Spanner problems in temporal graphs

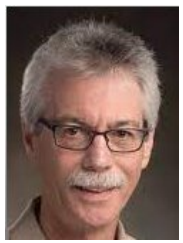
Arnaud Casteigts

LaBRI, Université de Bordeaux

November 9, 2021

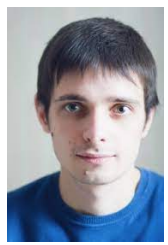
Based on joint works with:


Jason Schoeters


Joseph Peters


Michael Raskin

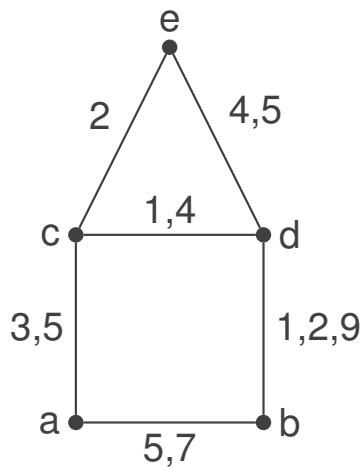

Malte Renken


Viktor Zamaraev

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 (here, discrete)

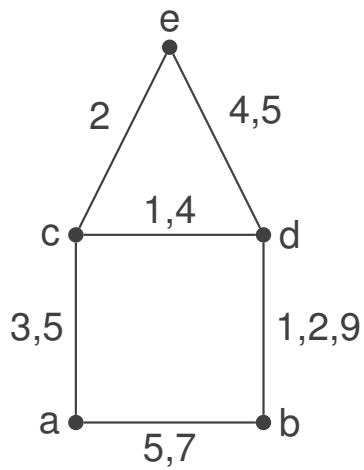
Example:



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

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

Example:



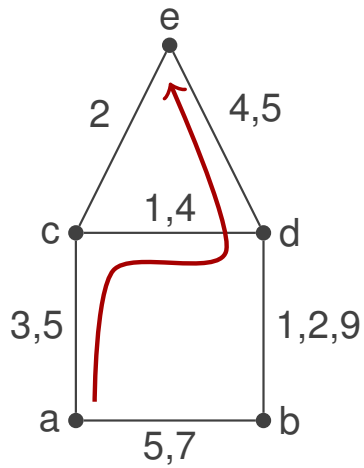
Temporal paths

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

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 (here, discrete)

Example:



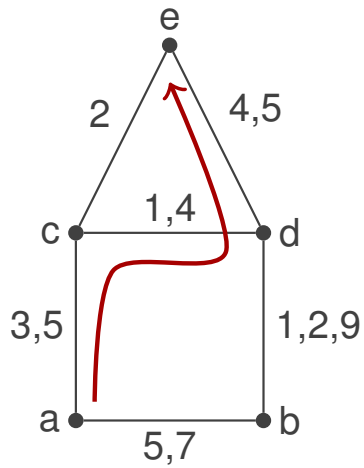
Temporal paths

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

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 (here, discrete)

Example:



Temporal paths

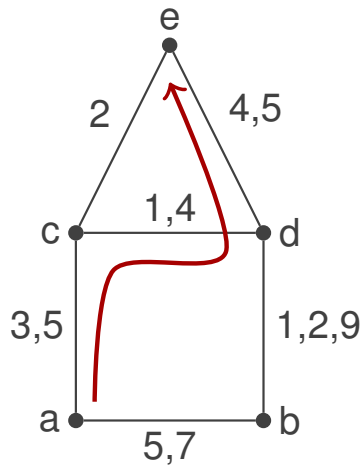
- Ex: $\langle (a, c, 3), (c, d, 4), (d, e, 4) \rangle$ (*non-decreasing*)
- Ex: $\langle (a, c, 3), (c, d, 4), (d, e, 5) \rangle$ **(*increasing*)**

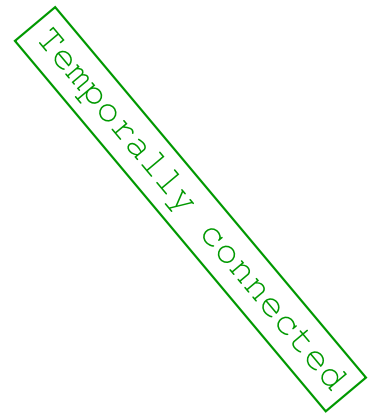
Temporal connectivity: all vertices can reach each other through temporal paths

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 (here, discrete)

Example:





Temporal paths

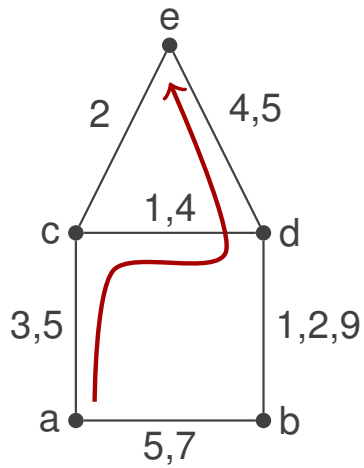
- Ex: $\langle (a, c, 3), (c, d, 4), (d, e, 4) \rangle$ (*non-decreasing*)
- Ex: $\langle (a, c, 3), (c, d, 4), (d, e, 5) \rangle$ **(*increasing*)**

Temporal connectivity: all vertices can reach each other through temporal paths

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 (here, discrete)

Example:



Temporally connected

Temporal paths

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

Temporal connectivity: all vertices can reach each other through temporal paths

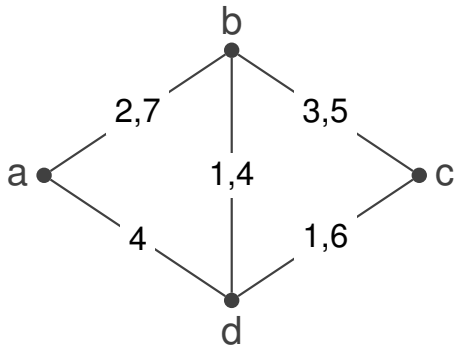
Remark: reachability is non-transitive in general!

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 (in number of time labels)

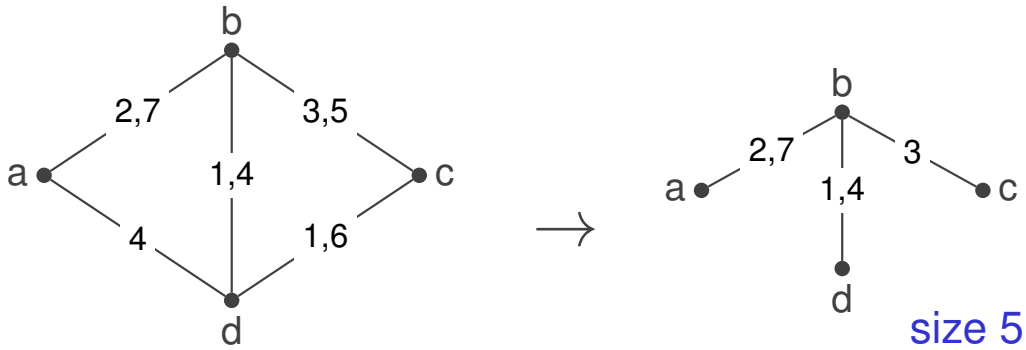


Temporal spanners

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 (in number of time labels)

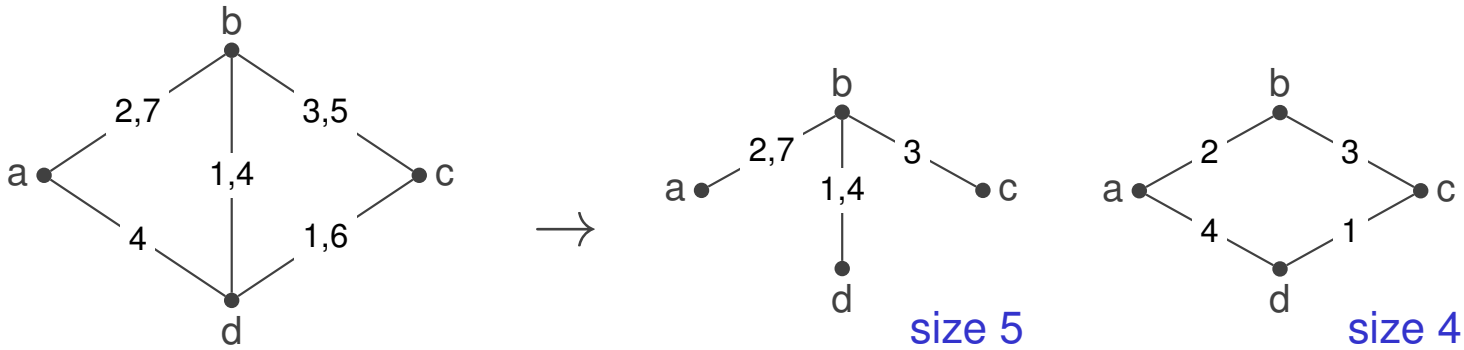


Temporal spanners

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 (in number of time labels)

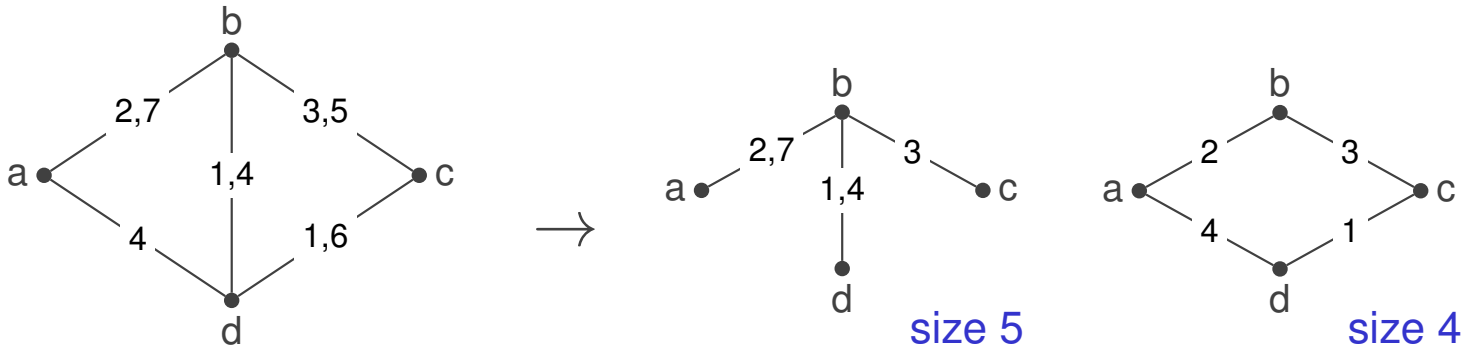


Temporal spanners

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 (in number of time labels)



Can we do better?

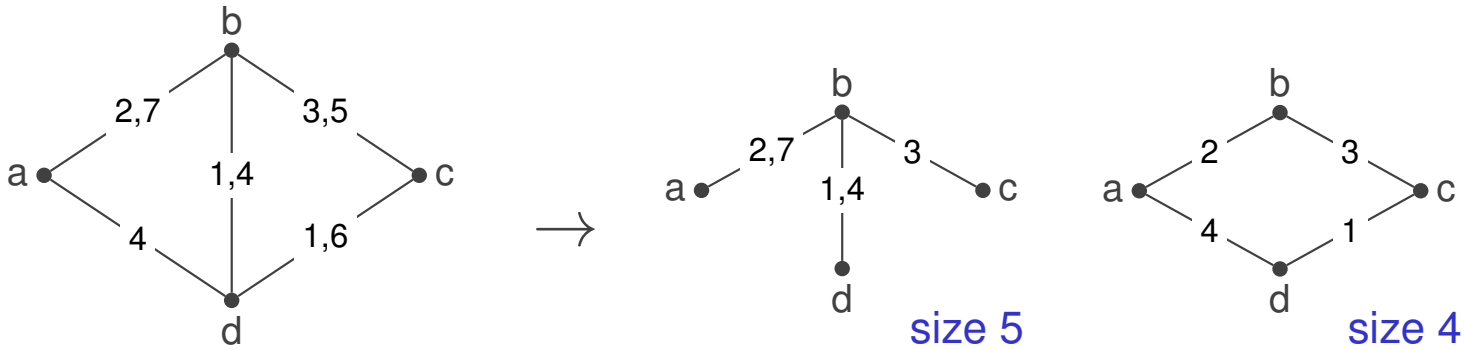
- $2n - 4$ labels needed, even if you choose the values! (Bumby'79, gossip theory)

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 (in number of time labels)



Can we do better?

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

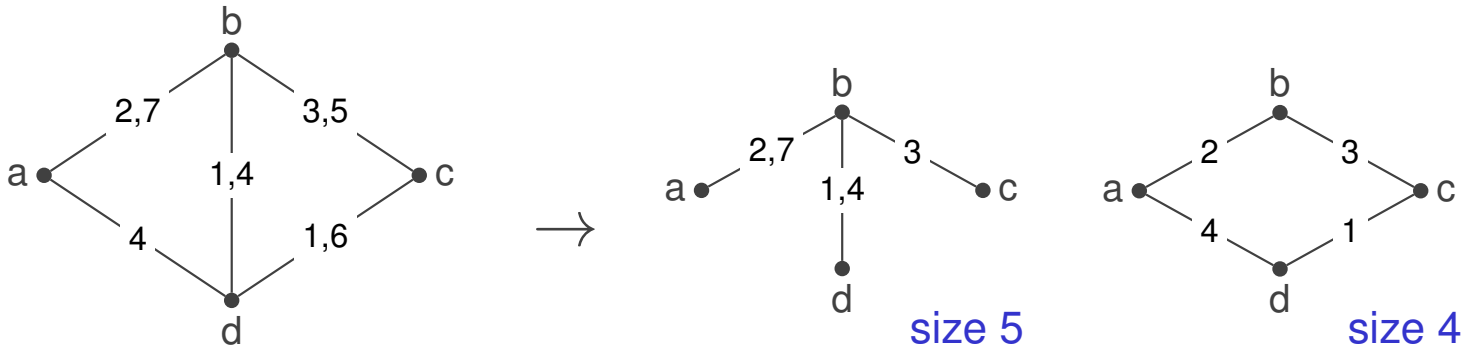
Do spanners of size $2n - 4$ always exist?

