Data Structures for Temporal Queries on Dynamic Temporal Graphs*

Davide Bilò

Algorithmic Aspects of Temporal Graphs IX
Royal Holloway, University of London
United Kingdom
July 6th, 2026

* Based on a joint work with Luciano Gualà, Stefano Leucci, Guido Proietti, and Alessandro Straziota

Goal of this talk

- Design a compact **data structure** (a.k.a. **oracle**) that
  - Stores some information about a temporal graph
  - Can quickly answer to specific temporal path queries

Goal of this talk

- Design a compact **data structure** (a.k.a. **oracle**) that
  - Stores some information about a temporal graph
  - Can quickly answer to specific temporal path queries

- Two scenarios

Goal of this talk

- Design a compact **data structure** (a.k.a. **oracle**) that
  - Stores some information about a temporal graph
  - Can quickly answer to specific temporal path queries

- Two scenarios
  1. **Static**: the temporal graph does not change

Goal of this talk

- Design a compact **data structure** (a.k.a. **oracle**) that
  - Stores some information about a temporal graph
  - Can quickly answer to specific temporal path queries

- Two scenarios
  1. **Static**: the temporal graph does not change
  2. **Dynamic**: the temporal graph is subject to specific updates

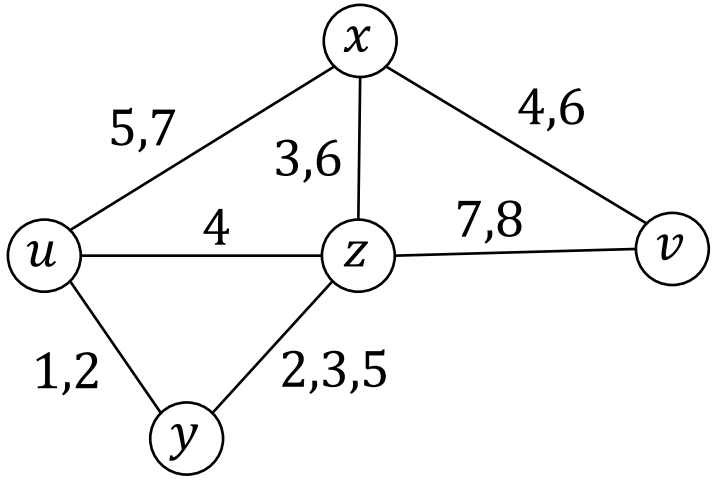
Goal of this talk

- Design a compact **data structure** (a.k.a. **oracle**) that
  - Stores some information about a temporal graph
  - Can quickly answer to specific temporal path queries

- Two scenarios
  1. **Static**: the temporal graph does not change
  2. **Dynamic**: the temporal graph is subject to specific updates

Design oracles with good tradeoffs among
size, query time, and update time (only if dynamic)

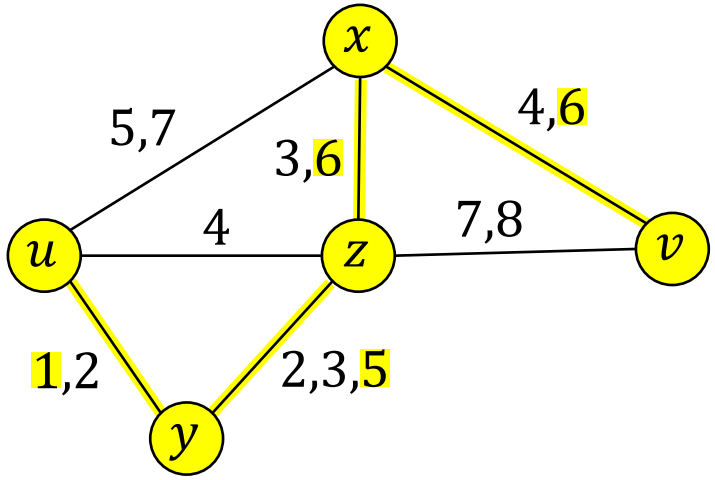
Temporal graph model



Temporal graph

- undirected graph
- each edge e has a set $\lambda(e)$ of integer time labels

Temporal graph model



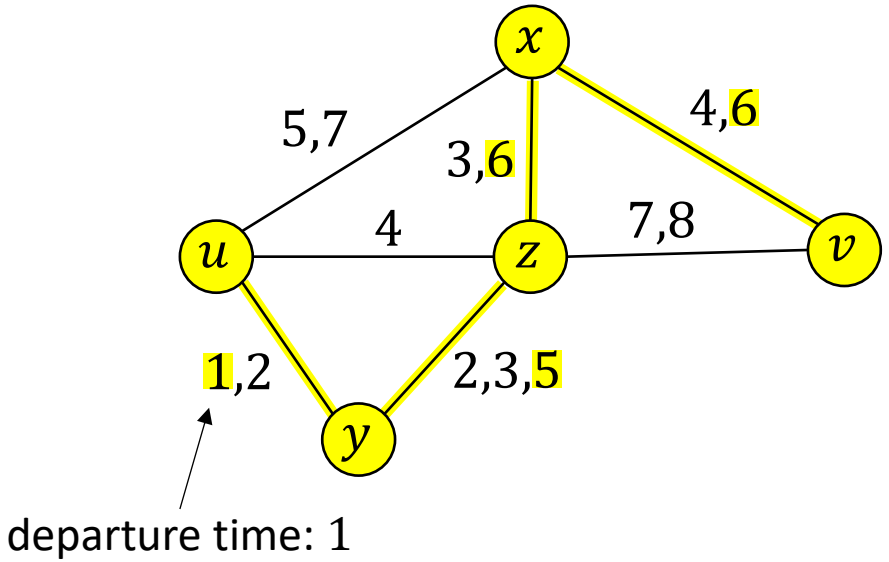
Temporal graph

- undirected graph
- each edge e has a set $\lambda(e)$ of integer time labels

non-strict temporal $u - v$ path

A $u - v$ path P, with a selected $t_e \in \lambda(e)$ for each edge e in P, s.t. the sequence of traversed t_e's is monotonically non-decreasing

Temporal graph model



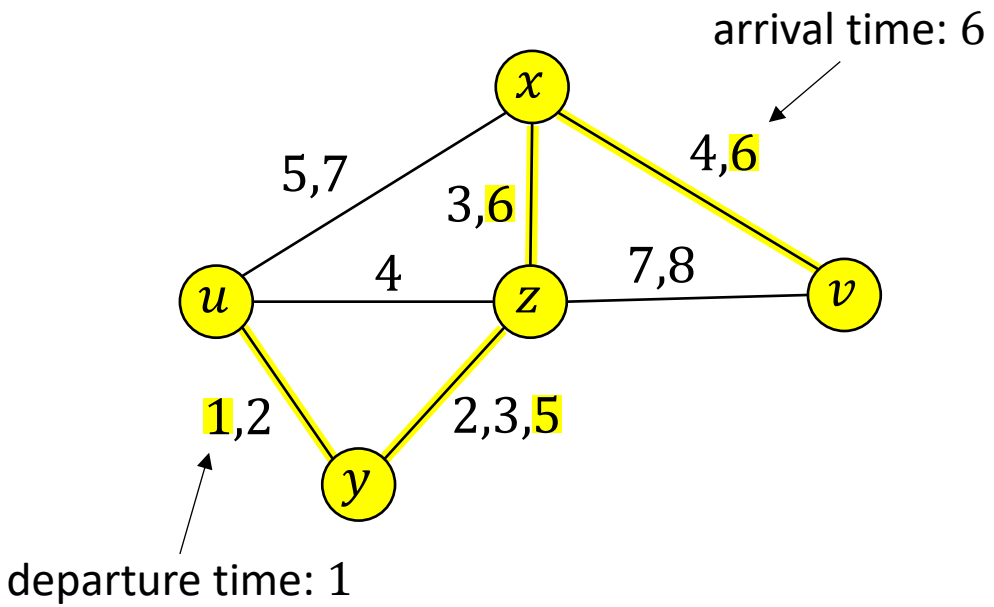
Temporal graph

- undirected graph
- each edge e has a set $\lambda(e)$ of integer time labels

non-strict temporal $u - v$ path

A $u - v$ path P, with a selected $t_e \in \lambda(e)$ for each edge e in P, s.t. the sequence of traversed t_e's is monotonically non-decreasing

