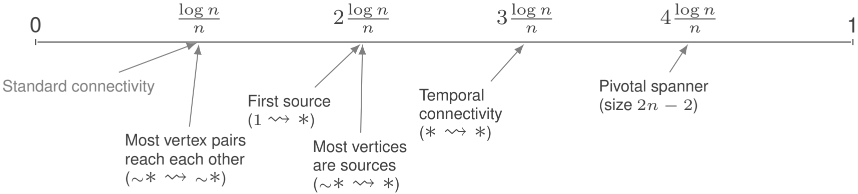
Sharp thresholds in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random simple temporal graphs:

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



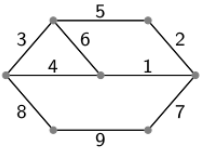
Timeline for p (as $n \to \infty$):



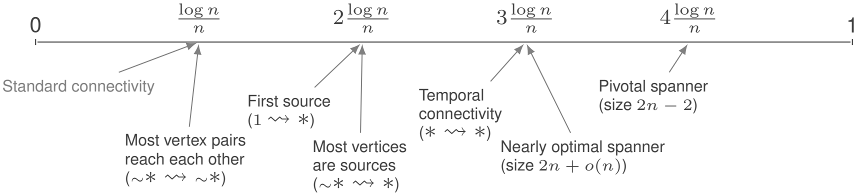
Sharp thresholds in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random simple temporal graphs:

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



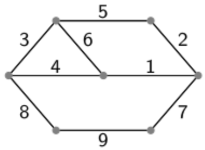
Timeline for p (as $n \to \infty$):



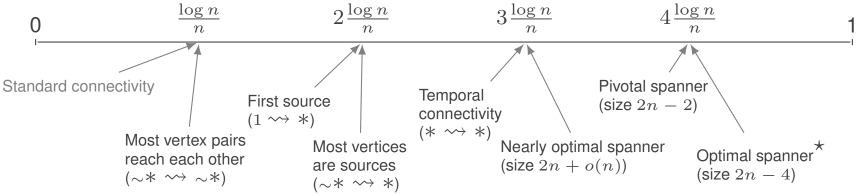
Sharp thresholds in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random simple temporal graphs:

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



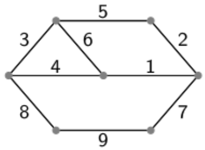
Timeline for p (as $n \to \infty$):



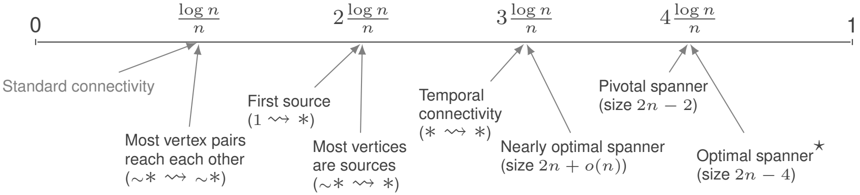
Sharp thresholds in random temporal graphs (C., Raskin, Renken, Zamaraev, 2021)

Random simple 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, except $\star$ (open problem)

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

Random Simple Temporal Graphs (RSTGs)

Temporal analog of Erdös-Reyni graphs, same parameters n and p

Random Simple Temporal Graphs (RSTGs)

Temporal analog of Erdös-Reyni graphs, same parameters n and p

An RSTG $\mathcal{G} \sim \mathcal{G}_{n,p}$:

1. Pick a footprint $G \sim G_{n,p}$
2. Permute the edges randomly
   (interpret ranks as times)

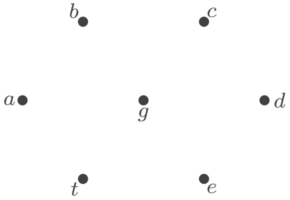
Random Simple Temporal Graphs (RSTGs)

Temporal analog of Erdös-Reyni graphs, same parameters n and p

An RSTG $\mathcal{G} \sim \mathcal{G}_{n,p}$:

1. Pick a footprint $G \sim G_{n,p}$
2. Permute the edges randomly (interpret ranks as times)

Ex: $n = 7, p = 0.4$



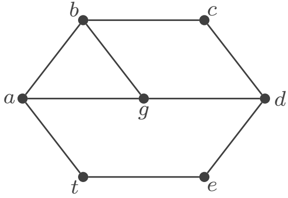
Random Simple Temporal Graphs (RSTGs)

Temporal analog of Erdös-Reyni graphs, same parameters n and p

An RSTG $\mathcal{G} \sim \mathcal{G}_{n,p}$:

1. Pick a footprint $G \sim G_{n,p}$
2. Permute the edges randomly (interpret ranks as times)

Ex: $n = 7, p = 0.4$



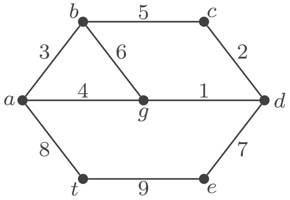
Random Simple Temporal Graphs (RSTGs)

Temporal analog of Erdös-Reyni graphs, same parameters n and p

An RSTG $\mathcal{G} \sim \mathcal{G}_{n,p}$:

1. Pick a footprint $G \sim G_{n,p}$
2. Permute the edges randomly (interpret ranks as times)

Ex: $n = 7, p = 0.4$



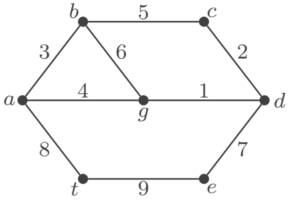
Random Simple Temporal Graphs (RSTGs)

Temporal analog of Erdös-Reyni graphs, same parameters n and p

An RSTG $\mathcal{G} \sim \mathcal{G}_{n,p}$:

1. Pick a footprint $G \sim G_{n,p}$
2. Permute the edges randomly (interpret ranks as times)

Ex: $n = 7, p = 0.4$



Another point of view:

1. Take a complete graph K_n
2. Assign random real times in $[0, 1]$ to every edge
3. Restrict your attention to $\mathcal{G}_{[0,p]}$

→ Better for analysis.

Timeline of temporal reachability in $\mathcal{G}_{n,p}$

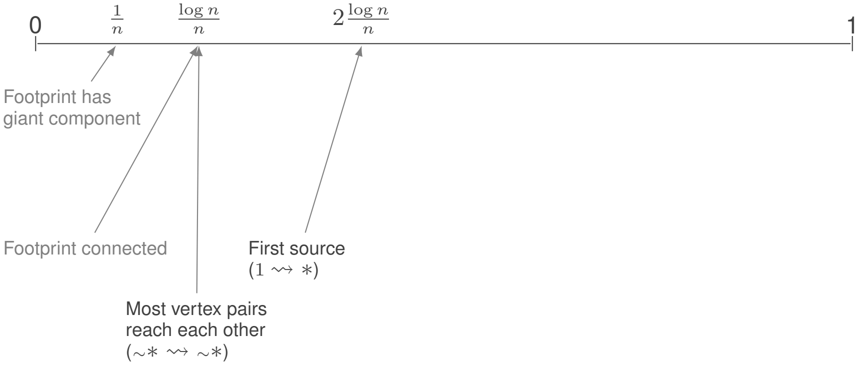
For sufficiently large n, what happens when p increases?