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 (in number of time 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?

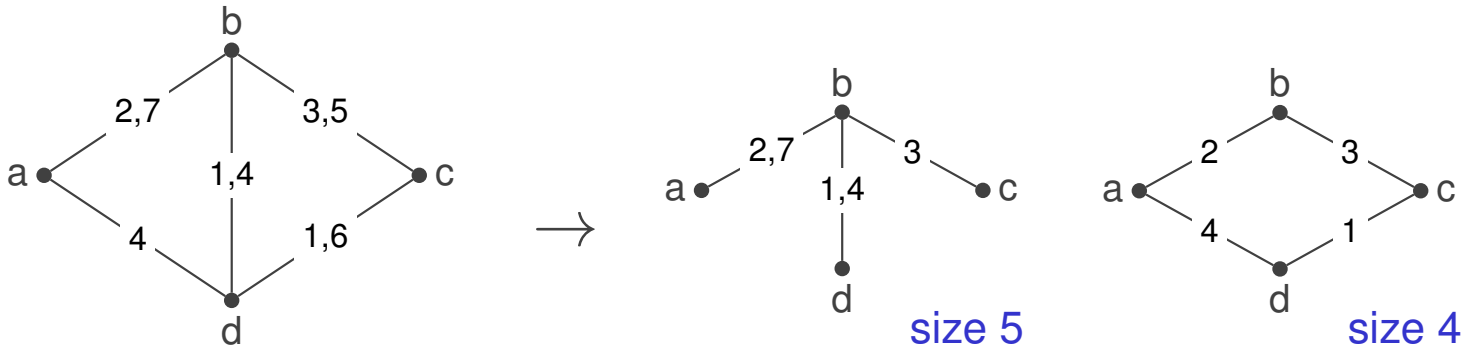
- $\Omega(n \log n)$ in some cases (Kleinberg, Kempe, Kumar, 2000)

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 (in number of time 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?

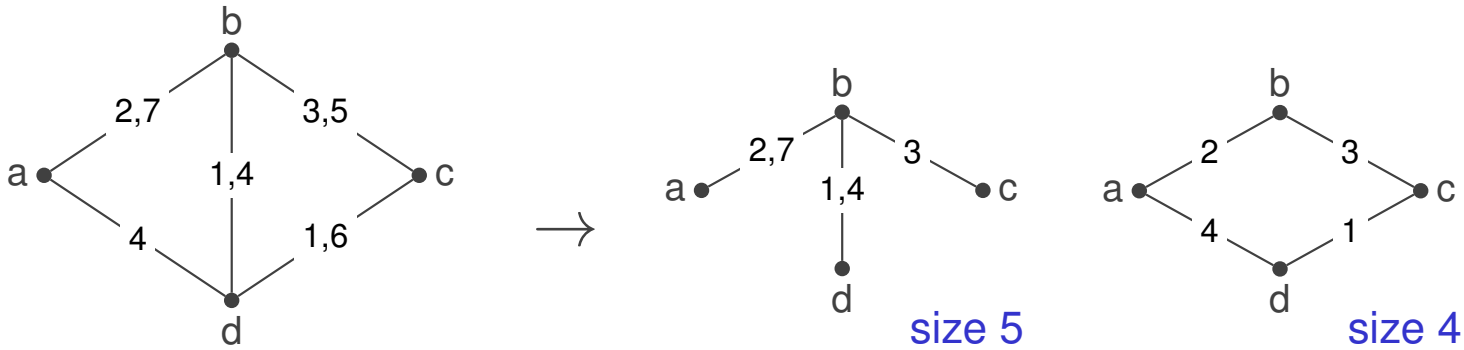
- $\Omega(n \log n)$ in some cases (Kleinberg, Kempe, Kumar, 2000)
- $\Omega(n^2)$ in some cases! (Axiotis, Fotakis, 2016)

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 (in number of time 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?

- $\Omega(n \log n)$ in some cases (Kleinberg, Kempe, Kumar, 2000)
- $\Omega(n^2)$ in some cases! (Axiotis, Fotakis, 2016)

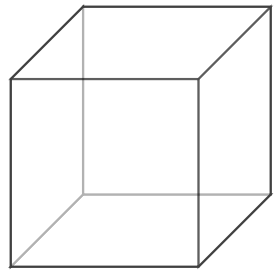
How about complexity?

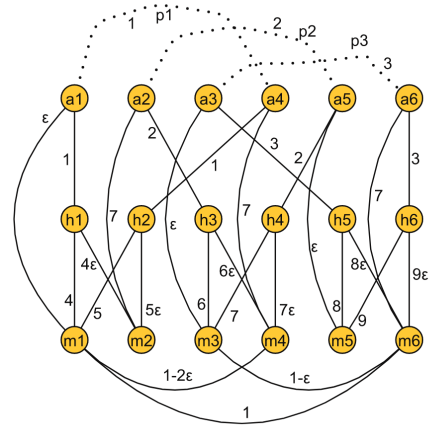
- Computing minimum-size spanner is APX-hard (Akrida et al., 2017)

What about the positive side?

Recall the bad news:

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

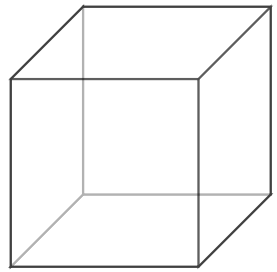


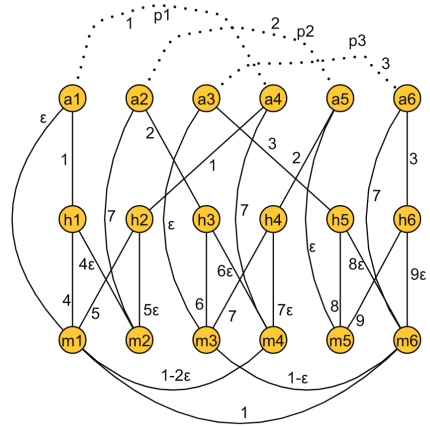


What about the positive side?

Recall the bad news:

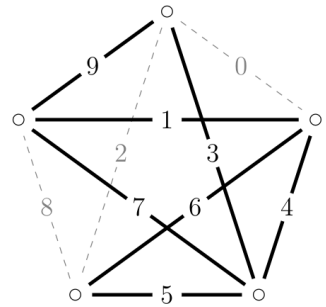
- $\Omega(n \log n)$ - easy
- $\Omega(n^2)$ - rather unexpected





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

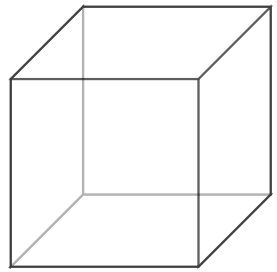
- Spanners of size $O(n \log n)$ always exist in **complete** temporal graphs

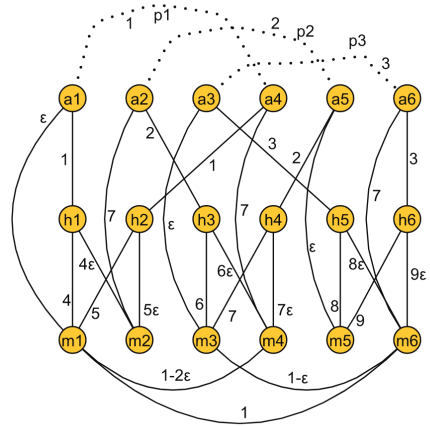


What about the positive side?

Recall the bad news:

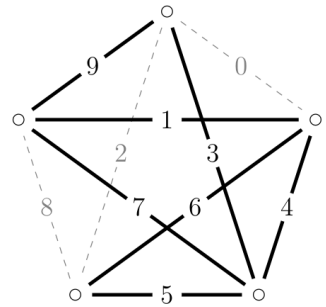
- $\Omega(n \log n)$ - easy
- $\Omega(n^2)$ - rather unexpected





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

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



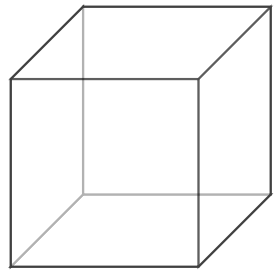
Good news 2: (C., Raskin, Renken, Zamaraev, FOCS 2021):

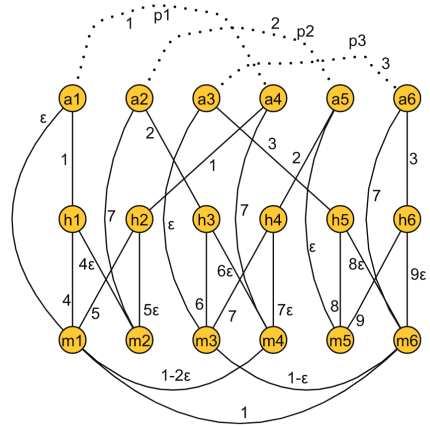
- Nearly optimal spanners (of size $2n + o(n)$) almost surely exist in **random** temporal graphs, as soon as the graph is temporally connected

What about the positive side?

Recall the bad news:

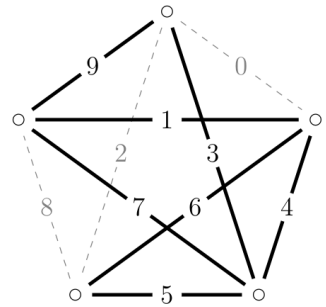
- $\Omega(n \log n)$ - easy
- $\Omega(n^2)$ - rather unexpected





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

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



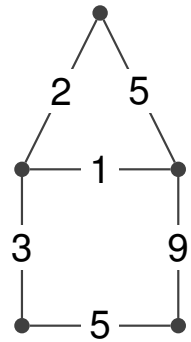
Good news 2: (C., Raskin, Renken, Zamaraev, FOCS 2021):

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

Before we start... an easier model

Simple Temporal Graphs (STGs):

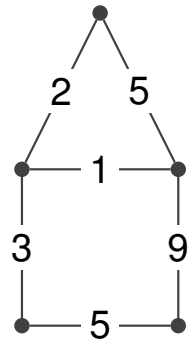
1. A single presence time per edge ($\lambda : E \to \mathbb{N}$)
2. Adjacent edges have different times (λ is locally injective)



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)



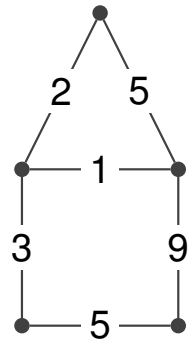
Generality for spanners:

- 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:

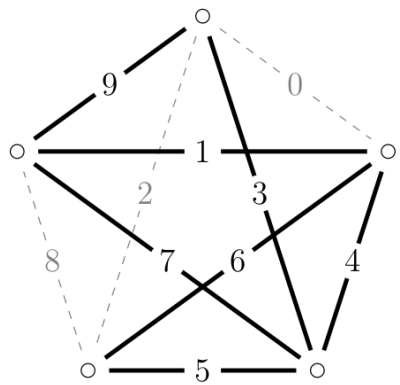
- Most negative results still apply
- Positive results extend to general case
- No distinction between strict and non-strict temporal paths

Further motivations:

- Distributed models by pairwise interactions,
  e.g. population protocols or gossip models (without repetition)
- Topics in edge-ordered graphs

Good news 1:

Temporal cliques admit sparse spanners



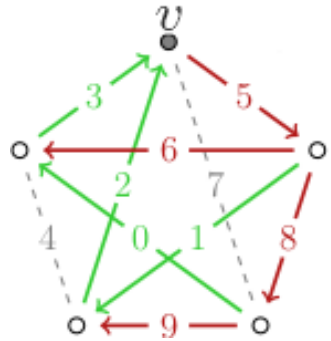
Two promising techniques...

Pivotability

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 is a spanner of size $2n - 2$



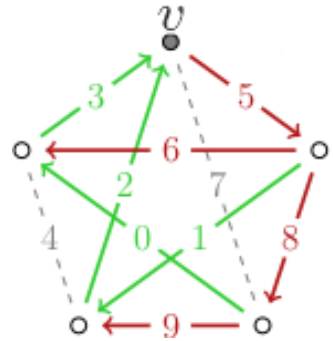
Two promising techniques...

Pivotability

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 is a spanner of size $2n - 2$

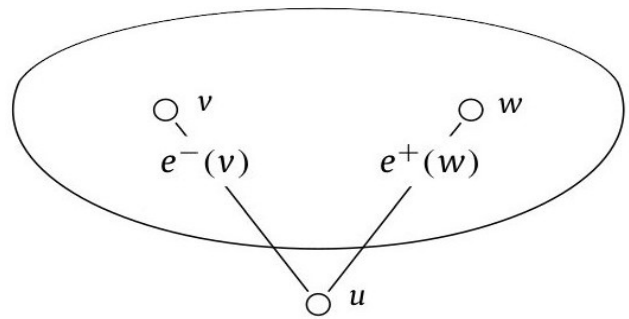


Dismountability

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



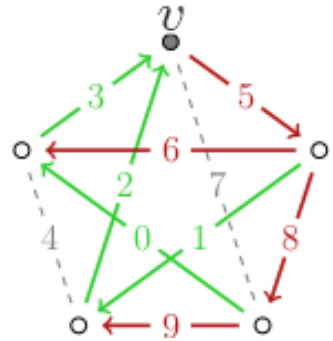
Two promising techniques...