Temporal graph model



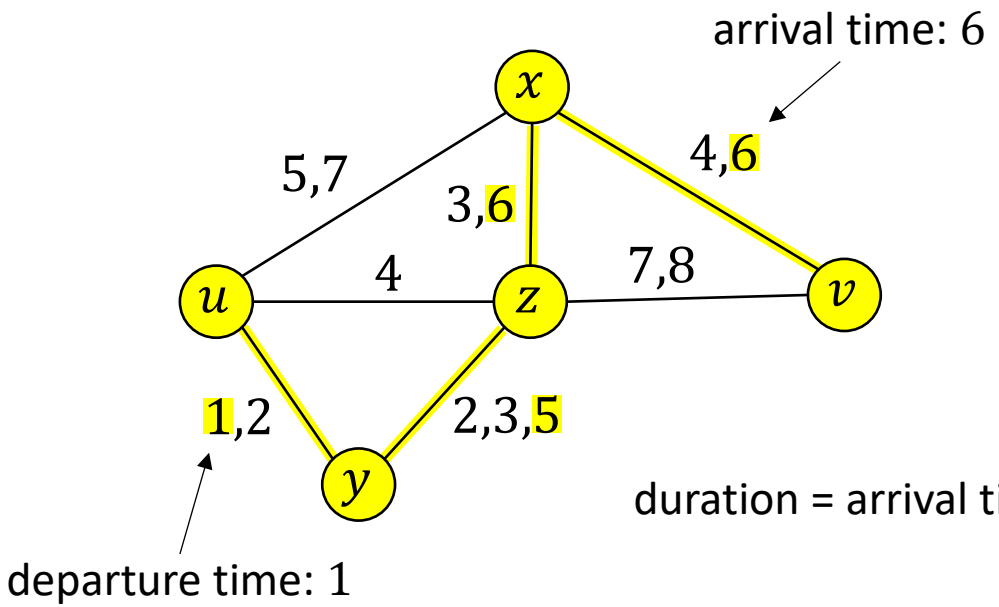
Temporal graph

- undirected graph
- each edge e has a set $\lambda(e)$ of integer time labels

non-strict temporal $u - v$ path

A $u - v$ path P, with a selected $t_e \in \lambda(e)$ for each edge e in P, s.t. the sequence of traversed t_e's is monotonically non-decreasing

Temporal graph model



Temporal graph

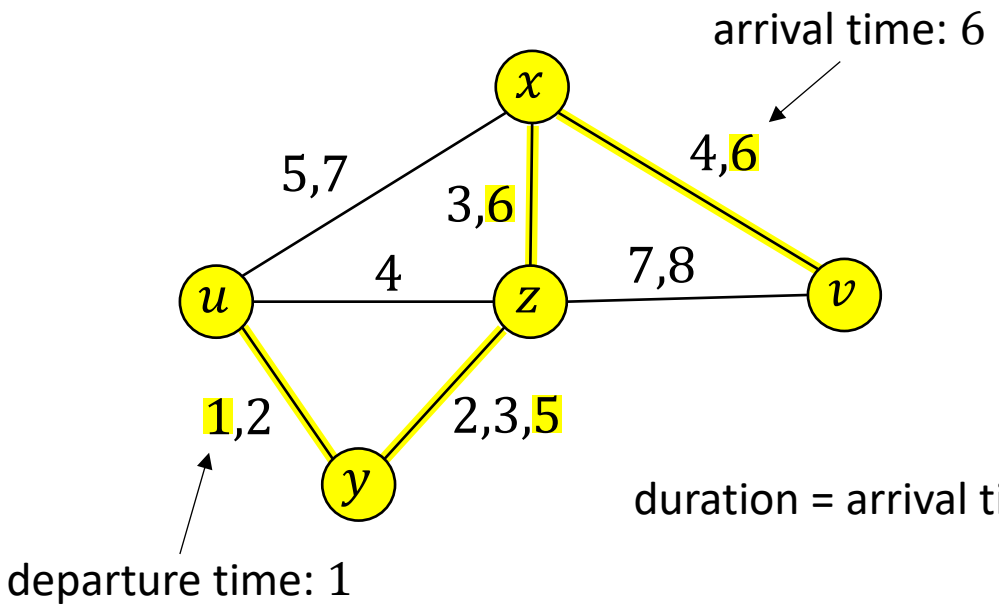
- undirected graph
- each edge e has a set $\lambda(e)$ of integer time labels

duration = arrival time – departure time = $6 - 1 = 5$

non-strict temporal $u - v$ path

A $u - v$ path P, with a selected $t_e \in \lambda(e)$ for each edge e in P, s.t. the sequence of traversed t_e's is monotonically non-decreasing

Temporal graph model



Temporal graph

- undirected graph
- each edge e has a set $\lambda(e)$ of integer time labels

duration = arrival time – departure time = $6 - 1 = 5$

non-strict temporal $u - v$ path

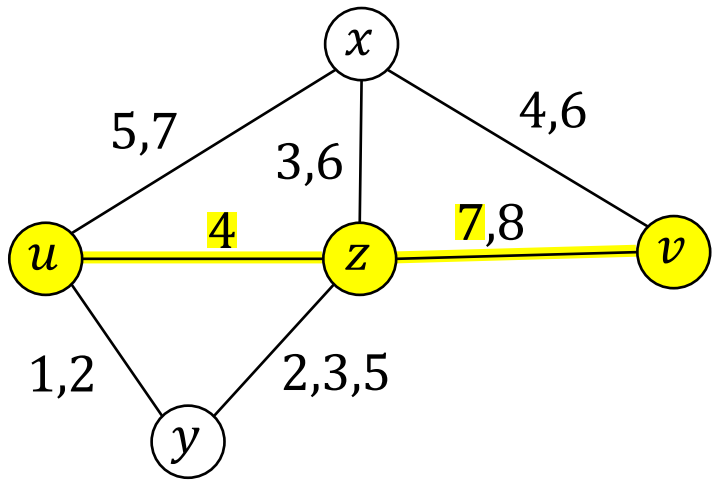
A $u - v$ path P, with a selected $t_e \in \lambda(e)$ for each edge e in P, s.t. the sequence of traversed t_e's is monotonically non-decreasing

All the results we will see can be adapted to strict temporal paths

Queries

Temporal Reachability $TR(u, v, [d, a])$

Existence of a temporal $u - v$ path restricted to the time interval $[d, a]$



$TR(u, v, [3,7]) = TRUE$

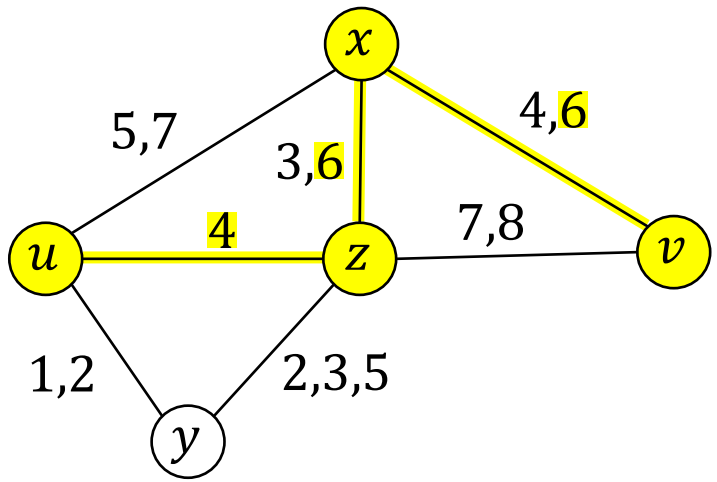
Queries

Temporal Reachability $TR(u, v, [d, a])$

Existence of a temporal $u - v$ path restricted to the time interval $[d, a]$

Earliest Arrival $EA(u, v, d)$

Minimum arrival time of a temporal $u - v$ path with departure time $\geq d$



$EA(u, v, 3) = 6$

Queries

Temporal Reachability $TR(u, v, [d, a])$

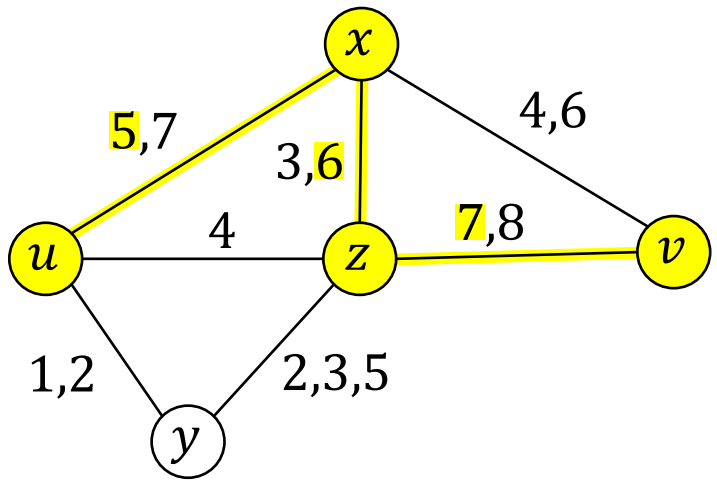
Existence of a temporal $u - v$ path restricted to the time interval $[d, a]$

Earliest Arrival $EA(u, v, d)$

Minimum arrival time of a temporal $u - v$ path with departure time $\geq d$

Latest Departure $LD(u, v, a)$

Maximum departure time of a temporal $u - v$ path with arrival time $\leq a$



$LD(u, v, 7) = 5$

Queries

Temporal Reachability $TR(u, v, [d, a])$

Existence of a temporal $u - v$ path restricted to the time interval $[d, a]$

Earliest Arrival $EA(u, v, d)$

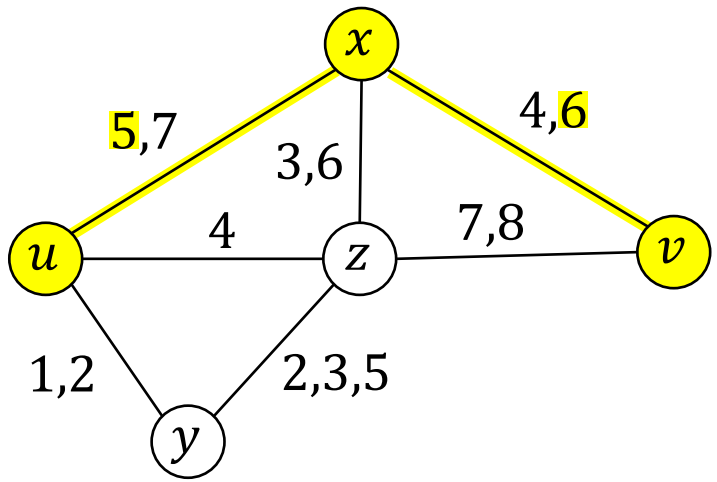
Minimum arrival time of a temporal $u - v$ path with departure time $\geq d$