Timeline of temporal reachability in $\mathcal{G}_{n,p}$

For sufficiently large n, what happens when p increases?

Timeline of temporal reachability in $\mathcal{G}_{n,p}$

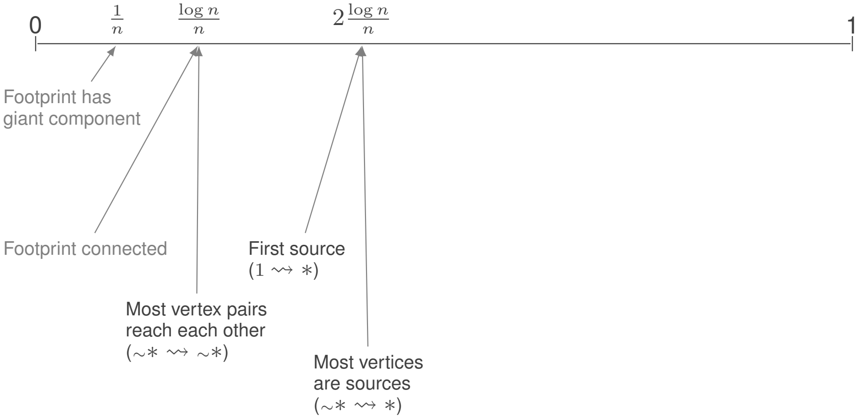
For sufficiently large n, what happens when p increases?



 19/1

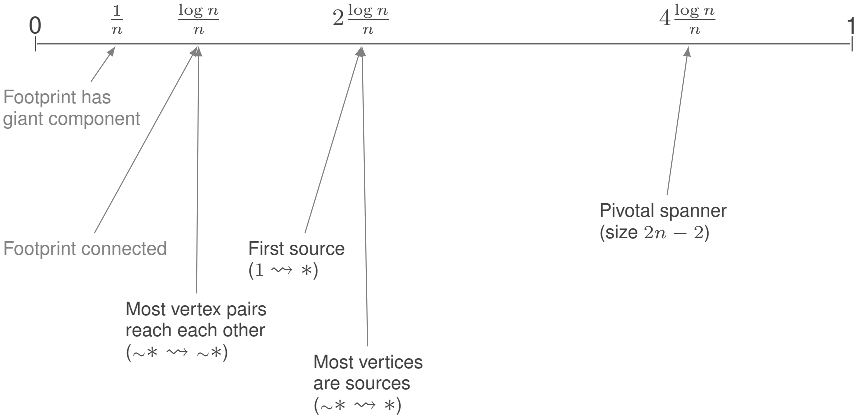
Timeline of temporal reachability in $\mathcal{G}_{n,p}$

For sufficiently large n, what happens when p increases?



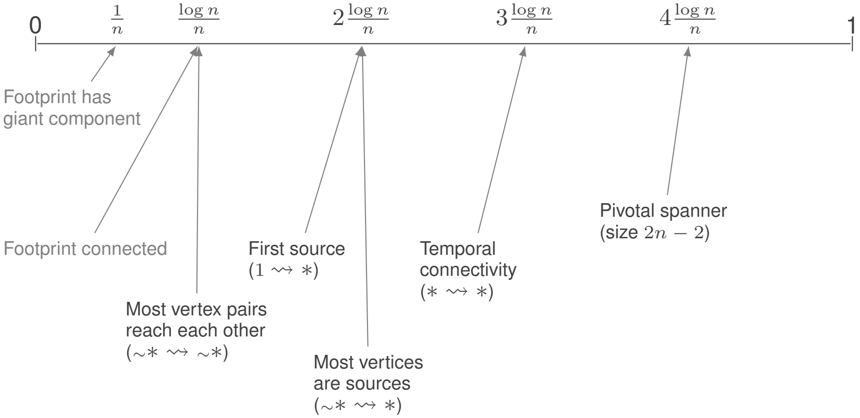
Timeline of temporal reachability in $\mathcal{G}_{n,p}$

For sufficiently large n, what happens when p increases?



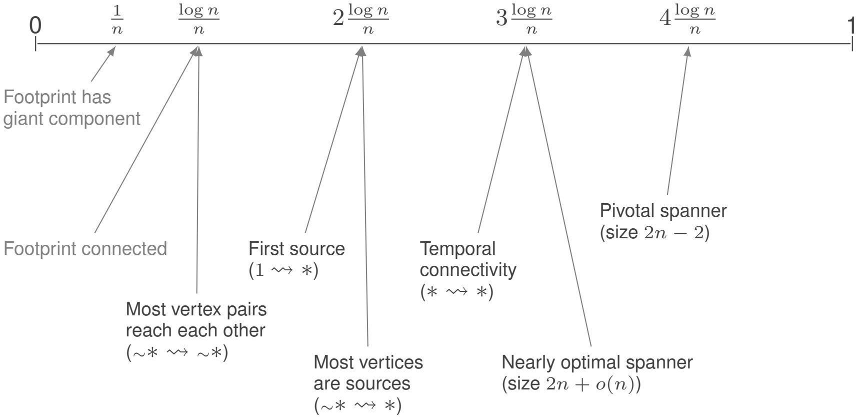
Timeline of temporal reachability in $\mathcal{G}_{n,p}$

For sufficiently large n, what happens when p increases?



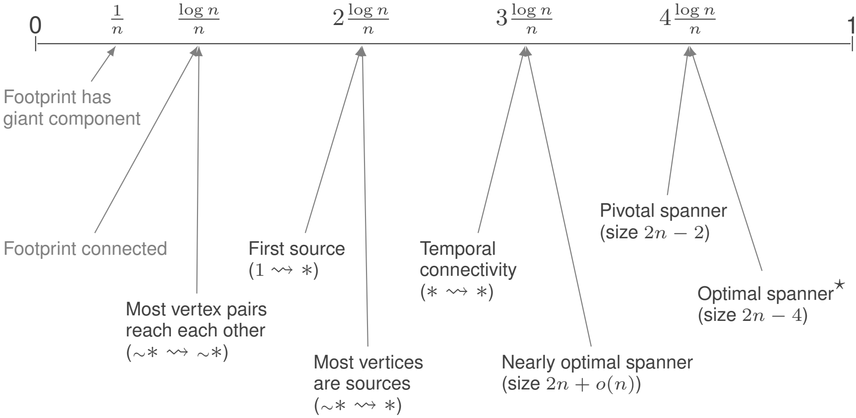
Timeline of temporal reachability in $\mathcal{G}_{n,p}$

For sufficiently large n, what happens when p increases?