Pivotability

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 is a spanner of size $2n - 2$

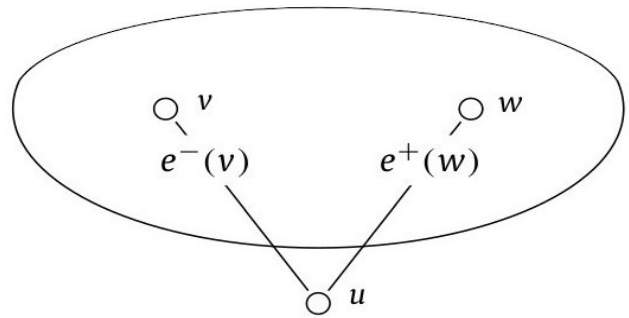


Dismountability

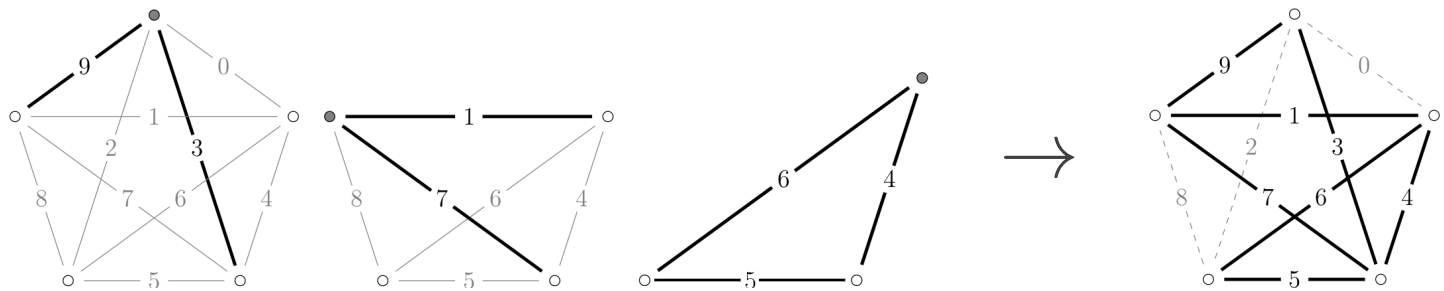
Three nodes u, v, w such that:

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

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



Recursively,



spanner of size $2n - 3$.

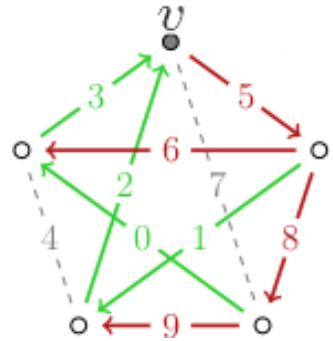
Two promising techniques...

Pivotability

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 is a spanner of size $2n - 2$

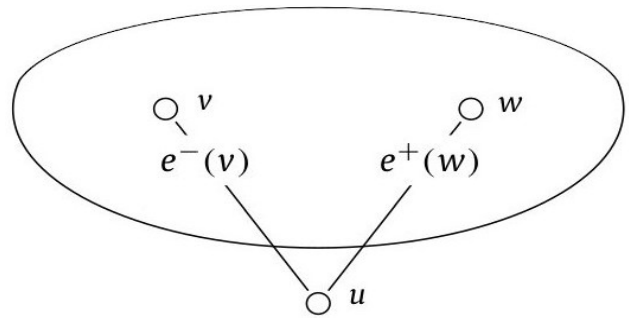


Dismountability

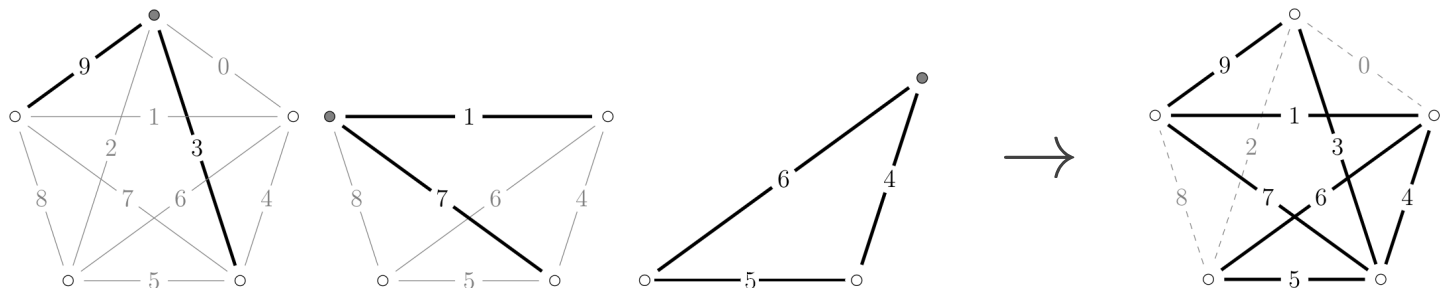
Three nodes u, v, w such that:

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

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



Recursively,



spanner of size $2n - 3$.

... but unfortunately

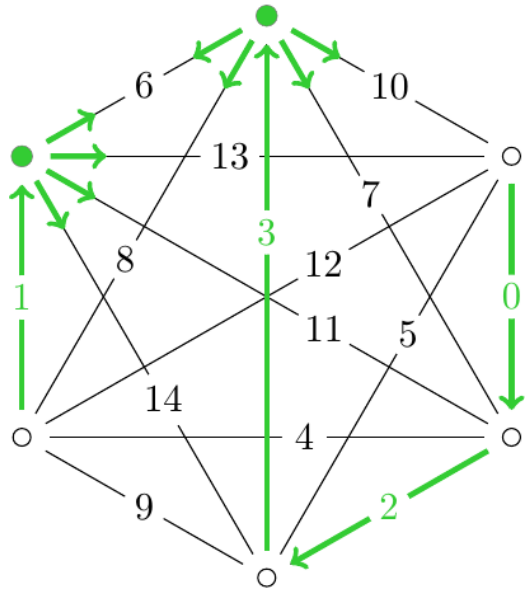
Both techniques fail in some cliques!

Transitive delegations (“fireworks”)

Transitive delegations (“fireworks”)

Principle:

- Min edges $\rightarrow$ “directed” forest
- Transitive delegations towards emitters (sinks)
- Spanner = min edges + all edges of emitters

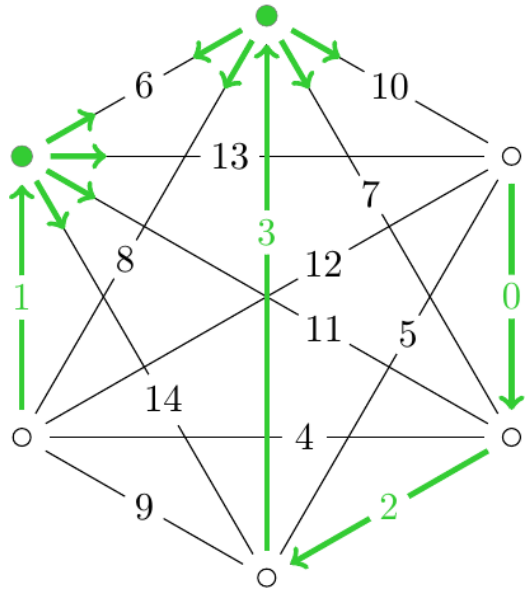


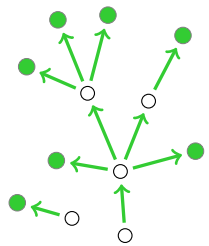
Transitive delegations (“fireworks”)

Principle:

- Min edges $\rightarrow$ “directed” forest
- Transitive delegations towards emitters (sinks)
- Spanner = min edges + all edges of emitters

Wait a minute... possibly too many emitters!

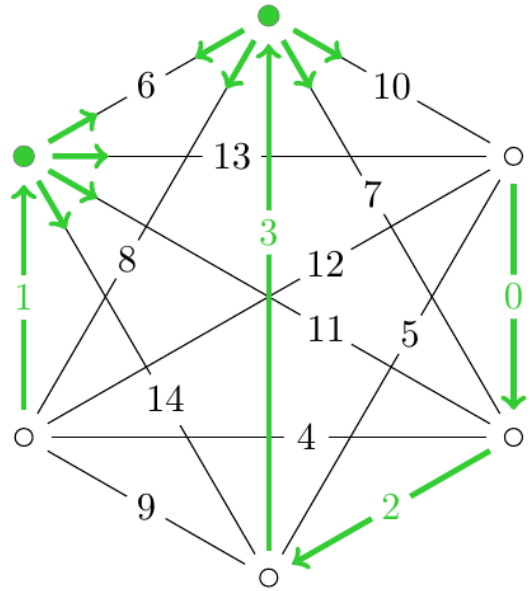




Transitive delegations (“fireworks”)

Principle:

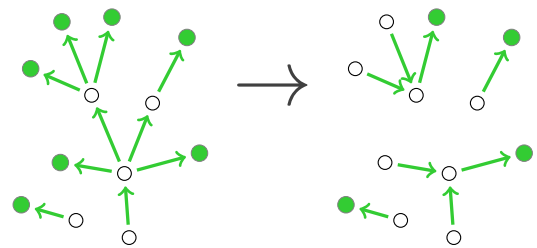
- Min edges $\rightarrow$ “directed” forest
- Transitive delegations towards emitters (sinks)
- Spanner = min edges + all edges of emitters



Wait a minute... possibly too many emitters!

$\rightarrow$ *Transformation of the forest*:

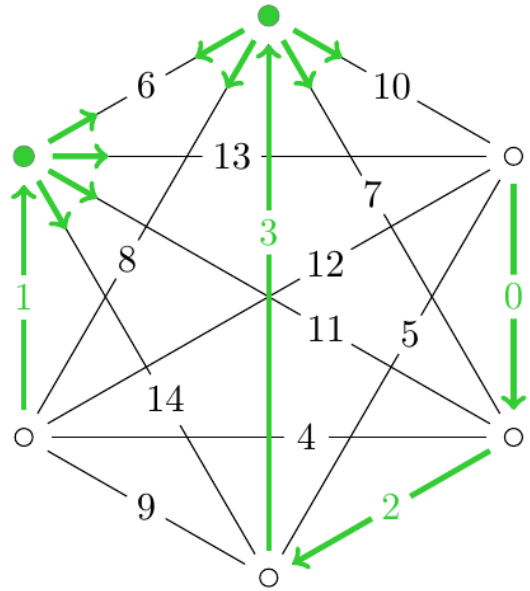
- At most $n/2$ emitters



Transitive delegations (“fireworks”)

Principle:

- Min edges $\rightarrow$ “directed” forest
- Transitive delegations towards emitters (sinks)
- Spanner = min edges + all edges of emitters

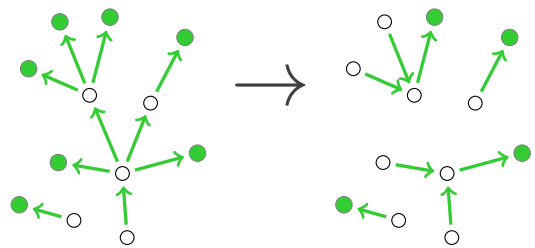


Wait a minute... possibly too many emitters!

$\rightarrow$ *Transformation of the forest*:

- At most $n/2$ emitters

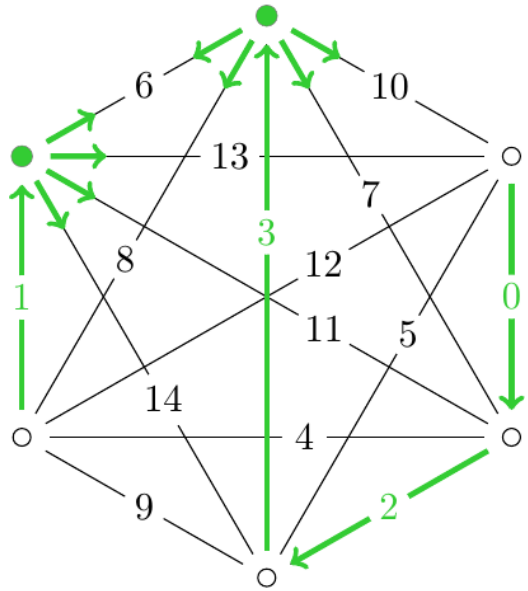
Theorem: $\exists$ spanners of size $\frac{3}{4}\binom{n}{2} + O(n)$



Transitive delegations (“fireworks”)

Principle:

- Min edges $\rightarrow$ “directed” forest
- Transitive delegations towards emitters (sinks)
- Spanner = min edges + all edges of emitters

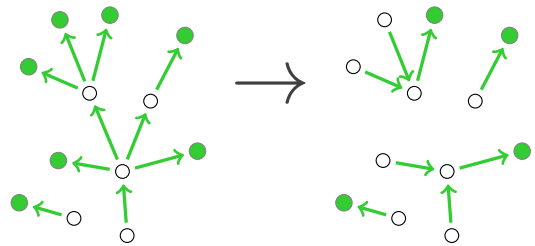


Wait a minute... possibly too many emitters!