Latest Departure $LD(u, v, a)$

Maximum departure time of a temporal $u - v$ path with arrival time $\leq a$

Min Duration $MD(u, v, [d, a])$

Minimum duration of a temporal $u - v$ path restricted to the time interval $[d, a]$



$MD(u, v, [3,7]) = 1$

Useful observations

1. TR queries are no more difficult than EA/LD/MD queries

Useful observations

1. TR queries are no more difficult than EA/LD/MD queries
   - $TR(u, v, [d, a]) \equiv (EA(u, v, d) \leq a)$

Useful observations

1. TR queries are no more difficult than EA/LD/MD queries
   - $TR(u, v, [d, a]) \equiv (EA(u, v, d) \leq a)$
   - $TR(u, v, [d, a]) \equiv (LD(u, v, a) \geq d)$

Useful observations

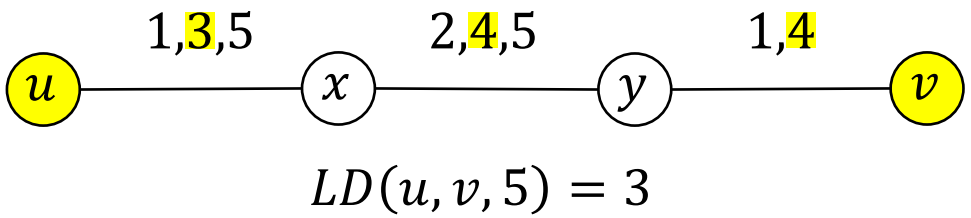
1. TR queries are no more difficult than EA/LD/MD queries
   - $TR(u, v, [d, a]) \equiv (EA(u, v, d) \leq a)$
   - $TR(u, v, [d, a]) \equiv (LD(u, v, a) \geq d)$
   - $TR(u, v, [d, a]) \equiv (MD(u, v, [d, a]) \leq d - a)$

Useful observations

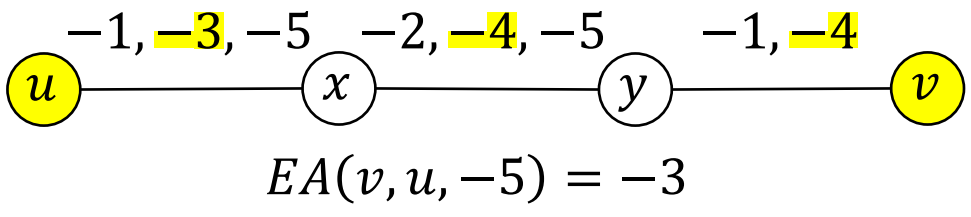
1. TR queries are no more difficult than EA/LD/MD queries
   - $TR(u, v, [d, a]) \equiv (EA(u, v, d) \leq a)$
   - $TR(u, v, [d, a]) \equiv (LD(u, v, a) \geq d)$
   - $TR(u, v, [d, a]) \equiv (MD(u, v, [d, a]) \leq d - a)$
2. EA queries $\equiv$ LD queries

Useful observations

1. TR queries are no more difficult than EA/LD/MD queries
   - $TR(u, v, [d, a]) \equiv (EA(u, v, d) \leq a)$
   - $TR(u, v, [d, a]) \equiv (LD(u, v, a) \geq d)$
   - $TR(u, v, [d, a]) \equiv (MD(u, v, [d, a]) \leq d - a)$
2. EA queries ≡ LD queries



$LD(u, v, 5) = 3$



$EA(v, u, -5) = -3$

State of the art prior to our work

Surprisingly, we found only a few papers published on the topic

State of the art prior to our work

Surprisingly, we found only a few papers published on the topic

- **Brito, Albertini, Casteigts, and Travençolo, Social Network Analysis and Mining 2021**: Oracle for DYNAMIC temporal directed graphs

State of the art prior to our work

Surprisingly, we found only a few papers published on the topic

- **Brito, Albertini, Casteigts, and Travençolo, Social Network Analysis and Mining 2021**: Oracle for DYNAMIC temporal directed graphs
  - Maintain the *Timed Transitive Closure*, i.e., non-dominated 4-tuples (u, v, d, a) meaning *«There is a $u - v$ temporal path with departure time d and arrival time a»*

State of the art prior to our work

Surprisingly, we found only a few papers published on the topic

- **Brito, Albertini, Casteigts, and Travençolo, Social Network Analysis and Mining 2021**: Oracle for DYNAMIC temporal directed graphs
  - Maintain the *Timed Transitive Closure*, i.e., non-dominated 4-tuples (u, v, d, a) meaning *«There is a $u - v$ temporal path with departure time d and arrival time a»*
  - Size is $O(n^2\tau)$

- $n =$ # vertices
- $\tau = |\bigcup_e \lambda(e)|$ is *lifetime*

State of the art prior to our work

Surprisingly, we found only a few papers published on the topic

- **Brito, Albertini, Casteigts, and Travençolo, Social Network Analysis and Mining 2021**: Oracle for DYNAMIC temporal directed graphs
  - Maintain the *Timed Transitive Closure*, i.e., non-dominated 4-tuples (u, v, d, a) meaning *«There is a $u - v$ temporal path with departure time d and arrival time a»*
  - Size is $O(n^2\tau)$
  - TR queries in $O(\log \tau)$ time

- $n =$ # vertices
- $\tau = |\bigcup_e \lambda(e)|$ is *lifetime*

State of the art prior to our work

Surprisingly, we found only a few papers published on the topic

- **Brito, Albertini, Casteigts, and Travençolo, Social Network Analysis and Mining 2021**: Oracle for DYNAMIC temporal directed graphs
  - Maintain the *Timed Transitive Closure*, i.e., non-dominated 4-tuples (u, v, d, a) meaning *«There is a $u - v$ temporal path with departure time d and arrival time a»*
  - Size is $O(n^2\tau)$
  - TR queries in $O(\log \tau)$ time
  - Time-label insertions in $O(n^2 \log \tau)$ time amortized

- $n =$ # vertices
- $\tau = |\bigcup_e \lambda(e)|$ is *lifetime*

State of the art prior to our work

Surprisingly, we found only a few papers published on the topic

- **Brito, Albertini, Casteigts, and Travençolo, Social Network Analysis and Mining 2021**: Oracle for DYNAMIC temporal directed graphs
  - Maintain the *Timed Transitive Closure*, i.e., non-dominated 4-tuples (u, v, d, a) meaning *«There is a $u - v$ temporal path with departure time d and arrival time a»*
  - Size is $O(n^2\tau)$
  - TR queries in $O(\log \tau)$ time
  - Time-label insertions in $O(n^2 \log \tau)$ time amortized

- $n =$ # vertices
- $\tau = |\bigcup_e \lambda(e)|$ is *lifetime*

Results hold for uniform *latencies,* time labels are pairs (t, ℓ), where $\ell \geq 0$ is the latency

State of the art prior to our work

Surprisingly, we found only a few papers published on the topic

- **Brito, Albertini, Casteigts, and Travençolo, Social Network Analysis and Mining 2021**: Oracle for DYNAMIC temporal directed graphs
  - Maintain the *Timed Transitive Closure*, i.e., non-dominated 4-tuples (u, v, d, a) meaning *«There is a $u - v$ temporal path with departure time d and arrival time a»*
  - Size is $O(n^2\tau)$
  - TR queries in $O(\log \tau)$ time
  - Time-label insertions in $O(n^2 \log \tau)$ time amortized

- $n =$ # vertices
- $\tau = |\bigcup_e \lambda(e)|$ is *lifetime*

Results hold for uniform *latencies,* time labels are pairs (t, ℓ), where $\ell \geq 0$ is the latency

- **Brito, Albertini, and Travençolo, arXiv 2021**: oracle optimized for external memories

State of the art prior to our work

Surprisingly, we found only a few papers published on the topic

- **Brito, Albertini, Casteigts, and Travençolo, Social Network Analysis and Mining 2021**: Oracle for DYNAMIC temporal directed graphs
  - Maintain the *Timed Transitive Closure*, i.e., non-dominated 4-tuples (u, v, d, a) meaning *«There is a $u - v$ temporal path with departure time d and arrival time a»*
  - Size is $O(n^2\tau)$
  - TR queries in $O(\log \tau)$ time
  - Time-label insertions in $O(n^2 \log \tau)$ time amortized