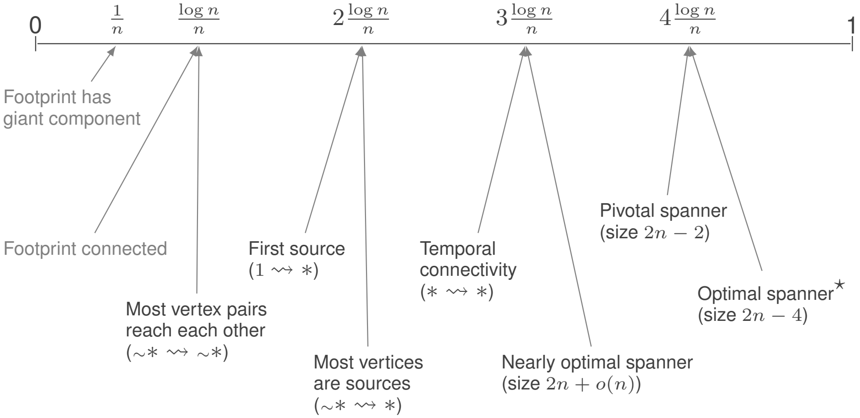
Timeline of temporal reachability in $\mathcal{G}_{n,p}$

For sufficiently large n, what happens when p increases?



Timeline of temporal reachability in $\mathcal{G}_{n,p}$

For sufficiently large n, what happens when p increases?



All the thresholds are sharp, except $\star$ (open problem)

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

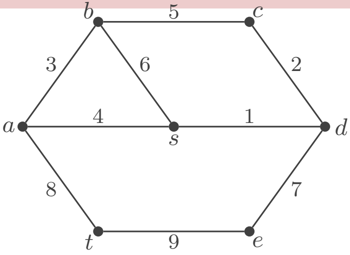
 19/1

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



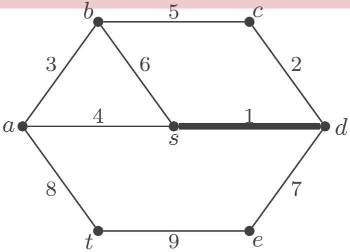
$t = 0$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- “Prim-like” algorithm.



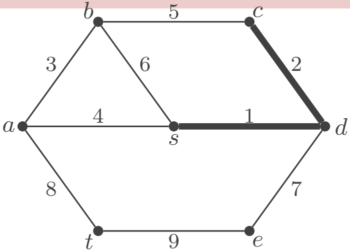
$t = 1$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



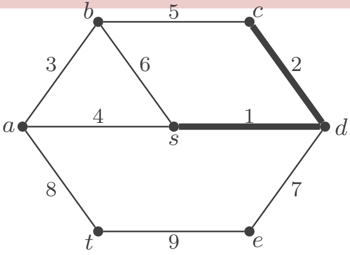
$t = 2$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



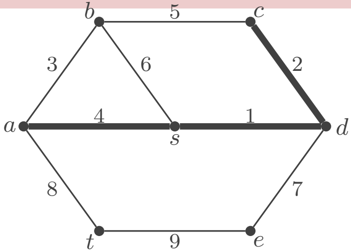
$t = 3$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.

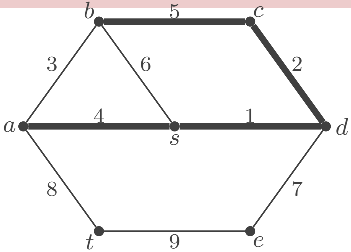


$t = 4$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all
- "Prim-like" algorithm.

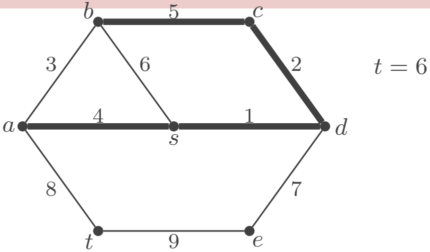


Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- “Prim-like” algorithm.

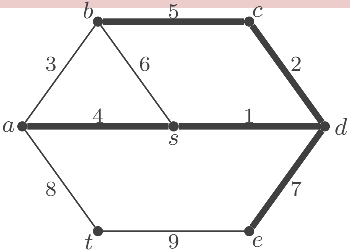


Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- “Prim-like” algorithm.

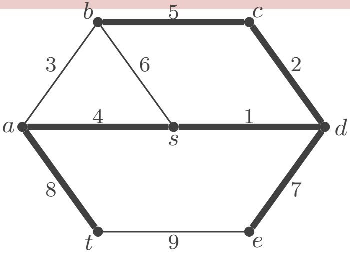


$t = 7$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all
- “Prim-like” algorithm.



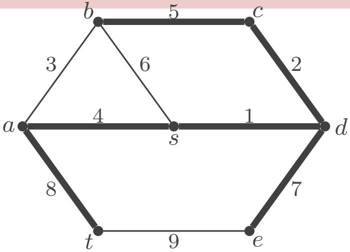
$t = 8$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- “Prim-like” algorithm.



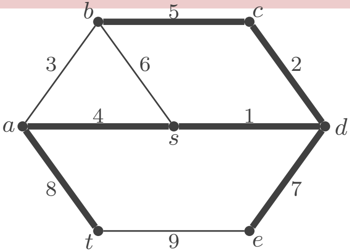
Analysis

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



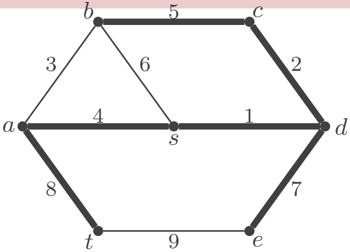
Analysis

- Consider "growing" a foremost tree in $\mathcal{G} \sim \mathcal{G}_{n,1}$ from a vertex s.

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all
- "Prim-like" algorithm.



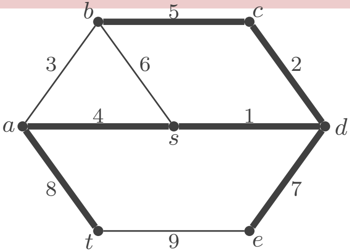
Analysis

- Consider "growing" a foremost tree in $\mathcal{G} \sim \mathcal{G}_{n,1}$ from a vertex s.
- Once we have reached k vertices, there are $k(n-k)$ potential edges.

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all
- “Prim-like” algorithm.



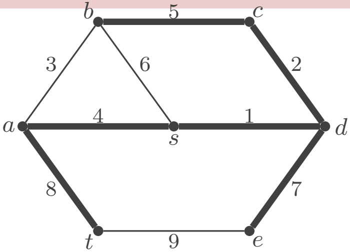
Analysis

- Consider “growing” a foremost tree in $\mathcal{G} \sim \mathcal{G}_{n,1}$ from a vertex s.
- Once we have reached k vertices, there are $k(n-k)$ potential edges.
- The waiting time for one of these to appear is $\approx \frac{1}{k(n-k)}$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all
- "Prim-like" algorithm.