$\rightarrow$ *Transformation of the forest*:

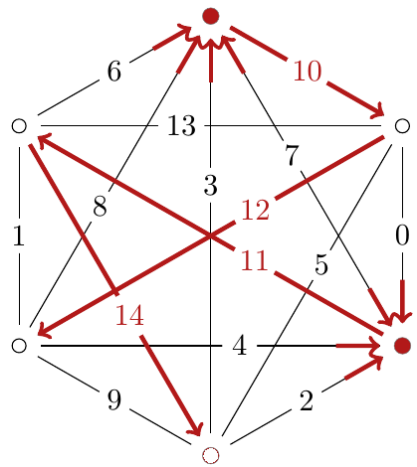
- At most $n/2$ emitters

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



Note: also works for receptions (“backward fireworks”):

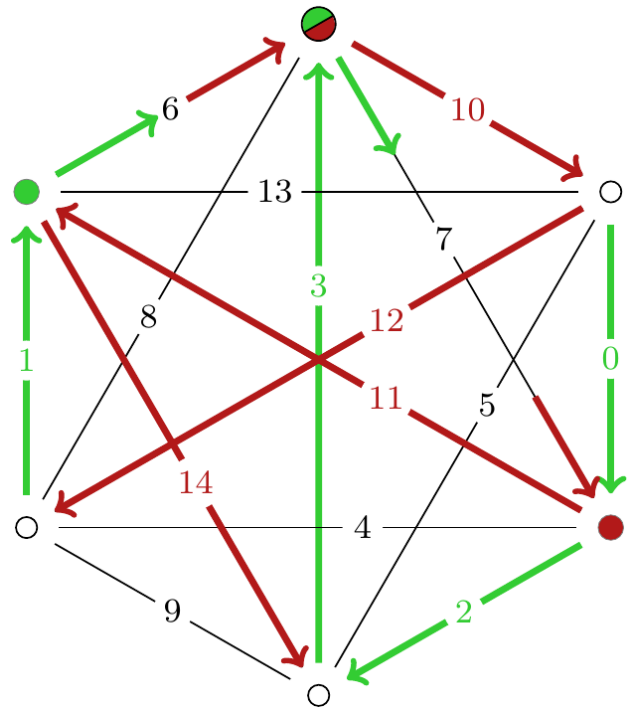
$\rightarrow$ Spanner = max edges + all edges of collectors



Combining both directions

Principle

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



$\rightarrow$ Spanner = min edges + max edges

+ edges between emitters and collectors

Theorem:

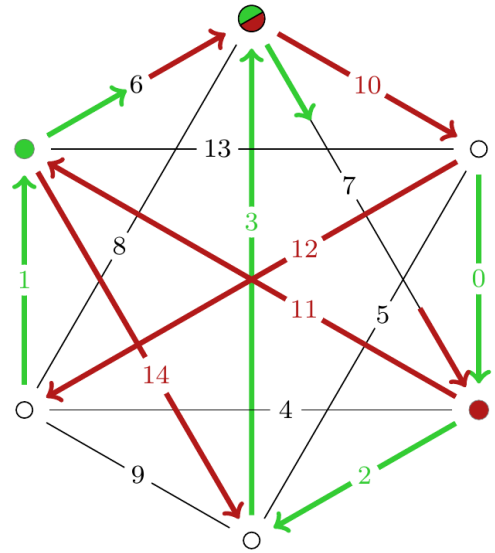
At most $n/2$ emitters and $n/2$ collectors $\Rightarrow$ $\boxed{\exists \text{ Spanners of size } \binom{n}{2}/2 + O(n)}$

$\approx$ half of the edges

Recurse or sparsify?

Two options:

- Case 1: emitters $\cup$ collectors $\subsetneq V$
- Case 2: emitters $\cup$ collectors $= V$



Recurse or sparsify?

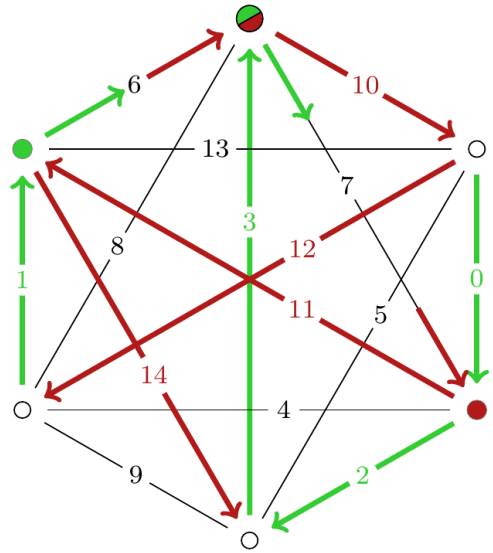
Two options:

- Case 1: emitters $\cup$ collectors $\subsetneq V$
- Case 2: emitters $\cup$ collectors $= V$

Case 1: One vertex v is neither emitter nor collector.

$\rightarrow v$ is "2-*hop dismountable*"

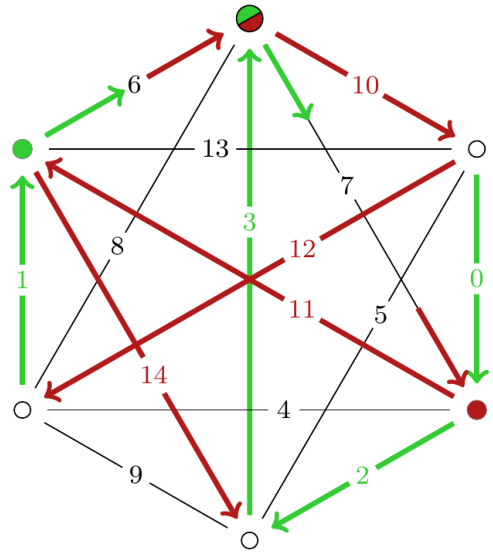
(select 4 edges selected, then recurse in $\mathcal{G}[V - v]$)



Recurse or sparsify?

Two options:

- Case 1: emitters $\cup$ collectors $\subsetneq V$
- Case 2: emitters $\cup$ collectors $= V$



Case 1: One vertex v is neither emitter nor collector.

$\rightarrow v$ is "2-*hop dismountable*"

(select 4 edges selected, then recurse in $\mathcal{G}[V - v]$)

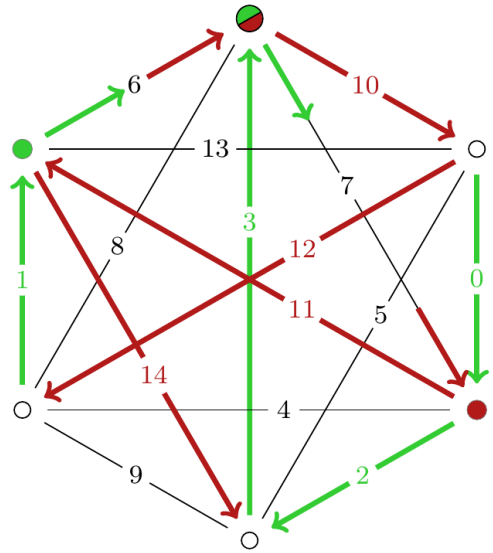
Case 2: emitters $\cup$ collectors $= V$

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

Recurse or sparsify?

Two options:

- Case 1: emitters $\cup$ collectors $\subsetneq V$
- Case 2: emitters $\cup$ collectors $= V$



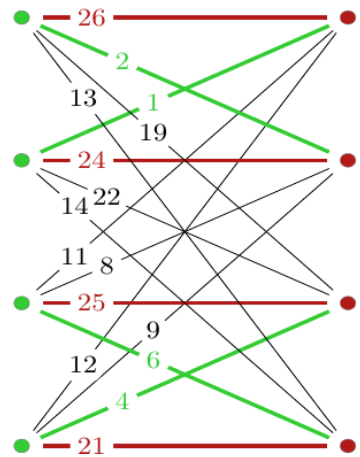
Case 1: One vertex v is neither emitter nor collector.

$\rightarrow v$ is "2-*hop dismountable*"

(select 4 edges selected, then recurse in $\mathcal{G}[V - v]$)

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

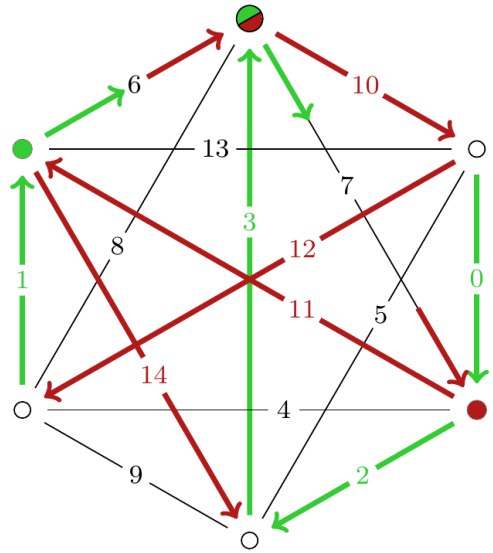
$\rightarrow$ All vertices are either emitters or collectors (not both)!



Recurse or sparsify?

Two options:

- Case 1: emitters $\cup$ collectors $\subsetneq V$
- Case 2: emitters $\cup$ collectors $= V$



Case 1: One vertex v is neither emitter nor collector.

$\rightarrow v$ is "2-*hop dismountable*"

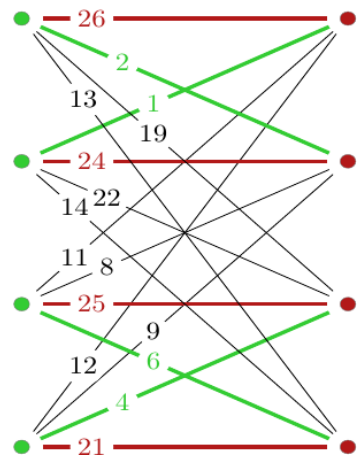
(select 4 edges selected, then recurse in $\mathcal{G}[V - v]$)

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

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

A lot of structure to work with:

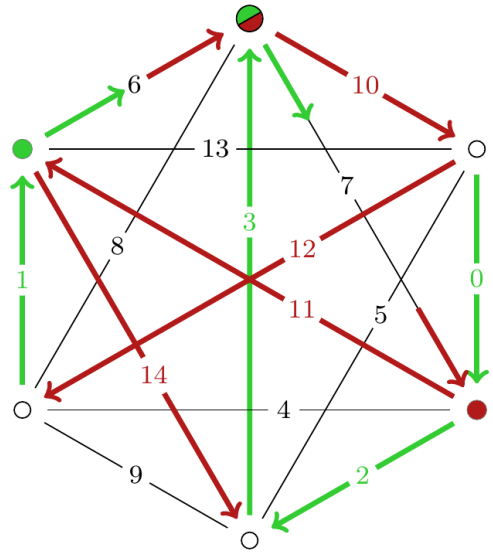
- Complete bipartite graph $\mathcal{H}$ between emitters and collectors
- Min edges and max edges form two perfect matchings
- W.l.o.g. min edges (max edges) are *reciprocal* in $\mathcal{H}$



Recurse or sparsify?

Two options:

- Case 1: emitters $\cup$ collectors $\subsetneq V$
- Case 2: emitters $\cup$ collectors $= V$



Case 1: One vertex v is neither emitter nor collector.

$\rightarrow v$ is "2-*hop dismountable*"

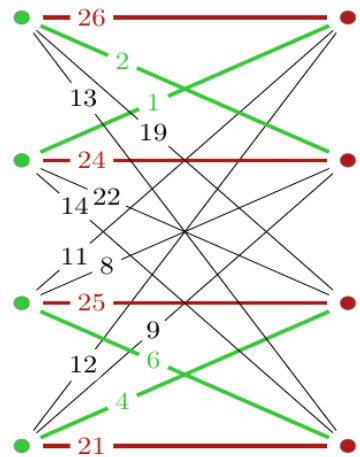
(select 4 edges selected, then recurse in $\mathcal{G}[V - v]$)

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 $\mathcal{H}$



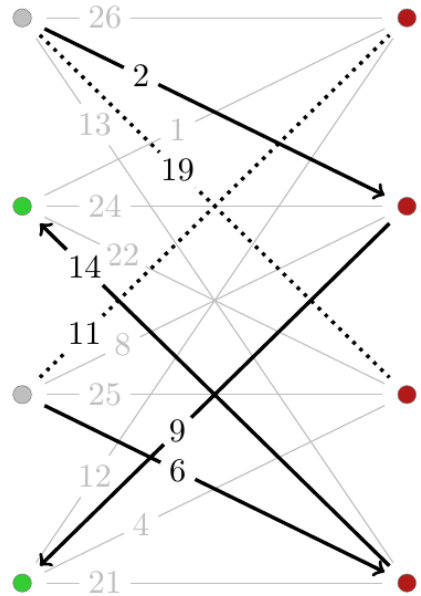
New objective:

$\rightarrow$ 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



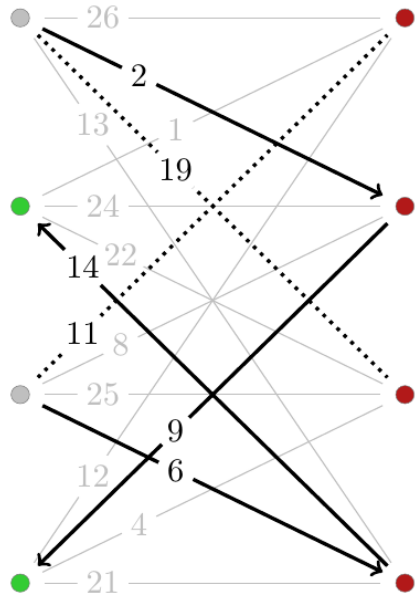
Technique: Partial delegations among emitters

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

Iterative procedure:

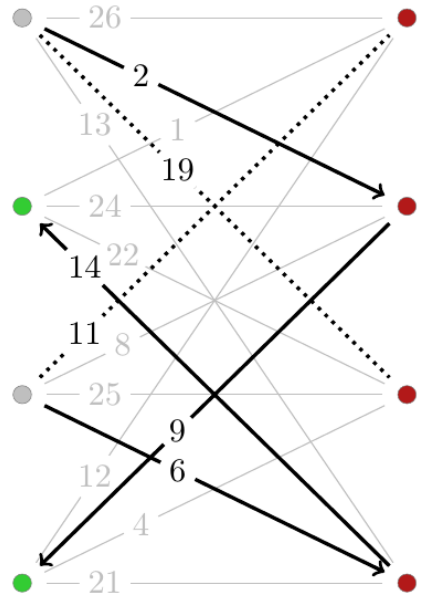
In each step i:

- Half of the emitters partially delegate to other half
- We pay direct edges to missed collectors (penalty)
- 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 to reach the missed collectors

Iterative procedure:

In each step i:

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

Conclusion (entire algorithm):

$\rightarrow$ $\exists$ spanner of size $O(n) + O(n \log n) = \boldsymbol{O(n \log n)}$. $\square$

Good news 2:

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

in random temporal graphs

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)

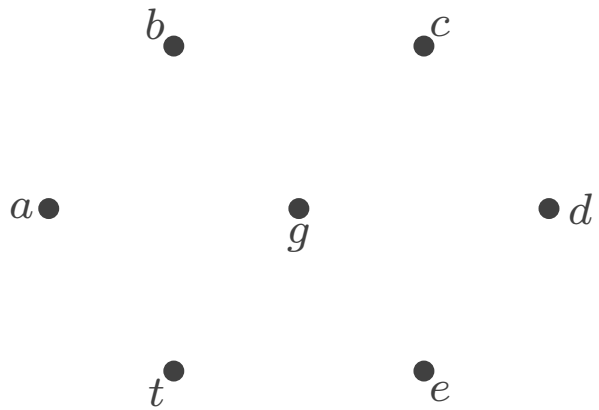
Random Simple Temporal Graphs (RSTGs)

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: $n = 7, p = 0.4$



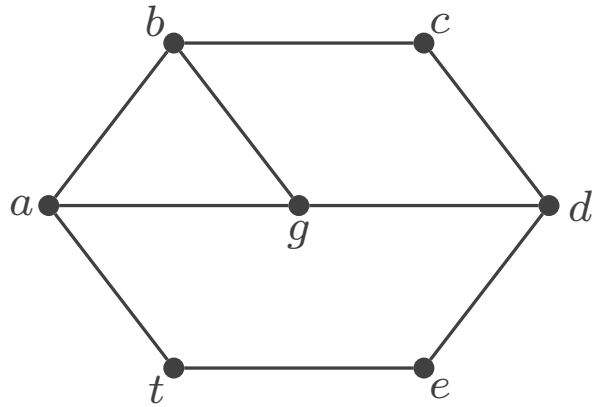
Random Simple Temporal Graphs (RSTGs)

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: $n = 7, p = 0.4$



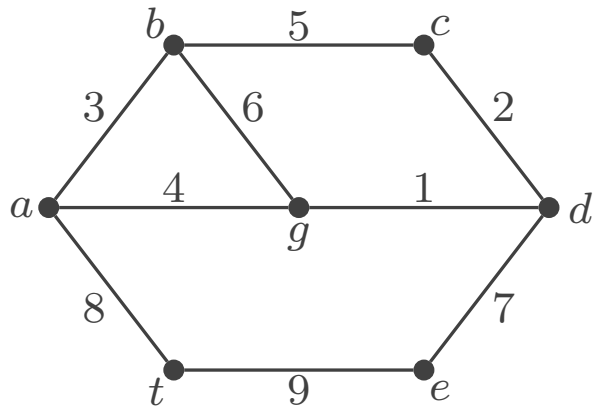
Random Simple Temporal Graphs (RSTGs)

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: $n = 7, p = 0.4$



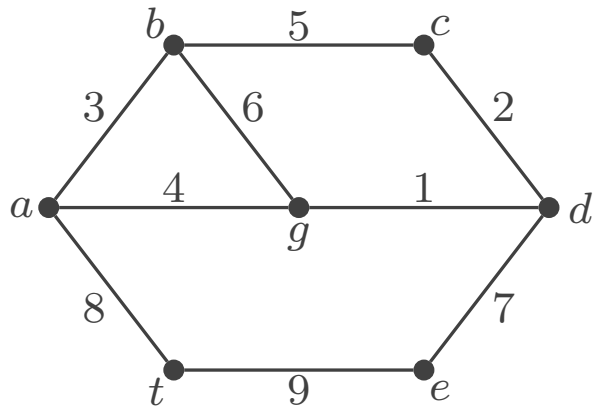
Random Simple Temporal Graphs (RSTGs)

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 $\mathcal{G}_{[0,p]}$

$\rightarrow$ Better for analysis.

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

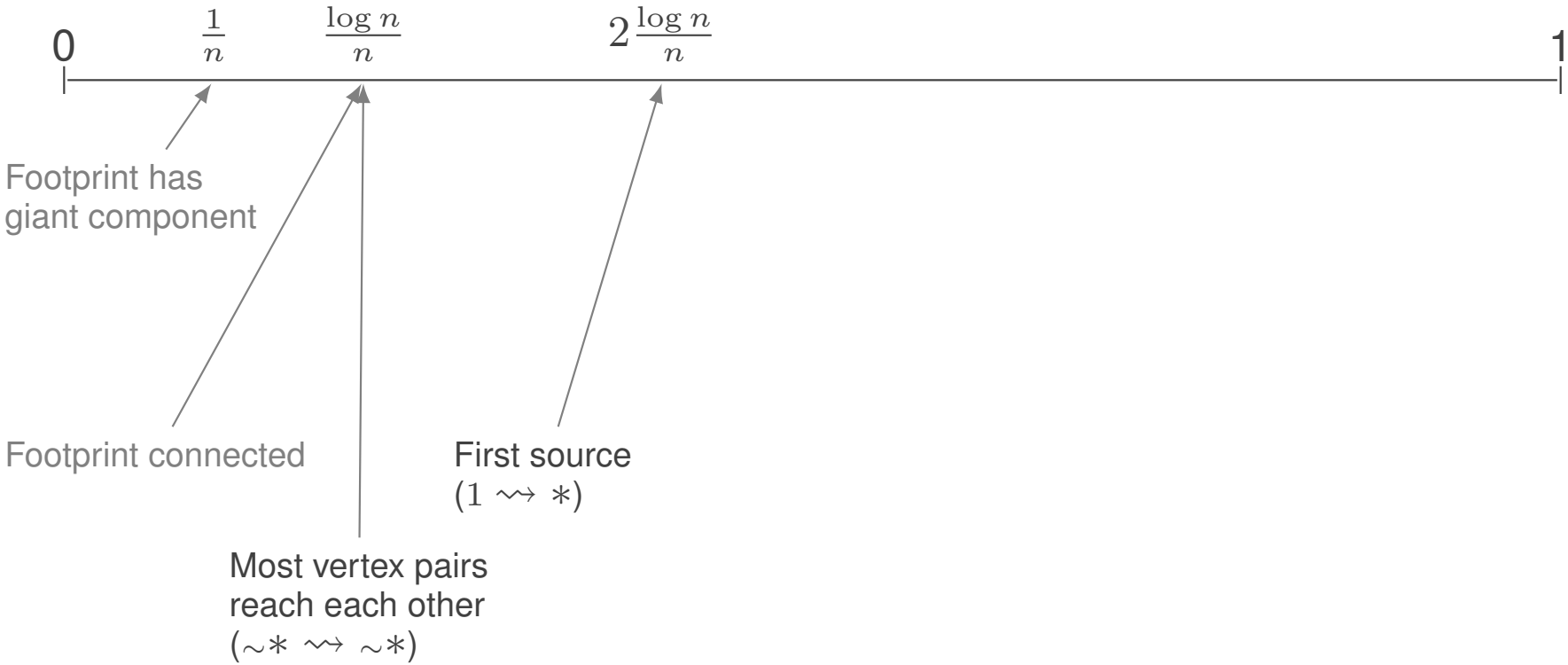
For sufficiently large n, what happens when p increases?

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

For sufficiently large n, what happens when p increases?

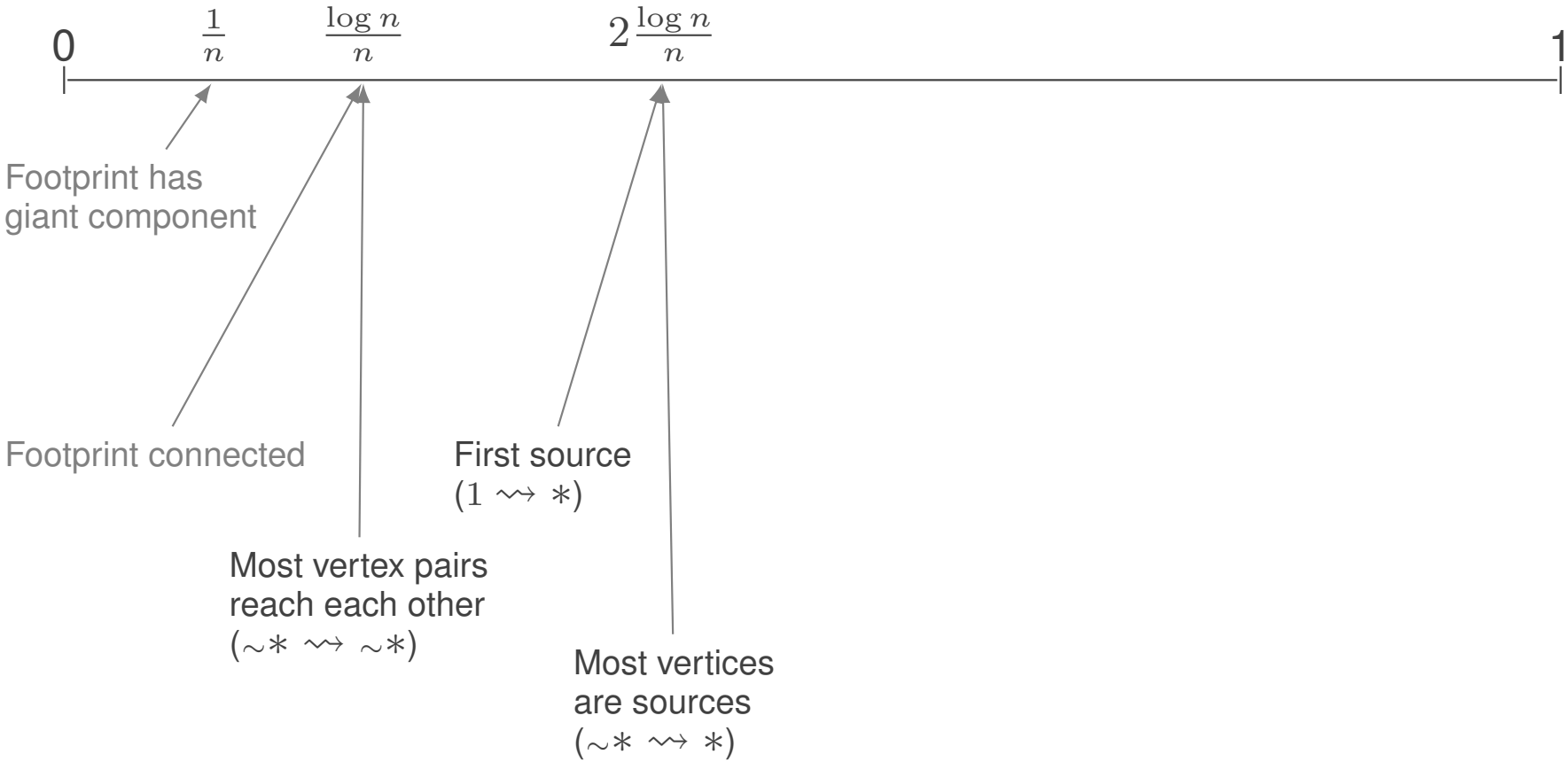
Timeline of temporal reachability in $\mathcal{G}_{n,p}$

For sufficiently large n, what happens when p increases?