- $n =$ # vertices
- $\tau = |\bigcup_e \lambda(e)|$ is *lifetime*

Results hold for uniform *latencies,* time labels are pairs (t, ℓ), where $\ell \geq 0$ is the latency

- **Brito, Albertini, and Travençolo, arXiv 2021**: oracle optimized for external memories
- **Brito, Albertini, Travençolo, and Navarro, The Computer Journal 2024**: compact representation in practice

Our contribution

[1] B., Gualà, Leucci, Proietti, and Straziota, ALGOWIN 2025

[2] B., Gualà, Leucci, Proietti, and Straziota, ISAAC 2024

- $M = \sum_e |\lambda(e)|$
- $L = \max_e |\lambda(e)|$

	Size	**Time of supported queries**			**Time of supported updates (both insertions and deletions)**		
		TR	**EA/LD**	**MD**	**Time-labels**	**Singletons**	**Edge**
(STATIC) Trees [1]	Choose $t > 0$ $O(M + M^2/t^2)$	NO	NO	$O(t \log(1 + L))$	NO	NO	NO

Our contribution

[1] B., Gualà, Leucci, Proietti, and Straziota, ALGOWIN 2025

[2] B., Gualà, Leucci, Proietti, and Straziota, ISAAC 2024

- $M = \sum_e |\lambda(e)|$
- $L = \max_e |\lambda(e)|$

	Size	**Time of supported queries**			**Time of supported updates (both insertions and deletions)**		
		TR	**EA/LD**	**MD**	**Time-labels**	**Singletons**	**Edge**
(STATIC) Trees [1]	Choose $t > 0$ $O(M + M^2/t^2)$	NO	NO	$O(t \log(1 + L))$	NO	NO	NO

- Tradeoffs in [1] are asymptotically tight, up to polylog factors, under the STRONG SET DISJOINTNESS CONJECTURE

Our contribution

[1] B., Gualà, Leucci, Proietti, and Straziota, ALGOWIN 2025

[2] B., Gualà, Leucci, Proietti, and Straziota, ISAAC 2024

- $M = \sum_e |\lambda(e)|$
- $L = \max_e |\lambda(e)|$

	Size	**Time of supported queries**			**Time of supported updates (both insertions and deletions)**		
		TR	**EA/LD**	**MD**	**Time-labels**	**Singletons**	**Edge**
(STATIC) Trees [1]	Choose $t > 0$ $O(M + M^2/t^2)$	NO	NO	$O(t \log(1 + L))$	NO	NO	NO
Paths [2]	$O(n + M)$	$O(\log M)$	$O(\log M)$	NO	$O(\log M)$	NO	NO

- Tradeoffs in [1] are asymptotically tight, up to polylog factors, under the STRONG SET DISJOINTNESS CONJECTURE

Our contribution

[1] B., Gualà, Leucci, Proietti, and Straziota, ALGOWIN 2025

[2] B., Gualà, Leucci, Proietti, and Straziota, ISAAC 2024

- $M = \sum_e |\lambda(e)|$
- $L = \max_e |\lambda(e)|$

	Size	Time of supported queries			Time of supported updates (both insertions and deletions)		
		TR	EA/LD	MD	Time-labels	Singletons	Edge
(STATIC) Trees [1]	Choose $t > 0$ $O(M + M^2/t^2)$	NO	NO	$O(t \log(1 + L))$	NO	NO	NO
Paths [2]	$O(n + M)$	$O(\log M)$	$O(\log M)$	NO	$O(\log M)$	NO	NO
Forests of rooted trees [2]	$O(n + M)$	$O(\log M)$	$O(\log L \cdot \log M)$	NO	$O(\log M)$	$O(\log M)$	$O(\log M)$

- Tradeoffs in [1] are asymptotically tight, up to polylog factors, under the STRONG SET DISJOINTNESS CONJECTURE

Our contribution

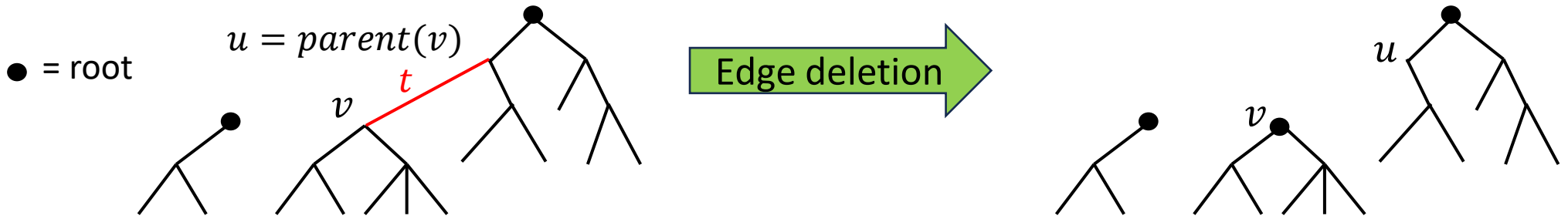
[1] B., Gualà, Leucci, Proietti, and Straziota, ALGOWIN 2025

[2] B., Gualà, Leucci, Proietti, and Straziota, ISAAC 2024

- $M = \sum_e |\lambda(e)|$
- $L = \max_e |\lambda(e)|$

	Size	Time of supported queries			Time of supported updates (both insertions and deletions)		
		TR	EA/LD	MD	Time-labels	Singletons	Edge
(STATIC) Trees [1]	Choose $t > 0$ $O(M + M^2/t^2)$	NO	NO	$O(t \log(1 + L))$	NO	NO	NO
Paths [2]	$O(n + M)$	$O(\log M)$	$O(\log M)$	NO	$O(\log M)$	NO	NO
Forests of rooted trees [2]	$O(n + M)$	$O(\log M)$	$O(\log L \cdot \log M)$	NO	$O(\log M)$	$O(\log M)$	$O(\log M)$

- Tradeoffs in [1] are asymptotically tight, up to polylog factors, under the STRONG SET DISJOINTNESS CONJECTURE
- Edge updates in a forest of rooted trees
  - Edge deletion: delete an edge $(parent(v), v)$ having only one time label (v is the root of the new formed tree)



Our contribution

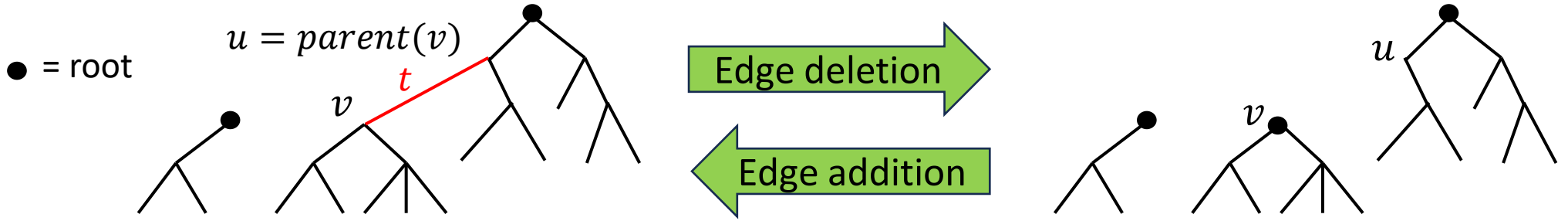
[1] B., Gualà, Leucci, Proietti, and Straziota, ALGOWIN 2025

[2] B., Gualà, Leucci, Proietti, and Straziota, ISAAC 2024

- $M = \sum_e |\lambda(e)|$
- $L = \max_e |\lambda(e)|$

	Size	Time of supported queries			Time of supported updates (both insertions and deletions)		
		TR	EA/LD	MD	Time-labels	Singletons	Edge
(STATIC) Trees [1]	Choose $t > 0$ $O(M + M^2/t^2)$	NO	NO	$O(t \log(1 + L))$	NO	NO	NO
Paths [2]	$O(n + M)$	$O(\log M)$	$O(\log M)$	NO	$O(\log M)$	NO	NO
Forests of rooted trees [2]	$O(n + M)$	$O(\log M)$	$O(\log L \cdot \log M)$	NO	$O(\log M)$	$O(\log M)$	$O(\log M)$

- Tradeoffs in [1] are asymptotically tight, up to polylog factors, under the STRONG SET DISJOINTNESS CONJECTURE
- Edge updates in a forest of rooted trees
  - Edge deletion: delete an edge $(parent(v), v)$ having only one time label (v is the root of the new formed tree)
  - Edge addition: add edge (u, v) with time label t (v must be the root of the absorbed tree)



Our contribution

[1] B., Gualà, Leucci, Proietti, and Straziota, ALGOWIN 2025

[2] B., Gualà, Leucci, Proietti, and Straziota, ISAAC 2024

- $M = \sum_e |\lambda(e)|$
- $L = \max_e |\lambda(e)|$

	Size	**Time of supported queries**			**Time of supported updates (both insertions and deletions)**		
		TR	**EA/LD**	**MD**	**Time-labels**	**Singletons**	**Edge**
(STATIC) Trees [1]	Choose $t > 0$ $O(M + M^2/t^2)$	NO	NO	$O(t \log(1 + L))$	NO	NO	NO
Paths [2]	$O(n + M)$	$O(\log M)$	$O(\log M)$	NO	$O(\log M)$	NO	NO
Forests of rooted trees [2]	$O(n + M)$	$O(\log M)$	$O(\log L \cdot \log M)$	NO	$O(\log M)$	$O(\log M)$	$O(\log M)$