Analysis

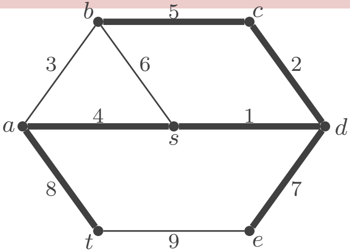
- Consider "growing" a foremost tree in $\mathcal{G} \sim \mathcal{G}_{n,1}$ from a vertex s.
- Once we have reached k vertices, there are $k(n-k)$ potential edges.
- The waiting time for one of these to appear is $\approx \frac{1}{k(n-k)}$

$\Longrightarrow$ Expect to reach all vertices at $\sum_{k=1}^{n} \frac{1}{k(n-k)} \approx 2\frac{\log n}{n}$.

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all
- "Prim-like" algorithm.



Analysis

- Consider "growing" a foremost tree in $\mathcal{G} \sim \mathcal{G}_{n,1}$ from a vertex s.
- Once we have reached k vertices, there are $k(n-k)$ potential edges.
- The waiting time for one of these to appear is $\approx \frac{1}{k(n-k)}$

$\Longrightarrow$ Expect to reach all vertices at $\sum_{k=1}^{n} \frac{1}{k(n-k)} \approx 2\frac{\log n}{n}$.

- Azuma's inequality for concentration.

Derived arguments

We note $foremost(u)$ the set of vertices reached by a foremost tree from u.

Reachability thresholds

- $\sim* \rightsquigarrow \sim* \iff \forall u, \forall v, a.a.s.\ v \in foremost(u)$ $(\log n/n)$

Derived arguments

We note $foremost(u)$ the set of vertices reached by a foremost tree from u.

Reachability thresholds

- $\sim\!* \rightsquigarrow \sim\!* \iff \forall u, \forall v, a.a.s.\ v \in foremost(u)$ $(\log n/n)$
- $1 \rightsquigarrow * \iff a.a.s.\ \exists u, \forall v, v \in foremost(u)$ $(2\log n/n)$

Derived arguments

We note $foremost(u)$ the set of vertices reached by a foremost tree from u.

Reachability thresholds

- $\sim * \rightsquigarrow \sim * \iff \forall u, \forall v, a.a.s.\ v \in foremost(u)$ $(\log n/n)$
- $1 \rightsquigarrow * \iff a.a.s.\ \exists u, \forall v, v \in foremost(u)$ $(2\log n/n)$
- $\sim * \rightsquigarrow * \iff \forall u, a.a.s.\ \forall v, v \in foremost(u)$ $(2\log n/n)$

Derived arguments

We note $foremost(u)$ the set of vertices reached by a foremost tree from u.

Reachability thresholds

- $\sim\!* \rightsquigarrow\!\sim\!* \iff \forall u, \forall v, a.a.s.\ v \in foremost(u)$ $(\log n/n)$
- $1 \rightsquigarrow * \iff a.a.s.\ \exists u, \forall v, v \in foremost(u)$ $(2 \log n/n)$
- $\sim\!* \rightsquigarrow * \iff \forall u, a.a.s.\ \forall v, v \in foremost(u)$ $(2 \log n/n)$
- $* \rightsquigarrow * \iff a.a.s\ \forall u, \forall v, v \in foremost(u)$ $(3 \log n/n)$

  LB: $(* \rightsquigarrow 1) + (\log n/n)$, each non sink must have at least one new edge.

  UB: $(* \rightsquigarrow\!\sim\!*) + (\log n/n)$, each non sink is reached from at least one sink.

Derived arguments

We note $foremost(u)$ the set of vertices reached by a foremost tree from u.

Reachability thresholds

- $\sim\!* \rightsquigarrow \sim\!* \iff \forall u, \forall v, a.a.s.\ v \in foremost(u)$ $(\log n/n)$
- $1 \rightsquigarrow * \iff a.a.s.\ \exists u, \forall v, v \in foremost(u)$ $(2\log n/n)$
- $\sim\!* \rightsquigarrow * \iff \forall u, a.a.s.\ \forall v, v \in foremost(u)$ $(2\log n/n)$
- $* \rightsquigarrow * \iff a.a.s\ \forall u, \forall v, v \in foremost(u)$ $(3\log n/n)$
  LB: $(* \rightsquigarrow 1) + (\log n/n)$, each non sink must have at least one new edge.
  UB: $(* \rightsquigarrow \sim\!*) + (\log n/n)$, each non sink is reached from at least one sink.

Spanners

Derived arguments

We note $foremost(u)$ the set of vertices reached by a foremost tree from u.

Reachability thresholds

- $\sim\!* \rightsquigarrow \sim\!* \iff \forall u, \forall v, a.a.s.\ v \in foremost(u)$ $(\log n/n)$
- $1 \rightsquigarrow * \iff a.a.s.\ \exists u, \forall v, v \in foremost(u)$ $(2\log n/n)$
- $\sim\!* \rightsquigarrow * \iff \forall u, a.a.s.\ \forall v, v \in foremost(u)$ $(2\log n/n)$
- $* \rightsquigarrow * \iff a.a.s\ \forall u, \forall v, v \in foremost(u)$ $(3\log n/n)$
  LB: $(* \rightsquigarrow 1) + (\log n/n)$, each non sink must have at least one new edge.
  UB: $(* \rightsquigarrow \sim\!*) + (\log n/n)$, each non sink is reached from at least one sink.

Spanners

- Pivotal $(* \rightsquigarrow 1 \rightsquigarrow *) \Longleftarrow (* \rightsquigarrow \sim\!*) + (\sim\!* \rightsquigarrow *)$ $(4\log n/n)$

Derived arguments

We note $foremost(u)$ the set of vertices reached by a foremost tree from u.

Reachability thresholds

- $\sim\!* \rightsquigarrow \sim\!* \iff \forall u, \forall v, a.a.s.\ v \in foremost(u)$ $(\log n/n)$
- $1 \rightsquigarrow * \iff a.a.s.\ \exists u, \forall v, v \in foremost(u)$ $(2\log n/n)$
- $\sim\!* \rightsquigarrow * \iff \forall u, a.a.s.\ \forall v, v \in foremost(u)$ $(2\log n/n)$
- $* \rightsquigarrow * \iff a.a.s\ \forall u, \forall v, v \in foremost(u)$ $(3\log n/n)$
  LB: $(* \rightsquigarrow 1) + (\log n/n)$, each non sink must have at least one new edge.
  UB: $(* \rightsquigarrow \sim\!*) + (\log n/n)$, each non sink is reached from at least one sink.

Spanners

- Pivotal $(* \rightsquigarrow 1 \rightsquigarrow *) \Longleftarrow (* \rightsquigarrow \sim\!*) + (\sim\!* \rightsquigarrow *)$ $(4\log n/n)$
- Optimal spanner (size $2n-4$) $(4\log n/n)$
  Pivotal square. Sharp ?

Derived arguments

We note $foremost(u)$ the set of vertices reached by a foremost tree from u.