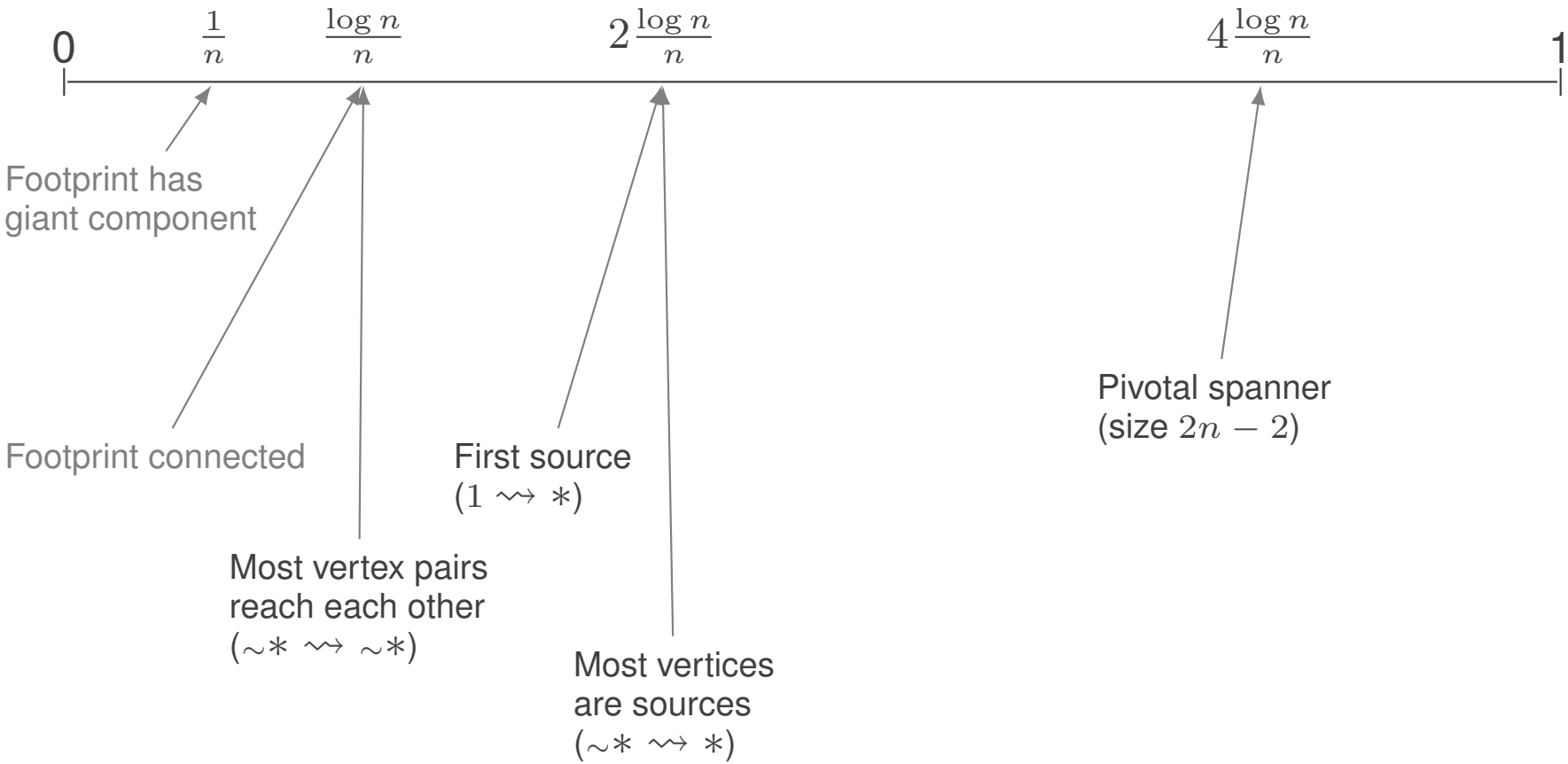
Timeline of temporal reachability in $\mathcal{G}_{n,p}$

For sufficiently large n, what happens when p increases?



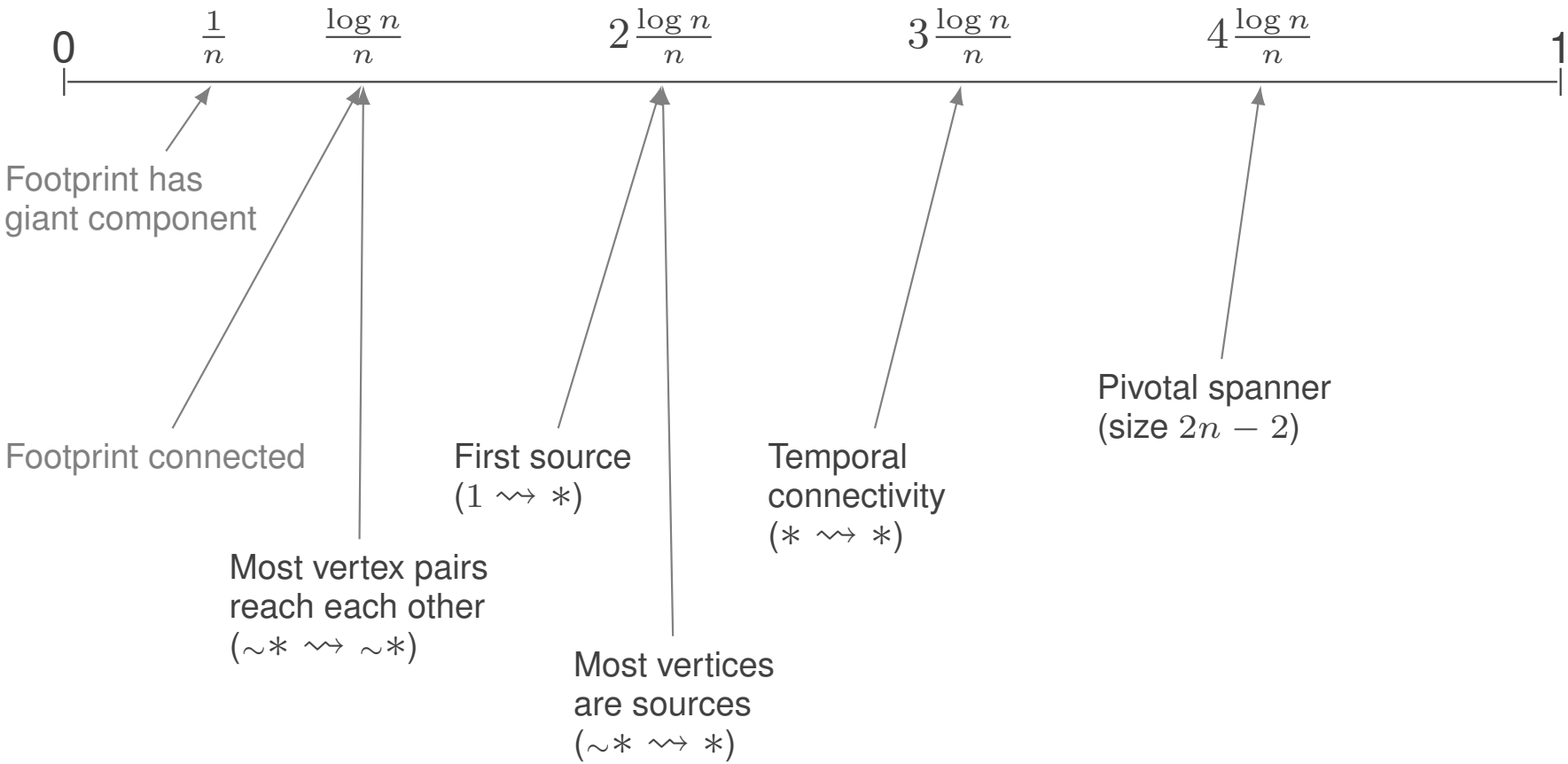
Timeline of temporal reachability in $\mathcal{G}_{n,p}$

For sufficiently large n, what happens when p increases?



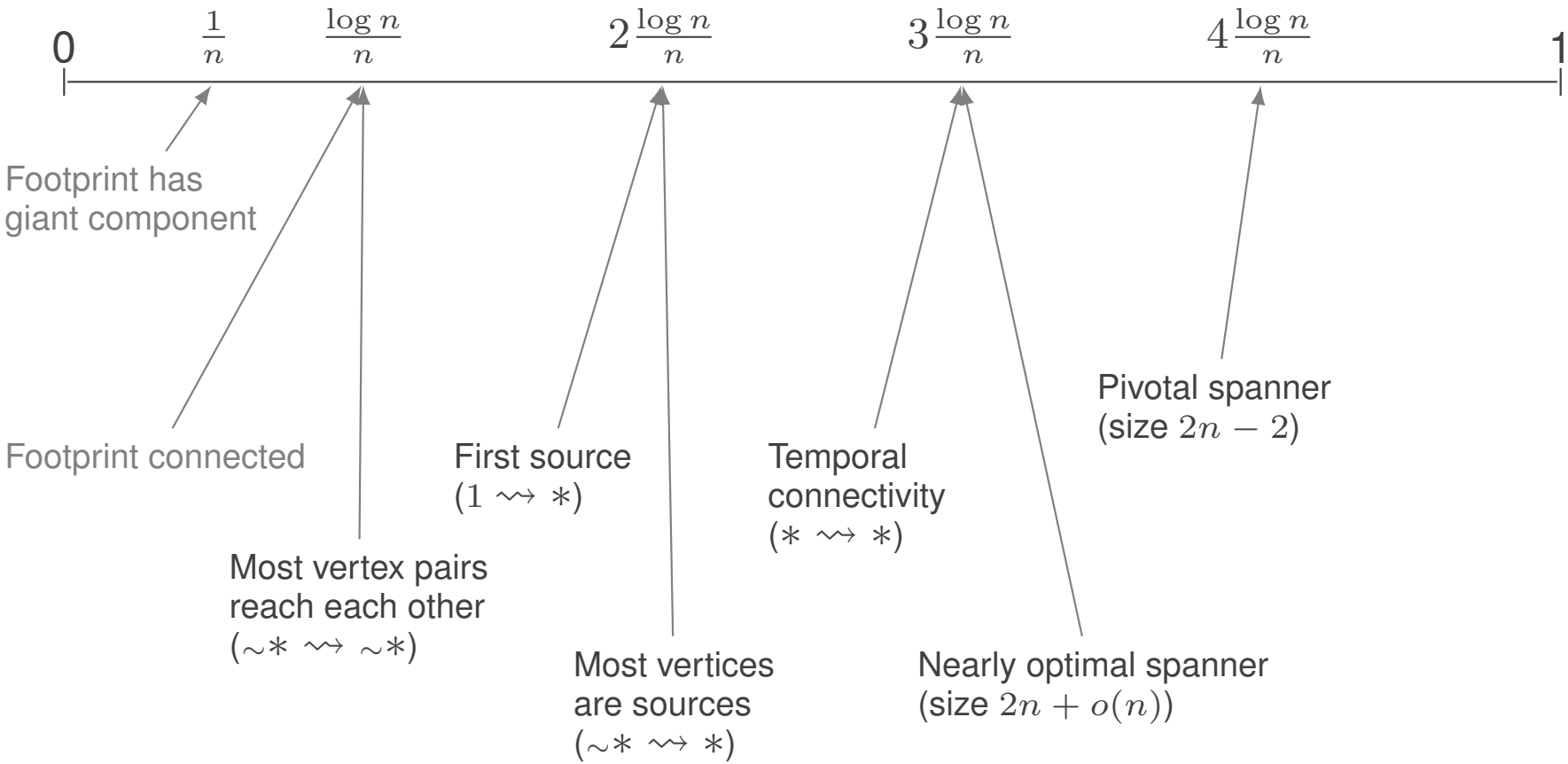
Timeline of temporal reachability in $\mathcal{G}_{n,p}$

For sufficiently large n, what happens when p increases?



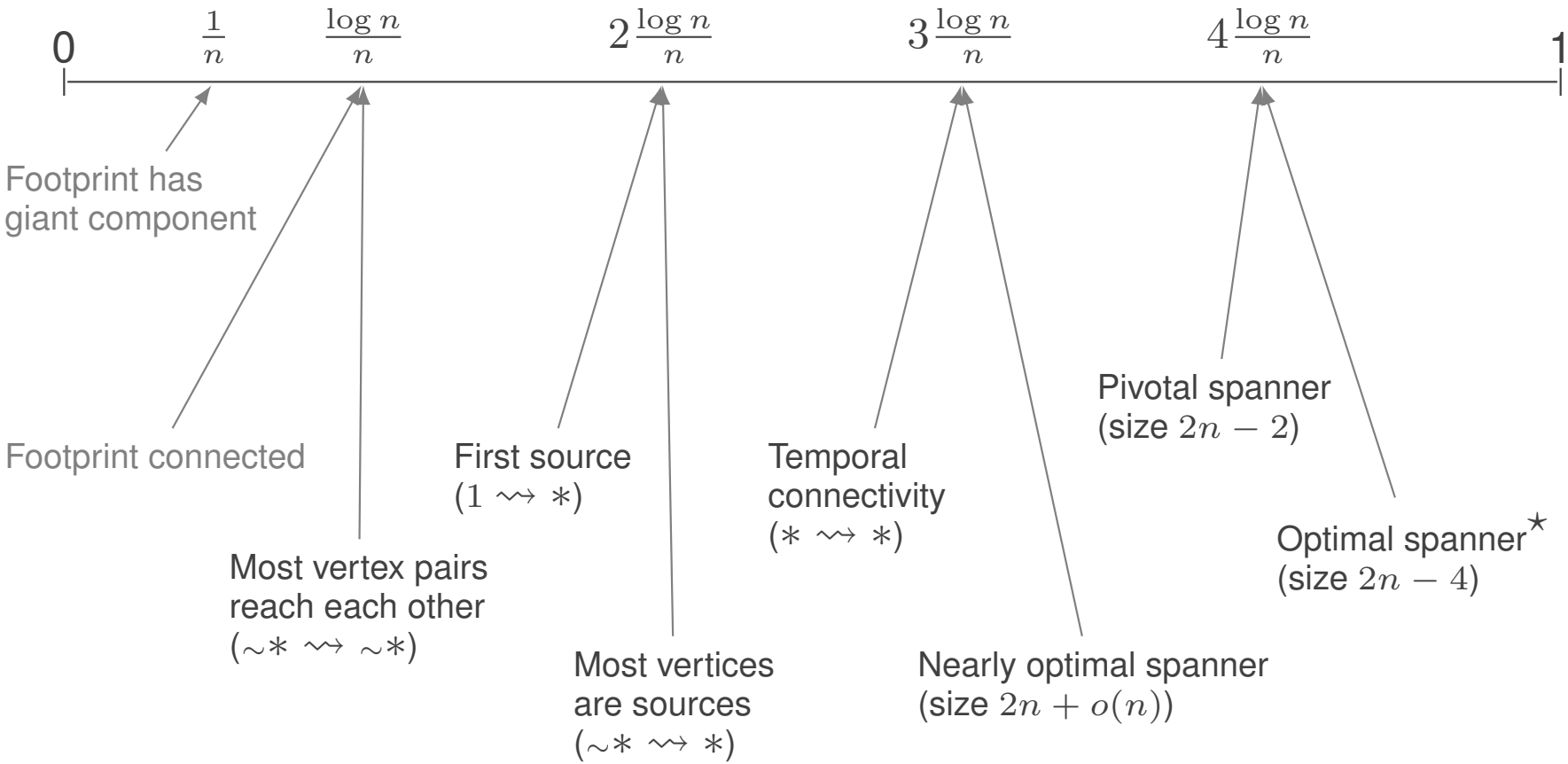
Timeline of temporal reachability in $\mathcal{G}_{n,p}$

For sufficiently large n, what happens when p increases?