- Tradeoffs in [1] are asymptotically tight, up to polylog factors, under the STRONG SET DISJOINTNESS CONJECTURE
- Edge updates in a forest of rooted trees
  - Edge deletion: delete an edge $(parent(v), v)$ having only one time label (v is the root of the new formed tree)
  - Edge addition: add edge (u, v) with time label t (v must be the root of the absorbed tree)
- If we add latencies to forests of rooted trees

Our contribution

[1] B., Gualà, Leucci, Proietti, and Straziota, ALGOWIN 2025

[2] B., Gualà, Leucci, Proietti, and Straziota, ISAAC 2024

- $M = \sum_e |\lambda(e)|$
- $L = \max_e |\lambda(e)|$

	Size	**Time of supported queries**			**Time of supported updates (both insertions and deletions)**		
		TR	**EA/LD**	**MD**	**Time-labels**	**Singletons**	**Edge**
(STATIC) Trees [1]	Choose $t > 0$ $O(M + M^2/t^2)$	NO	NO	$O(t \log(1 + L))$	NO	NO	NO
Paths [2]	$O(n + M)$	$O(\log M)$	$O(\log M)$	NO	$O(\log M)$	NO	NO
Forests of rooted trees [2]	$O(n + M)$	$O(\log M)$	$O(\log L \cdot \log M)$	NO	$O(\log M)$	$O(\log M)$	$O(\log M)$

- Tradeoffs in [1] are asymptotically tight, up to polylog factors, under the STRONG SET DISJOINTNESS CONJECTURE
- Edge updates in a forest of rooted trees
  - Edge deletion: delete an edge $(parent(v), v)$ having only one time label (v is the root of the new formed tree)
  - Edge addition: add edge (u, v) with time label t (v must be the root of the absorbed tree)
- If we add latencies to forests of rooted trees
  - Uniform latencies: same trade-offs

Our contribution

[1] B., Gualà, Leucci, Proietti, and Straziota, ALGOWIN 2025

[2] B., Gualà, Leucci, Proietti, and Straziota, ISAAC 2024

- $M = \sum_e |\lambda(e)|$
- $L = \max_e |\lambda(e)|$

	Size	Time of supported queries			Time of supported updates (both insertions and deletions)		
		TR	EA/LD	MD	Time-labels	Singletons	Edge
(STATIC) Trees [1]	Choose $t > 0$ $O(M + M^2/t^2)$	NO	NO	$O(t \log(1 + L))$	NO	NO	NO
Paths [2]	$O(n + M)$	$O(\log M)$	$O(\log M)$	NO	$O(\log M)$	NO	NO
Forests of rooted trees [2]	$O(n + M)$	$O(\log M)$	$O(\log L \cdot \log M)$	NO	$O(\log M)$	$O(\log M)$	$O(\log M)$

- Tradeoffs in [1] are asymptotically tight, up to polylog factors, under the STRONG SET DISJOINTNESS CONJECTURE
- Edge updates in a forest of rooted trees
  - Edge deletion: delete an edge $(parent(v), v)$ having only one time label (v is the root of the new formed tree)
  - Edge addition: add edge (u, v) with time label t (v must be the root of the absorbed tree)
- If we add latencies to forests of rooted trees
  - Uniform latencies: same trade-offs
  - Non-uniform latencies
    - Incremental and decremental scenarios (i.e., only insertions or deletions)
    - same trade-offs, but update time bounds become amortized

Results presented in this talk

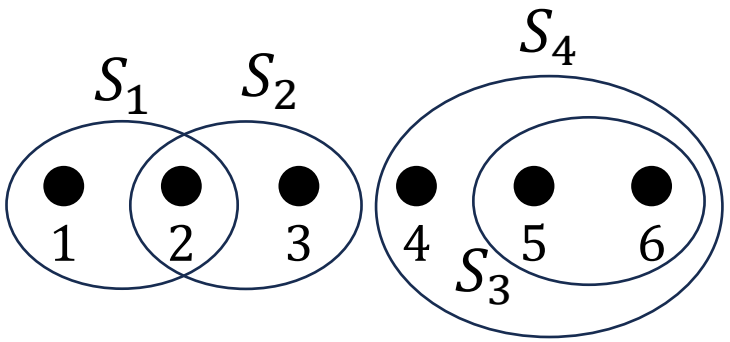
- $M = \sum_e |\lambda(e)|$
- $L = \max_e |\lambda(e)|$

	Size	Time of supported queries			Time of supported updates (both insertions and deletions)		
		TR	EA/LD	MD	Time-labels	Singletons	Edge
(STATIC) Trees [1]	Choose $t > 0$ $O(M + M^2/t^2)$	NO	NO	$O(t \log(1 + L))$	NO	NO	NO
Paths [2]	$O(n + M)$	$O(\log M)$	$O(\log M)$	NO	$O(\log M)$	NO	NO
Forests of rooted trees [2]	$O(n + M)$	$O(\log M)$	$O(\log L \cdot \log M)$	NO	$O(\log M)$	$O(\log M)$	$O(\log M)$

- Tradeoffs in [1] are asymptotically tight, up to polylog factors, under the STRONG SET DISJOINTNESS CONJECTURE

The conditional LB of oracles supporting MD queries

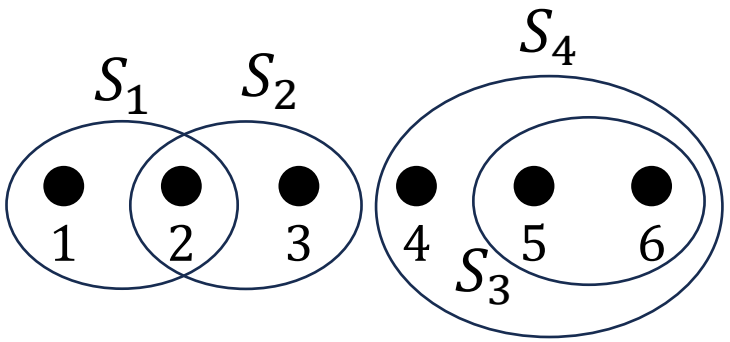
SET DISJOINTNESS PROBLEM: Given k subsets $S_1, \dots, S_k$ of a universe U with $N = \sum_i |S_i|$, design an oracle that can answer to queries *«Given i and j, are S_i and S_j disjoint?»*.



The conditional LB of oracles supporting MD queries

SET DISJOINTNESS PROBLEM: Given k subsets $S_1, \dots, S_k$ of a universe U with $N = \sum_i |S_i|$, design an oracle that can answer to queries *«Given i and j, are S_i and S_j disjoint?».*

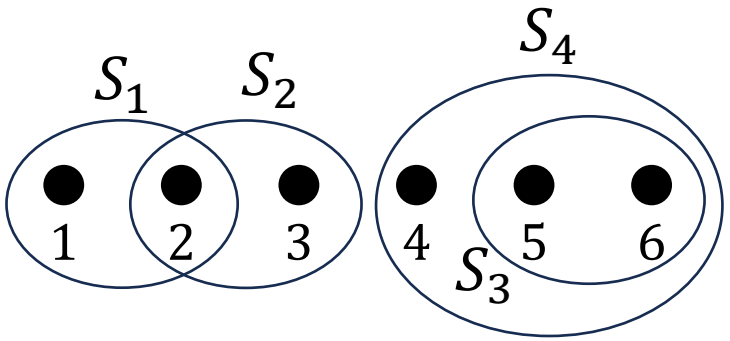
STRONG SET DISJOINTNESS CONJECTURE [Goldstein, Lewenstein, Porat, ISAAC 2019]:
Any oracle that can answer to a query in worst-case time t requires $\widetilde{\Omega}(N^2/t^2)$ space.

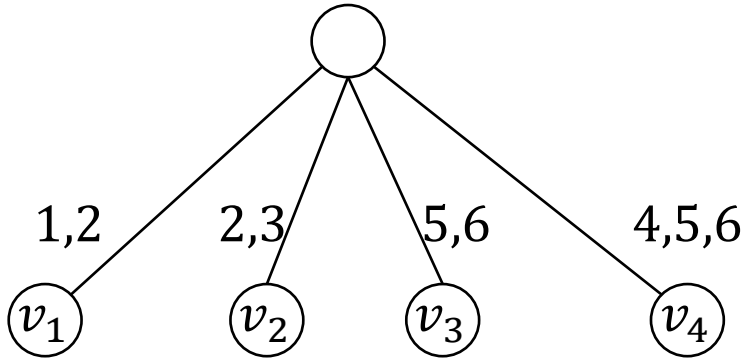


The conditional LB of oracles supporting MD queries

SET DISJOINTNESS PROBLEM: Given k subsets $S_1, \dots, S_k$ of a universe U with $N = \sum_i |S_i|$, design an oracle that can answer to queries *«Given i and j, are S_i and S_j disjoint?»*.

STRONG SET DISJOINTNESS CONJECTURE [Goldstein, Lewenstein, Porat, ISAAC 2019]:
Any oracle that can answer to a query in worst-case time t requires $\widetilde{\Omega}(N^2/t^2)$ space.