Reachability thresholds

- $\sim\!* \rightsquigarrow \sim\!* \iff \forall u, \forall v, a.a.s.\ v \in foremost(u)$ $(\log n/n)$
- $1 \rightsquigarrow * \iff a.a.s.\ \exists u, \forall v, v \in foremost(u)$ $(2\log n/n)$
- $\sim\!* \rightsquigarrow * \iff \forall u, a.a.s.\ \forall v, v \in foremost(u)$ $(2\log n/n)$
- $* \rightsquigarrow * \iff a.a.s\ \forall u, \forall v, v \in foremost(u)$ $(3\log n/n)$

  LB: $(* \rightsquigarrow 1) + (\log n/n)$, each non sink must have at least one new edge.

  UB: $(* \rightsquigarrow \sim\!*) + (\log n/n)$, each non sink is reached from at least one sink.

Spanners

- Pivotal $(* \rightsquigarrow 1 \rightsquigarrow *) \Longleftarrow (* \rightsquigarrow \sim\!*) + (\sim\!* \rightsquigarrow *)$ $(4\log n/n)$
- Optimal spanner (size $2n-4$) $(4\log n/n)$

  Pivotal square. Sharp ?
- Nearly optimal spanner (size $2n + o(n)$) $(3\log n/n)$

  LB: Trivial (not temporally connected)

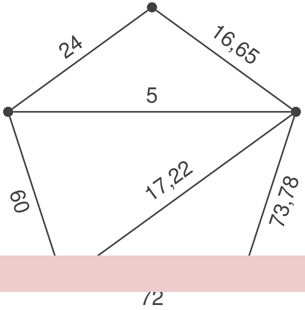
  UB: Explicit construction

  Three intervals of length $\log n/n$:

- $\sim\!* \rightsquigarrow 1$ (say u) — By $\log n/n$
- $u \rightsquigarrow \sim\!*$ — between $2\log n/n$ and $3\log n/n$.
- $missing \rightsquigarrow u$ — between 0 and $2\log n$
- $u \rightsquigarrow missing$ — between $\log n/n$ and $3\log n/n$
- $missing \rightsquigarrow missing$ — between 0 and $3\log n/n$

Random Non-Simple Temporal Graphs

$\mathcal{H}_{n,p}$: Each edge independently appears according to a rate 1 Poisson process stopped at time p.



Theorem

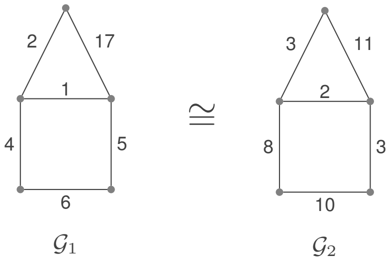
All our thresholds also hold for $\mathcal{H}_{n,p}$.

Simple temporal graphs (beyond spanners)

Special properties and symmetries

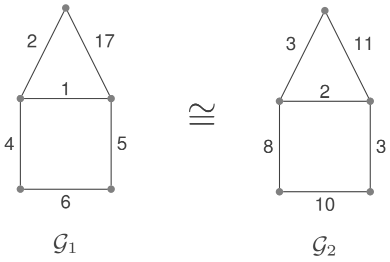
Equivalence based on reachability (up to time distortion)

Different STGs are equivalent in terms of *reachability*



Equivalence based on reachability (up to time distortion)

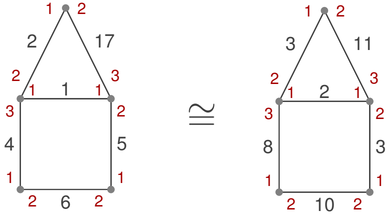
Different STGs are equivalent in terms of *reachability*



How to capture this equivalence?

Equivalence based on reachability (up to time distortion)

Different STGs are equivalent in terms of *reachability*



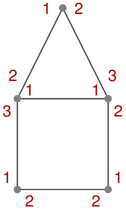
$\mathcal{G}_1$ $\mathcal{G}_2$

How to capture this equivalence?

- Option 1: Local ordering?

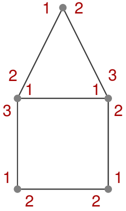
Equivalence based on reachability (up to time distortion)

Different STGs are equivalent in terms of *reachability*



$\mathcal{G}_1$

$\cong$



$\mathcal{G}_2$

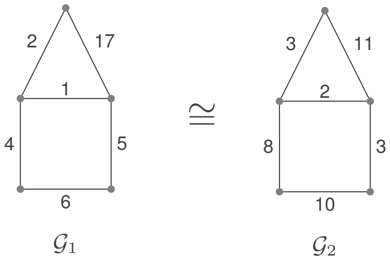
How to capture this equivalence?

- Option 1: Local ordering?

 24/1

Equivalence based on reachability (up to time distortion)

Different STGs are equivalent in terms of *reachability*

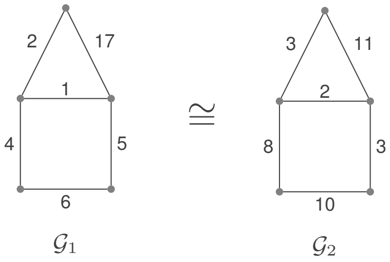


How to capture this equivalence?

- Option 1: Local ordering?

Equivalence based on reachability (up to time distortion)

Different STGs are equivalent in terms of *reachability*

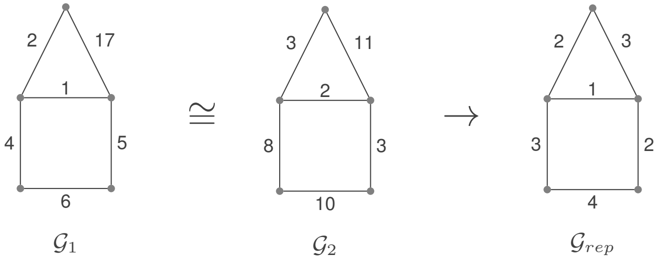


How to capture this equivalence?

- Option 1: Local ordering?
- Option 2: STG representative

Equivalence based on reachability (up to time distortion)

Different STGs are equivalent in terms of *reachability*



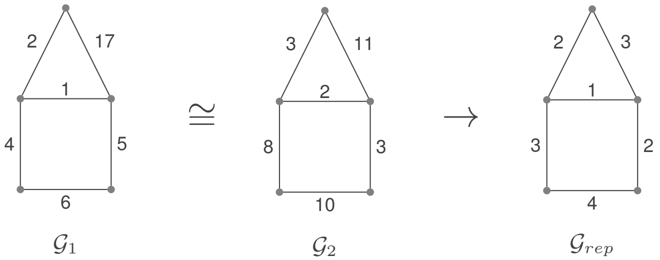
How to capture this equivalence?

- Option 1: Local ordering?
- Option 2: STG representative

 24/1

Equivalence based on reachability (up to time distortion)

Different STGs are equivalent in terms of *reachability*



How to capture this equivalence?

- Option 1: Local ordering?
- Option 2: STG representative ✓

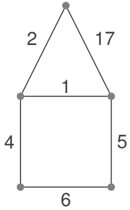
STG representatives have good properties for generation

+ canonization, isomorphism testing, and computation of generators for the automorphism group, are all feasible in *polynomial time*.