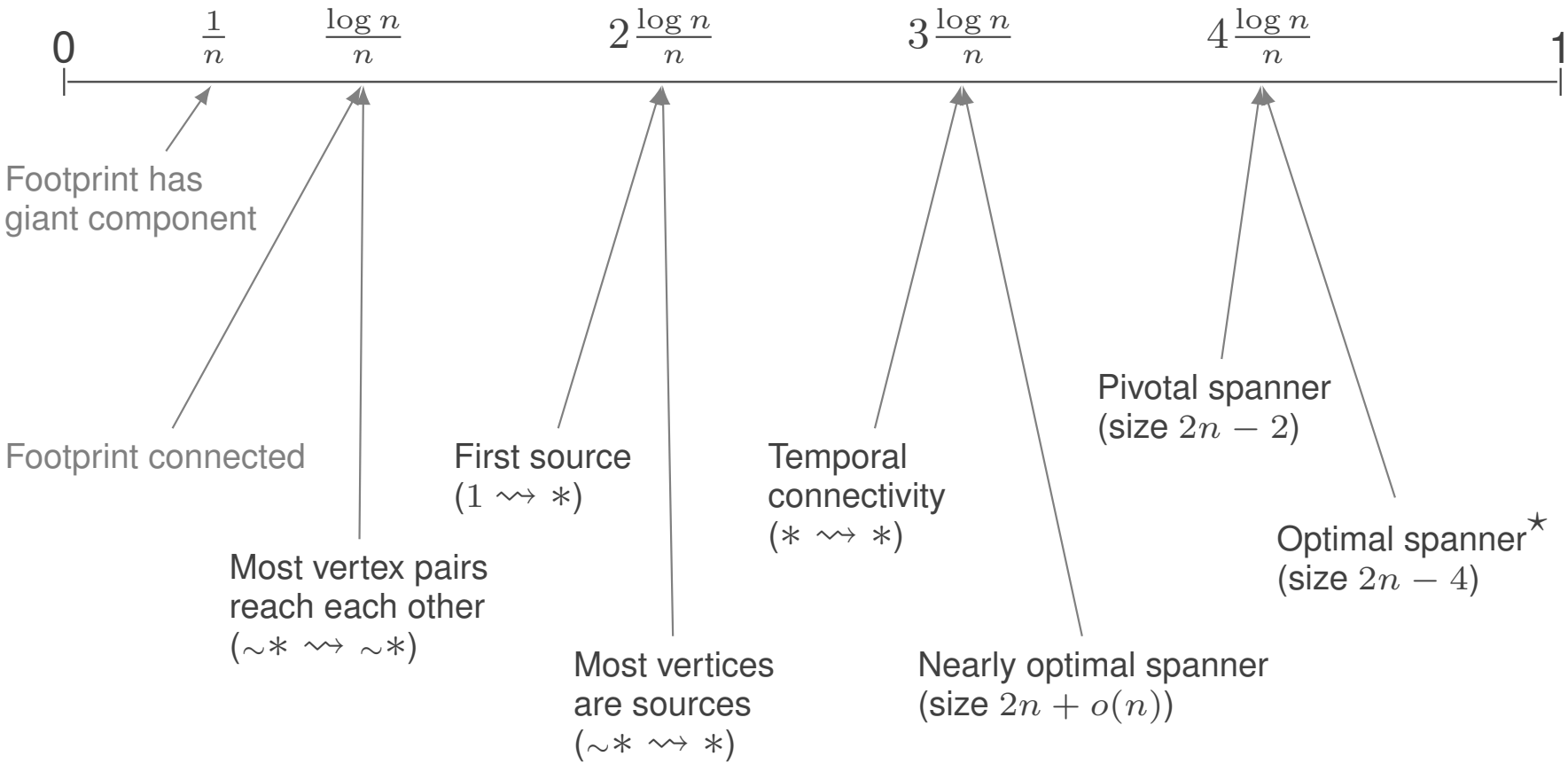
Timeline of temporal reachability in $\mathcal{G}_{n,p}$

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)

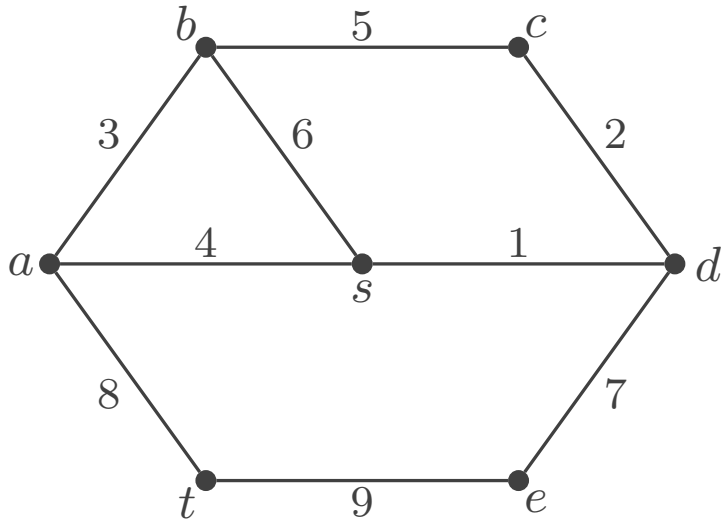
(sharp: $\exists \epsilon(n) = o(1)$, not true at $(1 - \epsilon(n))p$, true at $(1 + \epsilon(n))p$)

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



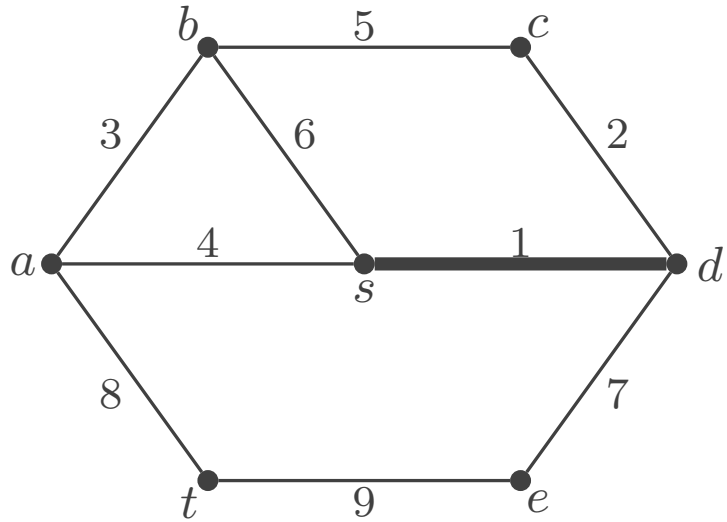
$t = 0$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



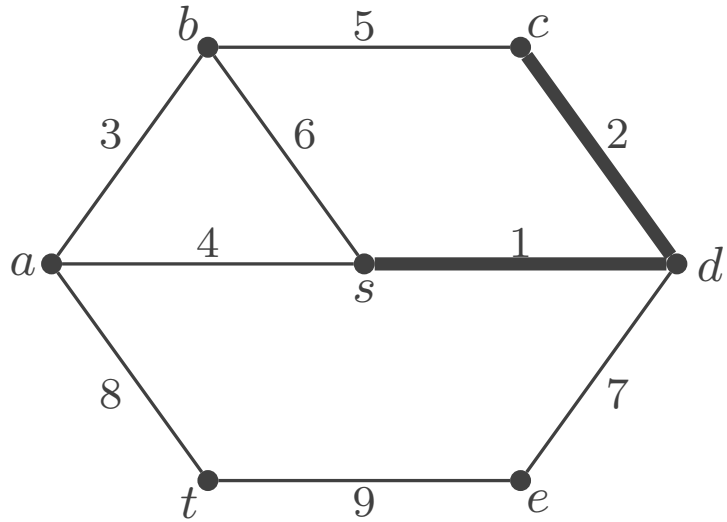
$t = 1$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



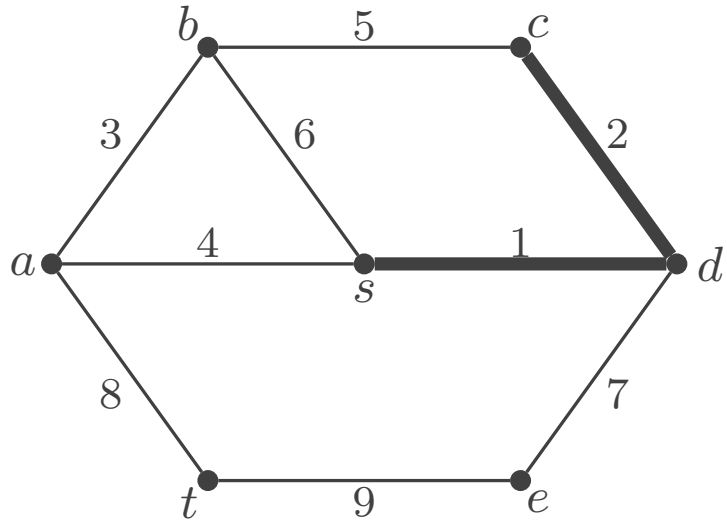
$t = 2$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



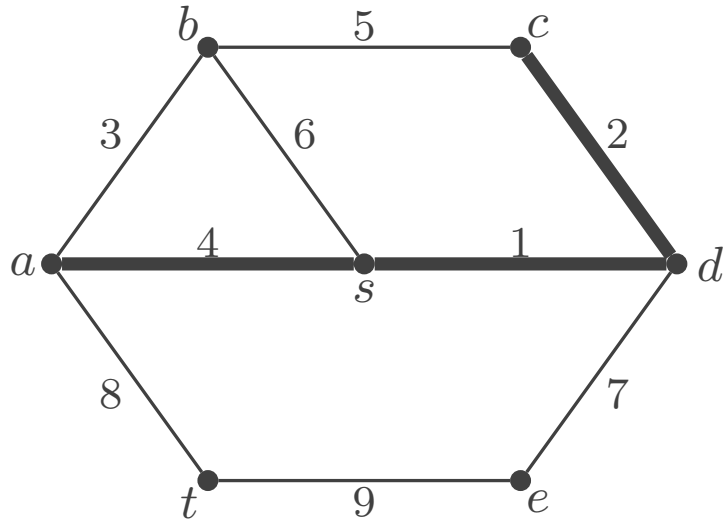
$t = 3$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



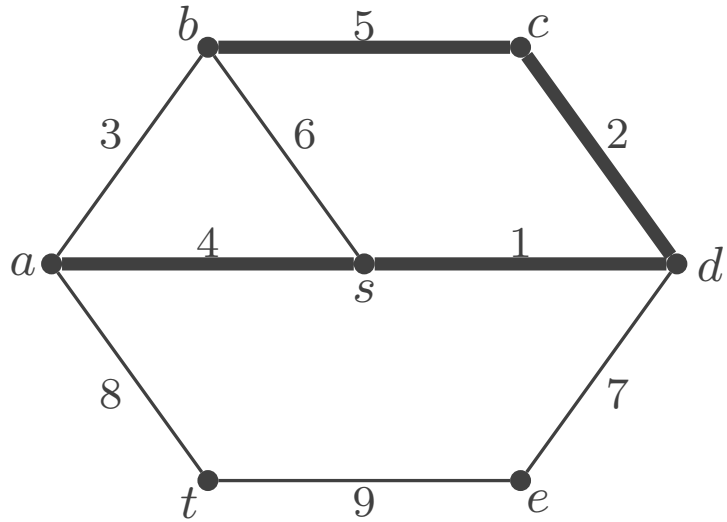
$t = 4$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- “Prim-like” algorithm.



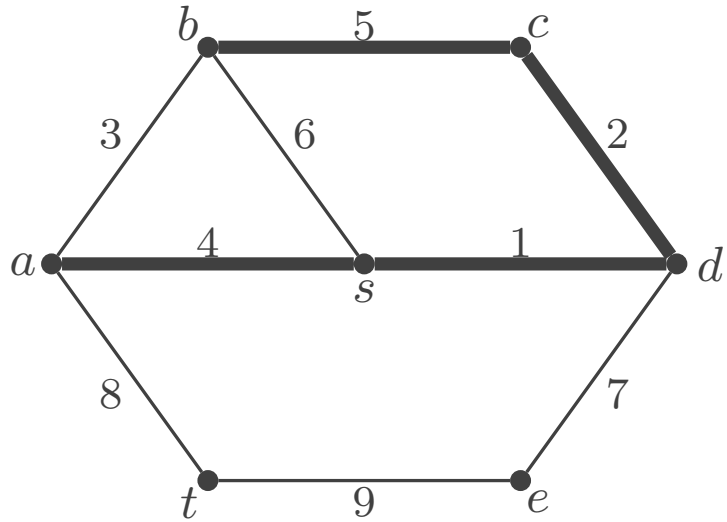
$t = 5$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- “Prim-like” algorithm.