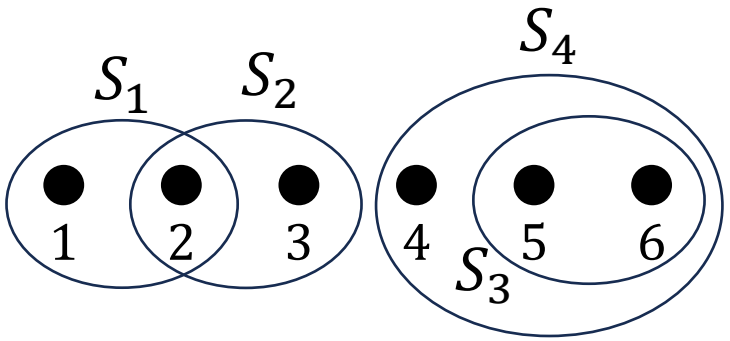


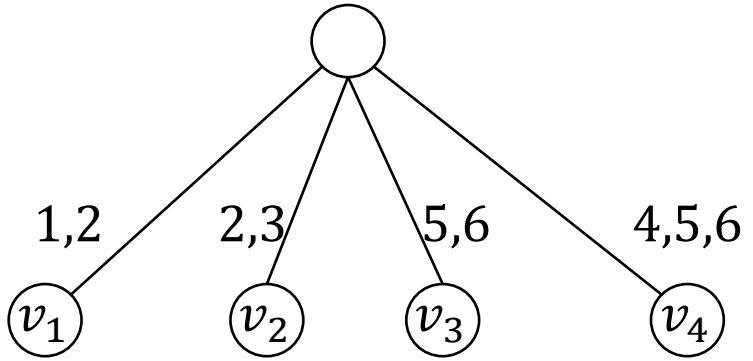


The conditional LB of oracles supporting MD queries

SET DISJOINTNESS PROBLEM: Given k subsets $S_1, \dots, S_k$ of a universe U with $N = \sum_i |S_i|$, design an oracle that can answer to queries *«Given i and j, are S_i and S_j disjoint?»*.

STRONG SET DISJOINTNESS CONJECTURE [Goldstein, Lewenstein, Porat, ISAAC 2019]:
Any oracle that can answer to a query in worst-case time t requires $\widetilde{\Omega}(N^2/t^2)$ space.



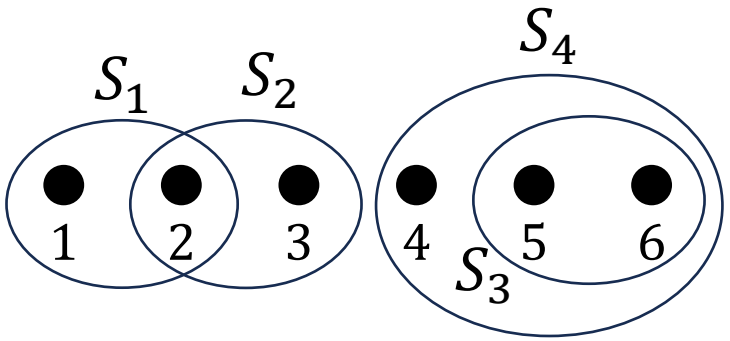


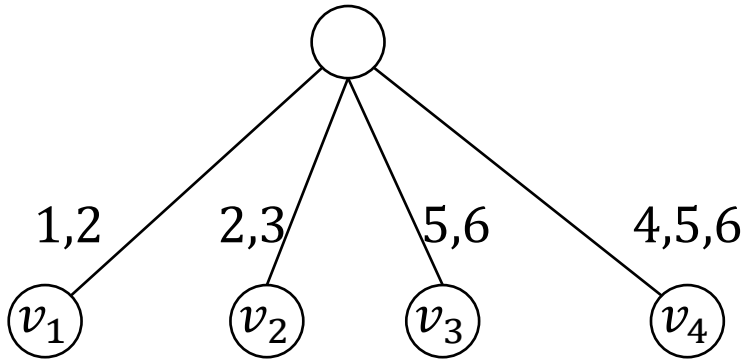
$$S_i \cap S_j \neq \emptyset \Leftrightarrow MD\big(v_i, v_j, [-\infty, +\infty]\big) = 0$$

The conditional LB of oracles supporting MD queries

SET DISJOINTNESS PROBLEM: Given k subsets $S_1, \dots, S_k$ of a universe U with $N = \sum_i |S_i|$, design an oracle that can answer to queries *«Given i and j, are S_i and S_j disjoint?»*.

STRONG SET DISJOINTNESS CONJECTURE [Goldstein, Lewenstein, Porat, ISAAC 2019]:
Any oracle that can answer to a query in worst-case time t requires $\widetilde{\Omega}(N^2/t^2)$ space.





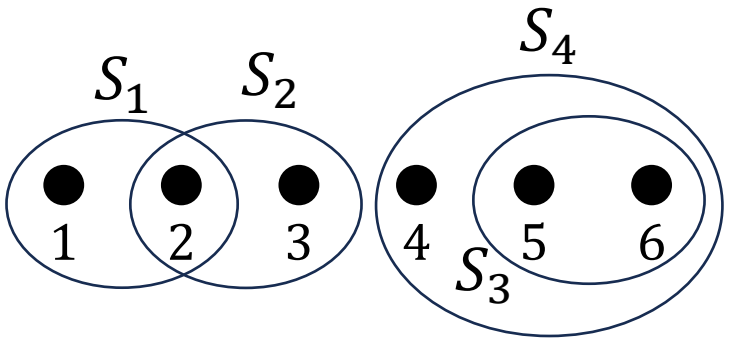
$$S_i \cap S_j \neq \emptyset \Leftrightarrow MD\big(v_i, v_j, [-\infty, +\infty]\big) = 0$$

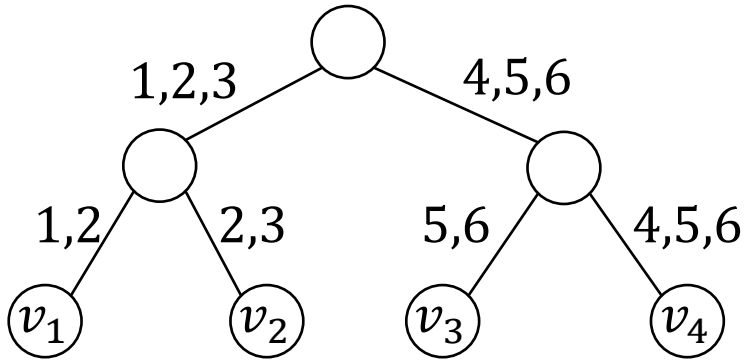
$M = N$ $\Rightarrow$ Any oracle for STATIC temporal TREES supporting MD queries in worst-case time t requires $\widetilde{\Omega}(M^2/t^2)$ space…

The conditional LB of oracles supporting MD queries

SET DISJOINTNESS PROBLEM: Given k subsets $S_1, \dots, S_k$ of a universe U with $N = \sum_i |S_i|$, design an oracle that can answer to queries *«Given i and j, are S_i and S_j disjoint?»*.

STRONG SET DISJOINTNESS CONJECTURE [Goldstein, Lewenstein, Porat, ISAAC 2019]:
Any oracle that can answer to a query in worst-case time t requires $\widetilde{\Omega}(N^2/t^2)$ space.





$$S_i \cap S_j \neq \emptyset \Leftrightarrow MD\big(v_i, v_j, [-\infty, +\infty]\big) = 0$$

$M = O(N \log N) \Rightarrow$ Any oracle for STATIC temporal TREES supporting MD queries in worst-case time t requires $\widetilde{\Omega}(M^2/t^2)$ space even if the tree has max degree 3.

Results presented in this talk

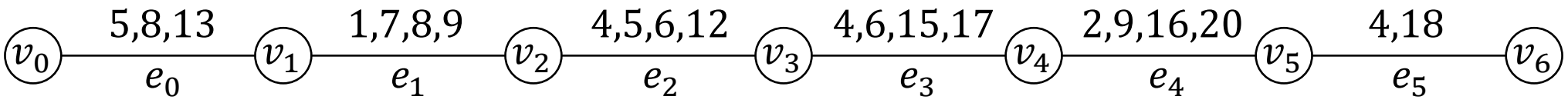
- $M = \sum_e |\lambda(e)|$
- $L = \max_e |\lambda(e)|$

	Size	Time of supported queries			Time of supported updates (both insertions and deletions)		
		TR	EA/LD	MD	Time-labels	Singletons	Edge
(STATIC) Trees [1]	Choose $t > 0$ $O(M + M^2/t^2)$	NO	NO	$O(t \log(1 + L))$	NO	NO	NO
Paths [2]	$O(n + M)$	$O(\log M)$	$O(\log M)$	NO	$O(\log M)$	NO	NO
Forests of rooted trees [2]	$O(n + M)$	$O(\log M)$	$O(\log L \cdot \log M)$	NO	$O(\log M)$	$O(\log M)$	$O(\log M)$

- Tradeoffs in [1] are asymptotically tight, up to polylog factors, under the STRONG SET DISJOINTNESS CONJECTURE