STG representatives

Canonization

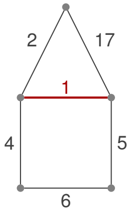
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

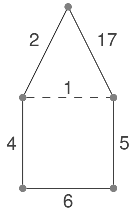
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

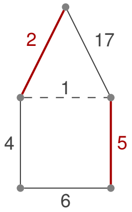
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

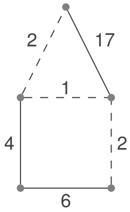
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

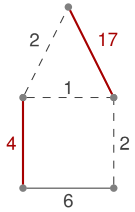
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

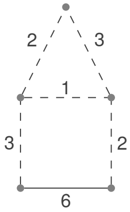
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

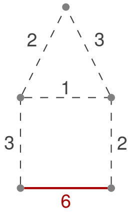
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

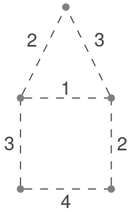
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

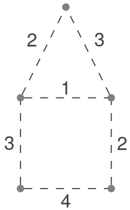
1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



STG representatives

Canonization

1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



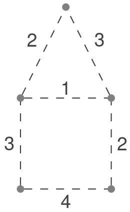
Properties of the labeling

Time induces a *proper* coloring of the edges (by definition of STGs).

STG representatives

Canonization

1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



Properties of the labeling

Time induces a *proper* coloring of the edges (by definition of STGs).

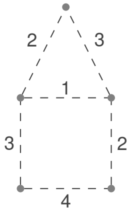
In addition,

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

STG representatives

Canonization

1. Find edges that are *local* minima
2. Assign them the smallest available time
3. Increment time
4. Repeat on remaining edges



Properties of the labeling

Time induces a *proper* coloring of the edges (by definition of STGs).

In addition,

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

(If you know a name for this type of edge coloring, please let me know.)

How to test for equivalence?

How to test for equivalence?

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

Output: Are they equivalent?

Two steps algorithm:

1. Canonize them
2. Test for (classical) isomorphism of the canonical forms

How to test for equivalence?

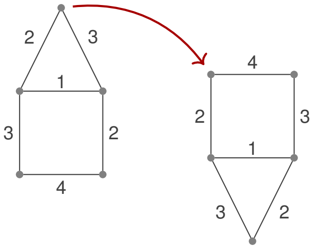
Input: Two STGs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

Two steps algorithm:

1. Canonize them
2. Test for (classical) isomorphism of the canonical forms

Algorithm for the second step:



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)

How to test for equivalence?

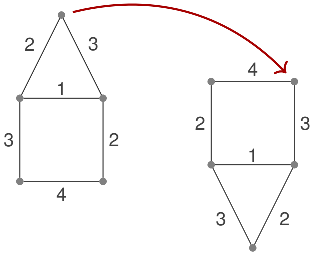
Input: Two STGs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

Two steps algorithm:

1. Canonize them
2. Test for (classical) isomorphism of the canonical forms

Algorithm for the second step:



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)

How to test for equivalence?

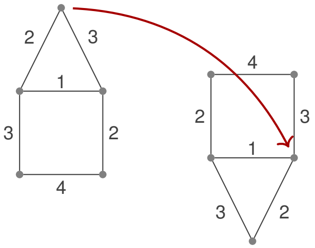
Input: Two STGs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

Two steps algorithm:

1. Canonize them
2. Test for (classical) isomorphism of the canonical forms

Algorithm for the second step:



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)

How to test for equivalence?

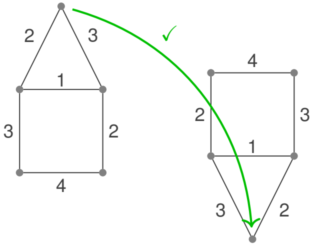
Input: Two STGs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

Two steps algorithm:

1. Canonize them
2. Test for (classical) isomorphism of the canonical forms

Algorithm for the second step:



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)

How to test for equivalence?

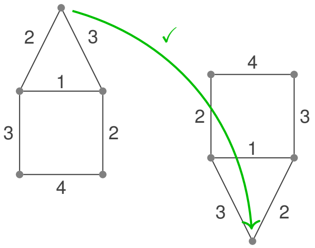
Input: Two STGs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

Two steps algorithm:

1. Canonize them
2. Test for (classical) isomorphism of the canonical forms

Algorithm for the second step:



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 among neighbors unfolds recursively without choices (due to the *proper coloring* of the edges)

$\rightarrow$ passes or fails in polynomial time.

How to test for equivalence?

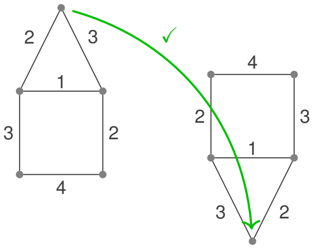
Input: Two STGs $\mathcal{G}_1$ and $\mathcal{G}_2$

Output: Are they equivalent?

Two steps algorithm:

1. Canonize them
2. Test for (classical) isomorphism of the canonical forms

Algorithm for the second step:



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 among neighbors unfolds recursively without choices (due to the *proper coloring* of the edges)

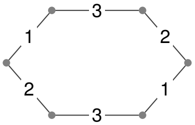
$\rightarrow$ passes or fails in polynomial time.

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

Automorphisms of an STG

Case 1: The underlying graph is connected.

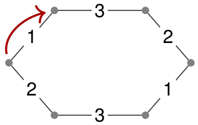
→ Same strategy as for isomorphism.



Automorphisms of an STG

Case 1: The underlying graph is connected.

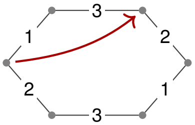
→ Same strategy as for isomorphism.



Automorphisms of an STG

Case 1: The underlying graph is connected.

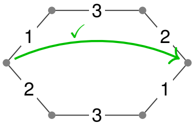
→ Same strategy as for isomorphism.



Automorphisms of an STG

Case 1: The underlying graph is connected.

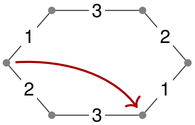
→ Same strategy as for isomorphism.



Automorphisms of an STG

Case 1: The underlying graph is connected.

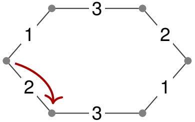
→ Same strategy as for isomorphism.



Automorphisms of an STG

Case 1: The underlying graph is connected.

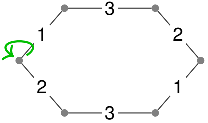
→ Same strategy as for isomorphism.



Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.

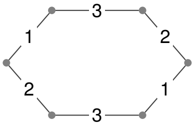


At most n automorphisms!

Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



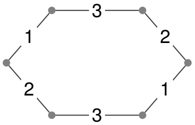
Case 2: The underlying graph is not connected

(the complement trick does not works for temporal graphs...)