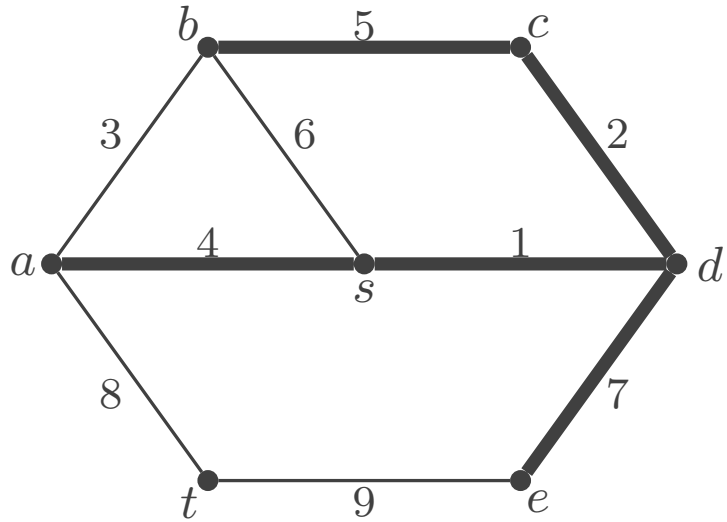
$t = 6$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



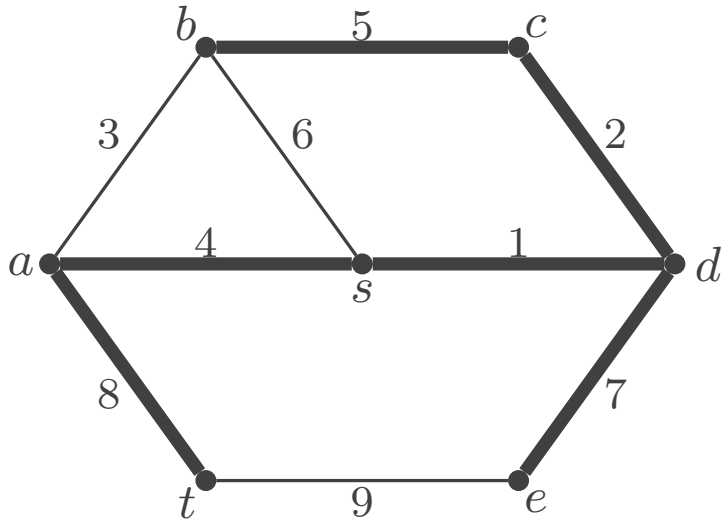
$t = 7$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



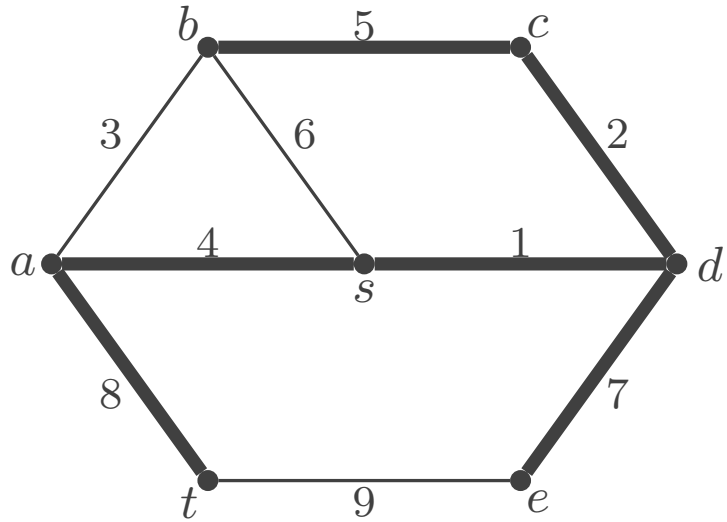
$t = 8$

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



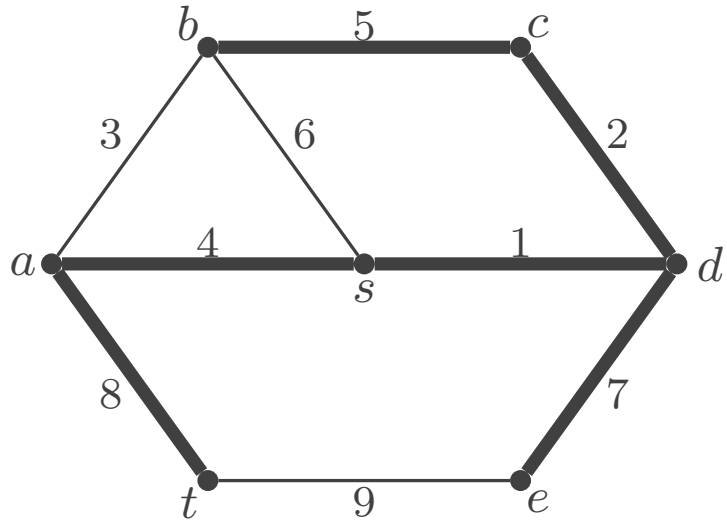
Analysis

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



Analysis

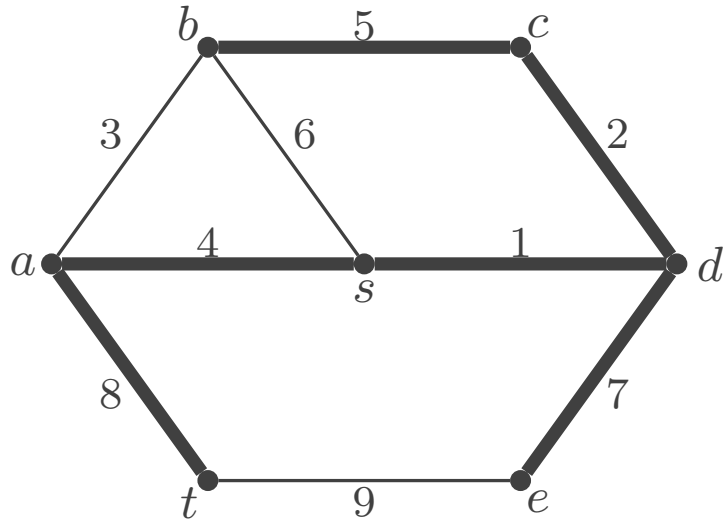
- Consider "growing" a foremost tree in $\mathcal{G} \sim \mathcal{G}_{n,1}$ from a vertex s.

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



Analysis

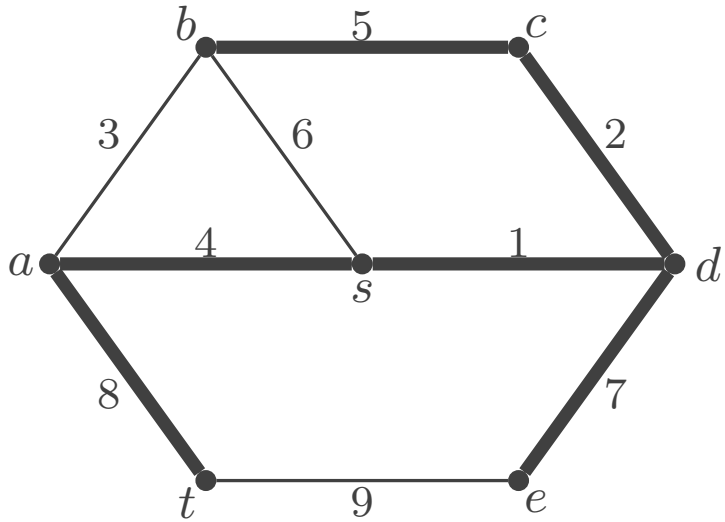
- Consider "growing" a foremost tree in $\mathcal{G} \sim \mathcal{G}_{n,1}$ from a vertex s.
- Once we have reached k vertices, there are $k(n-k)$ potential edges.

Main technical tool: growth of a foremost tree

Foremost tree (from s)

- Foremost temporal paths from s to all

- "Prim-like" algorithm.



Analysis

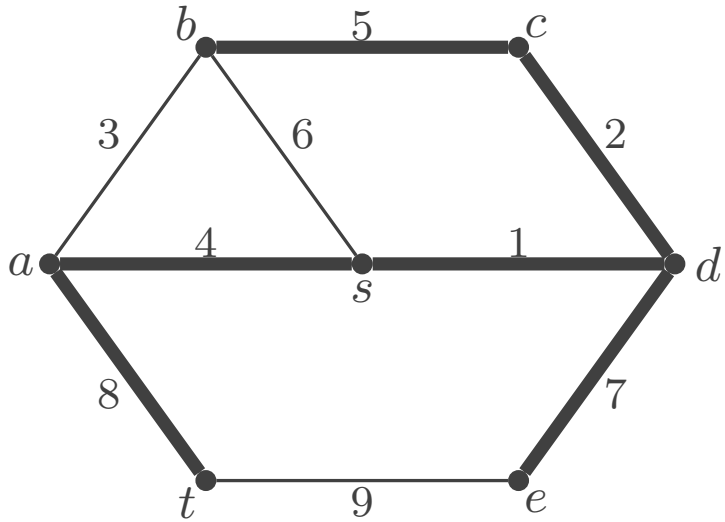
- Consider "growing" a foremost tree in $\mathcal{G} \sim \mathcal{G}_{n,1}$ from a vertex s.
- Once we have reached k vertices, there are $k(n-k)$ potential edges.
- The waiting time for one of these to appear is $\approx \frac{1}{k(n-k)}$

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 $\approx \frac{1}{k(n-k)}$

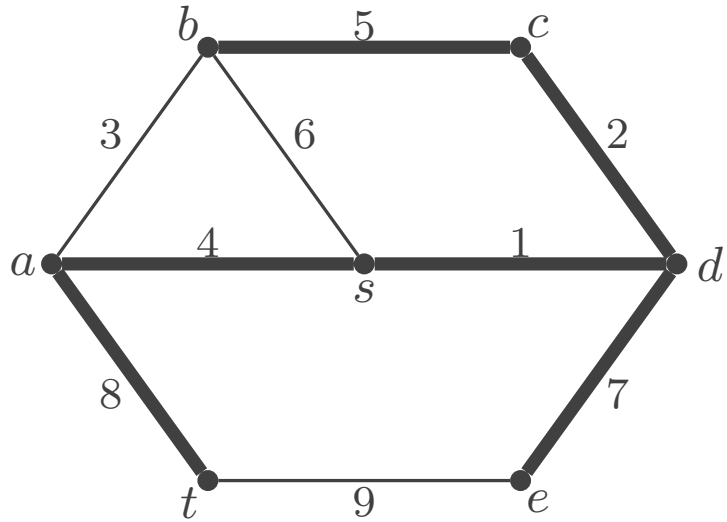
$\Longrightarrow$ Expect to reach all vertices at $\sum_{k=1}^{n} \frac{1}{k(n-k)} \approx 2\frac{\log n}{n}$.

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)

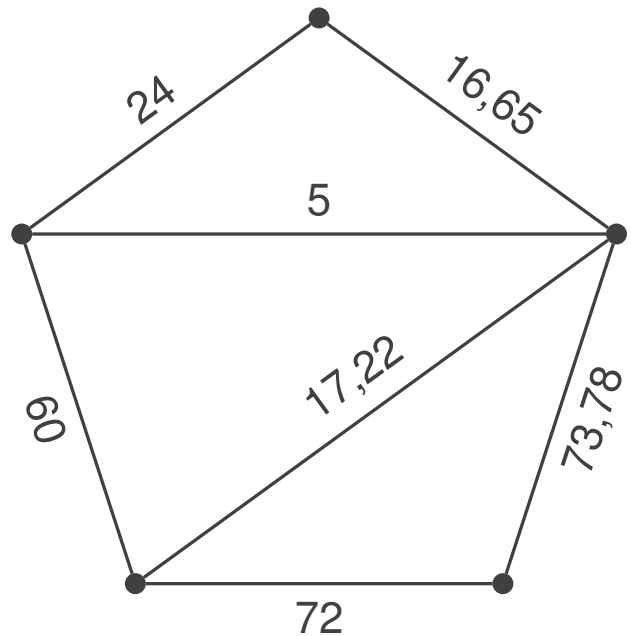
  UB: Explicit construction

  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 $\mathcal{H}_{n,p}$.

Open questions (deterministic)

Better spanners for temporal cliques

- Is $O(n \log n)$ optimal for cliques? Is $O(n)$ possible?

Open questions (deterministic)

Better spanners for temporal cliques

- Is $O(n \log n)$ optimal for cliques? Is $O(n)$ possible?

Experiments:

n	sparsest spanner (# edges)	
4	4 or 5	exhaustive search
5	6 or 7	exhaustive search
6	8 or 9	exhaustive search
7	10 or 11	exhaustive search
...	...	
20	36 or 37	millions random instances

$\rightarrow$ $2n - 4 \leq OPT \leq 2n - 3$?

Open questions (deterministic)

Better spanners for temporal cliques

- Is $O(n \log n)$ optimal for cliques? Is $O(n)$ possible?

Experiments:

n	sparsest spanner (# edges)	
4	4 or 5	exhaustive search
5	6 or 7	exhaustive search
6	8 or 9	exhaustive search
7	10 or 11	exhaustive search
...	...	
20	36 or 37	millions random instances

$\rightarrow$ $2n - 4 \leq OPT \leq 2n - 3$?

Relaxing the complete graph assumption

Open questions (deterministic)

Better spanners for temporal cliques

- Is $O(n \log n)$ optimal for cliques? Is $O(n)$ possible?

Experiments:

n	sparsest spanner (# edges)	
4	4 or 5	exhaustive search
5	6 or 7	exhaustive search
6	8 or 9	exhaustive search
7	10 or 11	exhaustive search
...	...	
20	36 or 37	millions random instances

$\rightarrow$ $2n - 4 \leq OPT \leq 2n - 3$?

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?

Experiments:

n	sparsest spanner (# edges)	
4	4 or 5	exhaustive search
5	6 or 7	exhaustive search
6	8 or 9	exhaustive search
7	10 or 11	exhaustive search
...	...	
20	36 or 37	millions random instances

$\rightarrow$ $2n - 4 \leq OPT \leq 2n - 3$?

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?

Thank you!