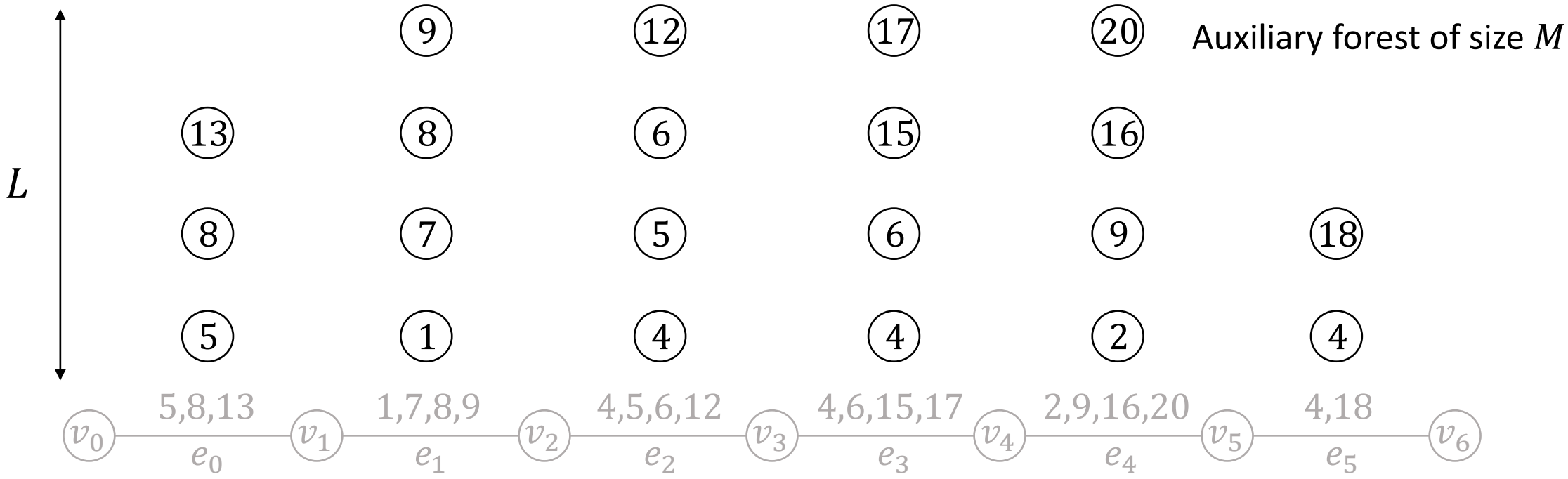
An almost successful attempt

IDEA 1: Model all useful *left-to-right* EA temporal paths.



An almost successful attempt

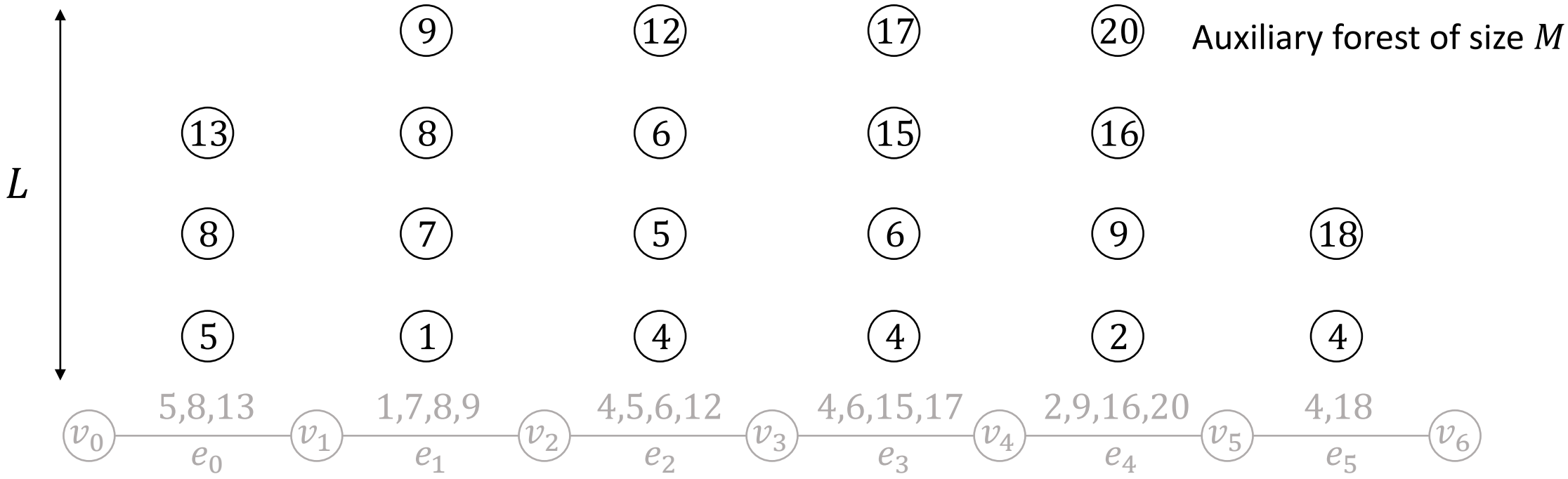
IDEA 1: Model all useful *left-to-right* EA temporal paths.



An almost successful attempt

IDEA 1: Model all useful *left-to-right* EA temporal paths.

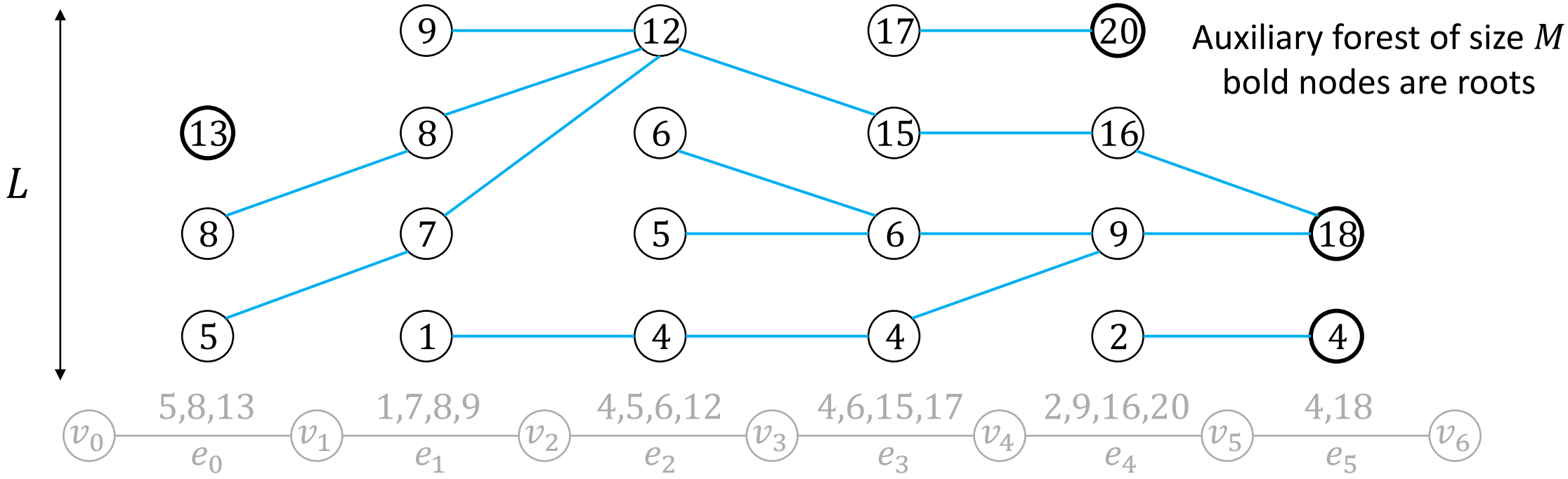
Parent of a node in column i is its non-strict successor in column $i+1$, if any



An almost successful attempt

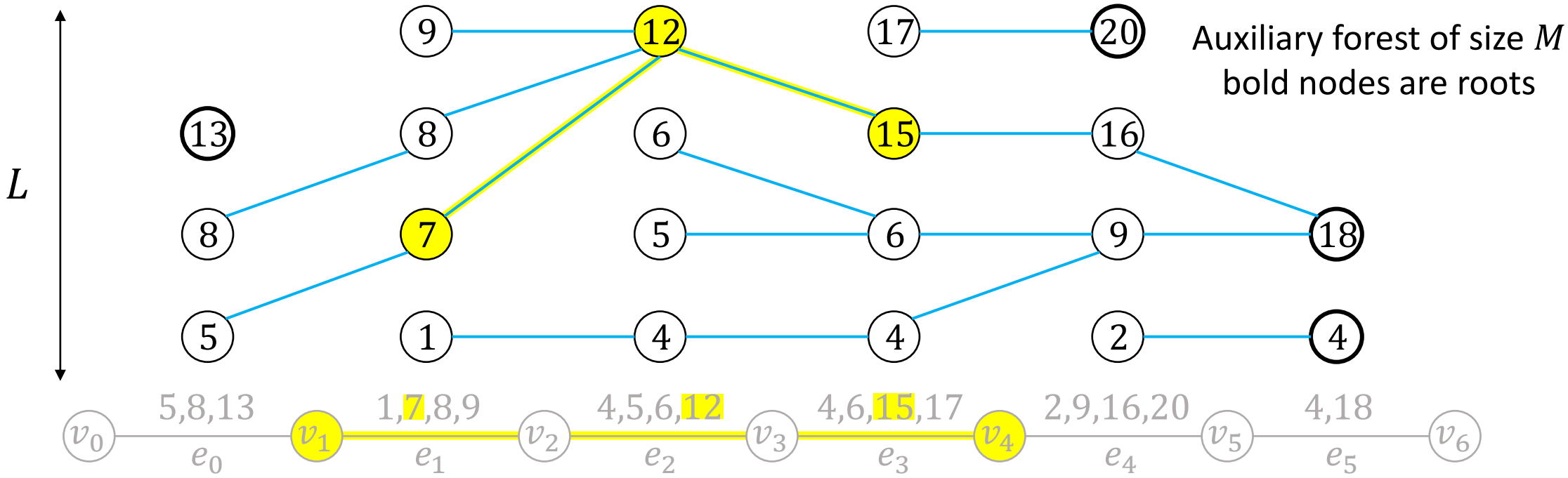
IDEA 1: Model all useful *left-to-right* EA temporal paths.