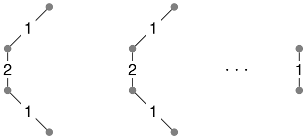
Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)

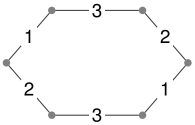


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

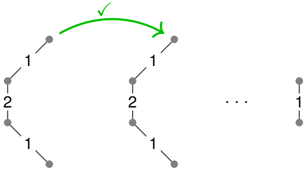
Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)

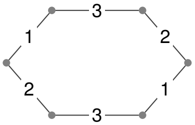


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

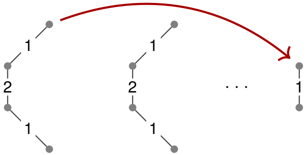
Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)

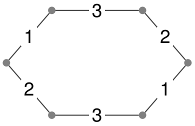


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

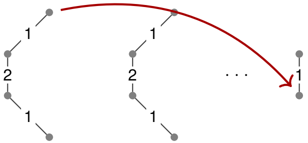
Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)

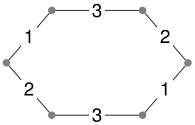


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

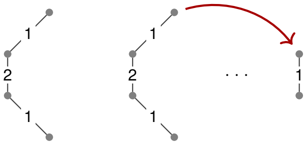
Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)

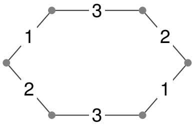


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

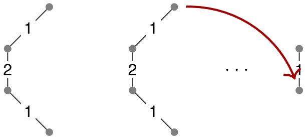
Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)

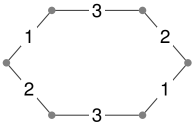


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

Automorphisms of an STG

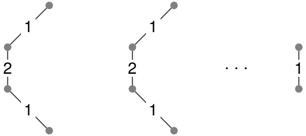
Case 1: The underlying graph is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected

(the complement trick does not works for temporal graphs...)

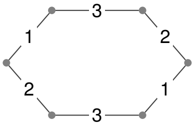


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

Automorphisms of an STG

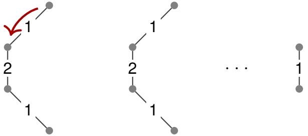
Case 1: The underlying graph is connected.

$\rightarrow$ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected

(the complement trick does not works for temporal graphs...)



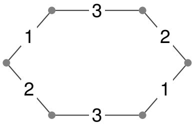
1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

 27/1

Automorphisms of an STG

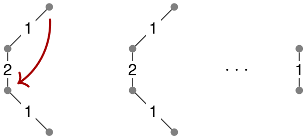
Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected

(the complement trick does not works for temporal graphs...)

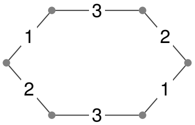


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

Automorphisms of an STG

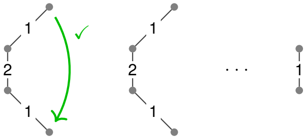
Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected

(the complement trick does not works for temporal graphs...)

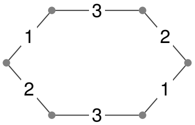


1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

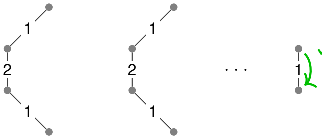
Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)

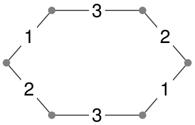


1. Find the underlying components
2. ✓ Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

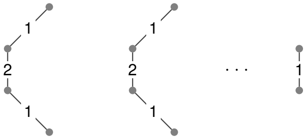
Automorphisms of an STG

Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected
(the complement trick does not works for temporal graphs...)



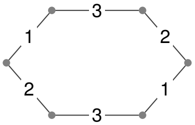
1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

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

Automorphisms of an STG

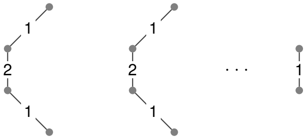
Case 1: The underlying graph is connected.

→ Same strategy as for isomorphism.



Case 2: The underlying graph is not connected

(the complement trick does not works for temporal graphs...)



1. Find the underlying components
2. Search for isomorphisms between pairs of components (remember *one* for each)
3. Find the automorphisms within each component type (trivially extended to $\mathcal{G}$)

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

→ ***Generators for $Aut(\mathcal{G})$ can be computed in polynomial time!***

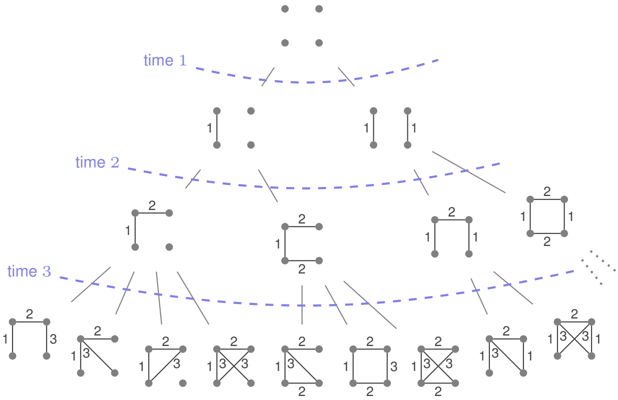
Enumeration up to equivalence

(motivated by conjecture refutation on spanners)

Generation tree

Principle: One level = one time unit

$\rightarrow$ children of a graph $=$ all the possible ways to add the *next time*



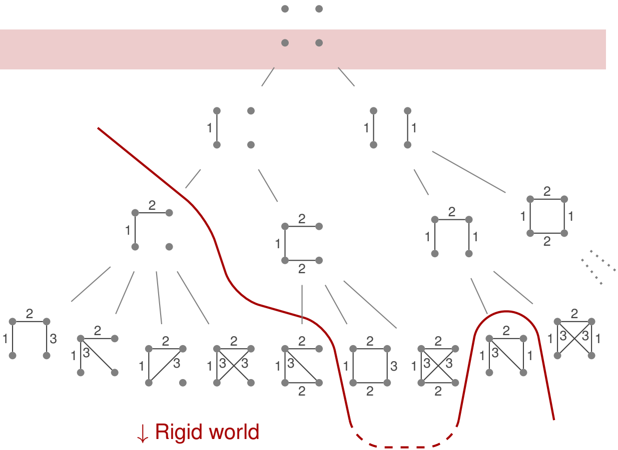
Generation tree

Principle: One level = one time unit

→ children of a graph = all the possible ways to add the *next time*

Key properties

1. Rigidity is inherited



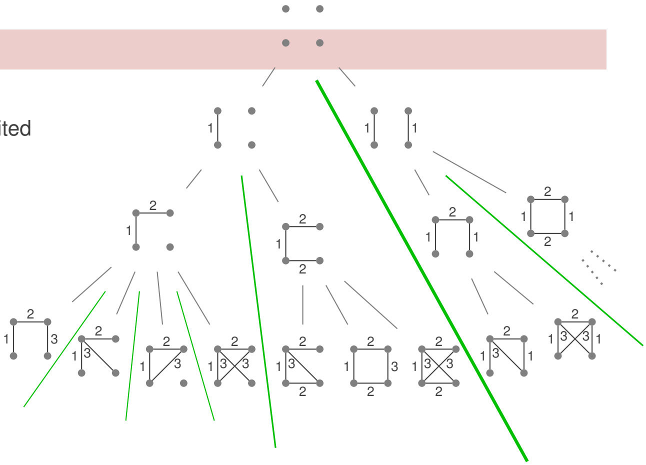
Generation tree

Principle: One level = one time unit

$\rightarrow$ children of a graph $=$ all the possible ways to add the *next time*

Key properties

1. Rigidity is inherited
2. Dissimilarity is inherited



↓ Isomorphism types separated (forever)

How to generate successors at each level?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

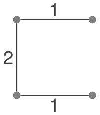
Output: All STG representatives that extend $\mathcal{G}$ with time $t + 1$.

How to generate successors at each level?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

Output: All STG representatives that extend $\mathcal{G}$ with time $t+1$.

First, how to decide if a *non-edge* is eligible to receive a $(t+1)$ time label? (E.g. here, time 3)

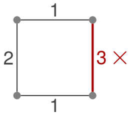


How to generate successors at each level?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

Output: All STG representatives that extend $\mathcal{G}$ with time $t+1$.

First, how to decide if a *non-edge* is eligible to receive a $(t+1)$ time label? (E.g. here, time 3)



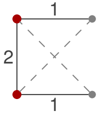
Coloring lemma: (t+1) must be adjacent to (t)

How to generate successors at each level?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

Output: All STG representatives that extend $\mathcal{G}$ with time $t + 1$.

First, how to decide if a *non-edge* is eligible to receive a $(t + 1)$ time label? (E.g. here, time 3)



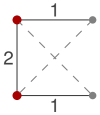
Coloring lemma: (t+1) must be adjacent to (t)

How to generate successors at each level?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

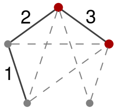
Output: All STG representatives that extend $\mathcal{G}$ with time $t+1$.

First, how to decide if a *non-edge* is eligible to receive a $(t+1)$ time label? (E.g. here, time 3)



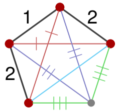
Coloring lemma: (t+1) must be adjacent to (t)

$\mathcal{G}$ is rigid



Two cases

$\mathcal{G}$ has symmetries

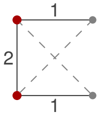


How to generate successors at each level?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

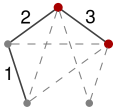
Output: All STG representatives that extend $\mathcal{G}$ with time $t+1$.

First, how to decide if a *non-edge* is eligible to receive a $(t+1)$ time label? (E.g. here, time 3)



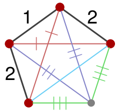
Coloring lemma: (t+1) must be adjacent to (t)

$\mathcal{G}$ is rigid



Two cases

$\mathcal{G}$ has symmetries



→ Enumerate all matchings of eligible *non-edges*.
Each one defines a successor.

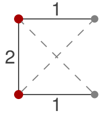
 30/1

How to generate successors at each level?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

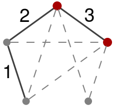
Output: All STG representatives that extend $\mathcal{G}$ with time $t+1$.

First, how to decide if a *non-edge* is eligible to receive a $(t+1)$ time label? (E.g. here, time 3)



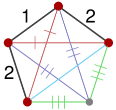
Coloring lemma: (t+1) must be adjacent to (t)

$\mathcal{G}$ is rigid



Two cases

$\mathcal{G}$ has symmetries



$\rightarrow$ Enumerate all matchings of eligible *non-edges*.
Each one defines a successor.

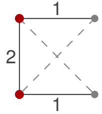
$\equiv$ Independent sets in the *line graph* of eligible *non-edges* (standard algorithm)

How to generate successors at each level?

Input: An STG representative $\mathcal{G}$, whose maximum time is t

Output: All STG representatives that extend $\mathcal{G}$ with time $t+1$.

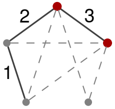
First, how to decide if a *non-edge* is eligible to receive a $(t+1)$ time label? (E.g. here, time 3)



Coloring lemma: (t+1) must be adjacent to (t)

Two cases

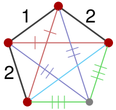
$\mathcal{G}$ is rigid



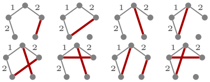
$\rightarrow$ Enumerate all matchings of eligible *non-edges*. Each one defines a successor.

$\equiv$ Independent sets in the *line graph* of eligible *non-edges* (standard algorithm)

$\mathcal{G}$ has symmetries



$\rightarrow$ Enumerate matchings of eligible *non-edges* whose *multisets of orbits* are distinct



Done using the generators for $Aut(\mathcal{G})$

Using the generator

https://github.com/acasteigts/STGen

How to use

```
include("generation.jl")

n = 5
for g in TGraphs(n)
    ...
end
```

Implemented in `Julia`

(other versions also in `Python` and `Java`)

Using the generator

https://github.com/acasteigts/STGen

Implemented in `Julia`
(other versions also in `Python` and `Java`)

How to use

```
include("generation.jl")

n = 5
for g in TGraphs(n)
    ...
end
```

Pruning is possible using `TGraphs(n, selection_predicate)`

Using the generator

https://github.com/acasteigts/STGen

Implemented in `Julia`
(other versions also in `Python` and `Java`)

How to use

```
include("generation.jl")

n = 5
for g in TGraphs(n)
    ...
end
```

Pruning is possible using `TGraphs(n, selection_predicate)`

Back to the spanner question

Do simple temporal cliques admit spanners of size $2n - 3$?

Using the generator

https://github.com/acasteigts/STGen

Implemented in `Julia`
(other versions also in `Python` and `Java`)

How to use

```
include("generation.jl")

n = 5
for g in TGraphs(n)
    ...
end
```

Pruning is possible using `TGraphs(n, selection_predicate)`

Back to the spanner question

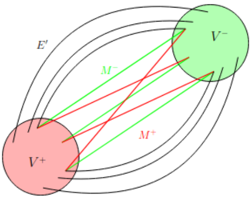
Do simple temporal cliques admit spanners of size $2n - 3$?

$\rightarrow$ True for $n \leq 7$ (and for all non-rigid graphs at $n = 8$). **Otherwise still open! :-)**

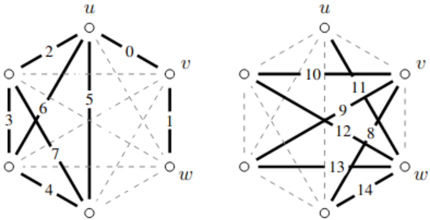
Some numbers

# Vertices	# STGs	# Temporally connected STGs	# Simple Temporal cliques
1	1	1	1
2	2	1	1
3	4	1	1
4	62	32	20
5	15378	10207	4524
6	89769096	70557834	23218501
7	13828417028594	?	3129434545680
8	?	?	?

Non dismountable clique:



Non pivotable clique (seen as a union of two graphs):



Thanks!

next time... :-)