Parent of a node in column i is its non-strict successor in column $i+1$, if any



An almost successful attempt

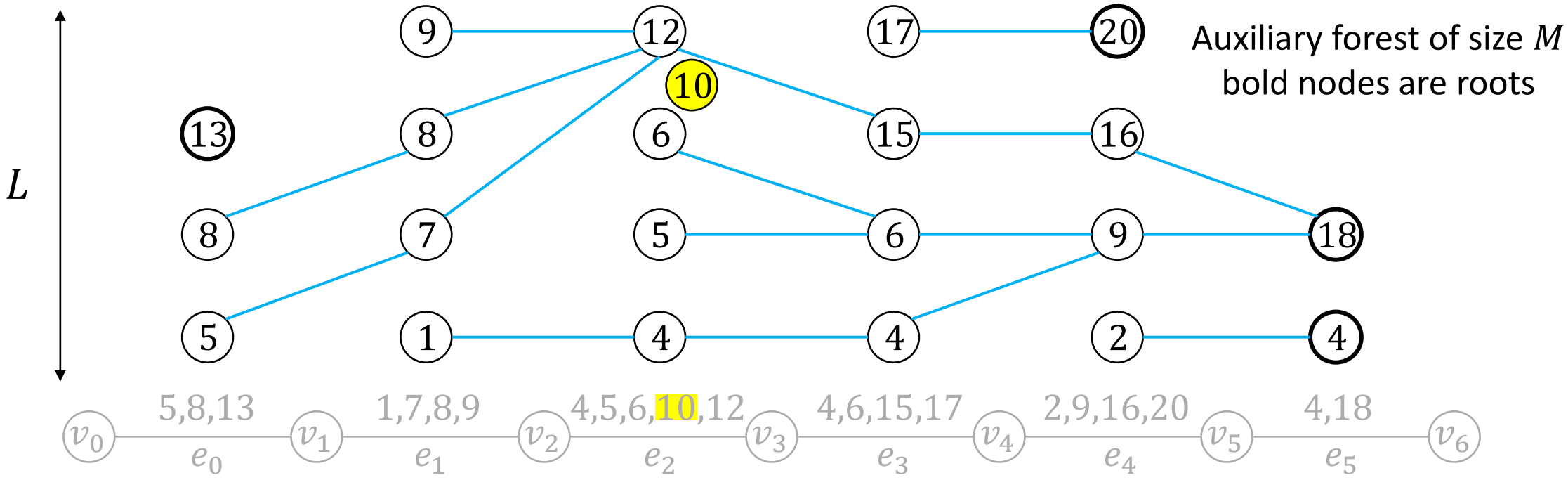
$EA(v_i, v_j, d)$: Find non-strict successor of d in column i and return its $(j - i - 1)$th ancestor



$EA(v_1, v_4, 5) = 15$

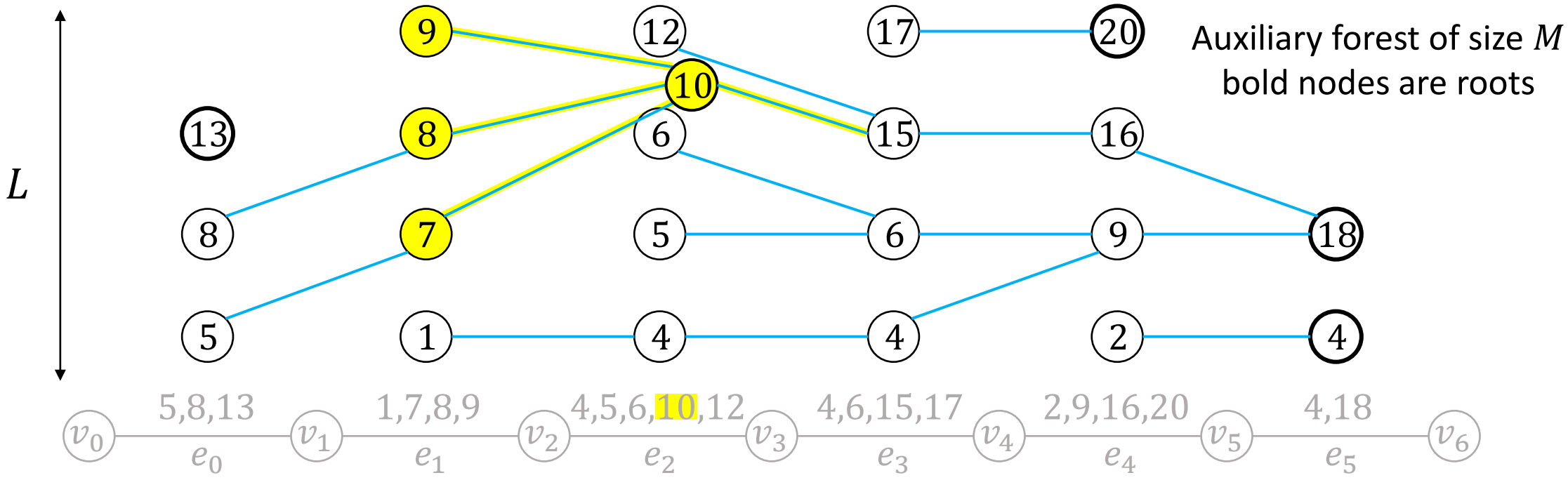
An almost successful attempt

Poor update time when inserting/deleting a time label



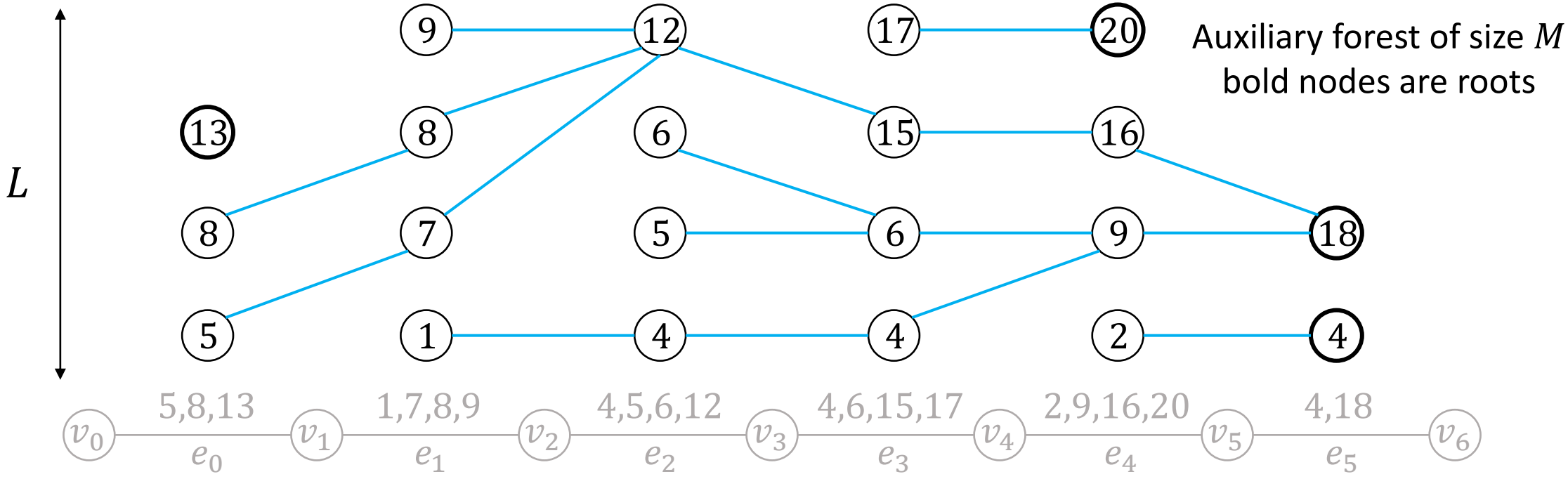
An almost successful attempt

Poor update time when inserting/deleting a time label



How to reduce node degrees

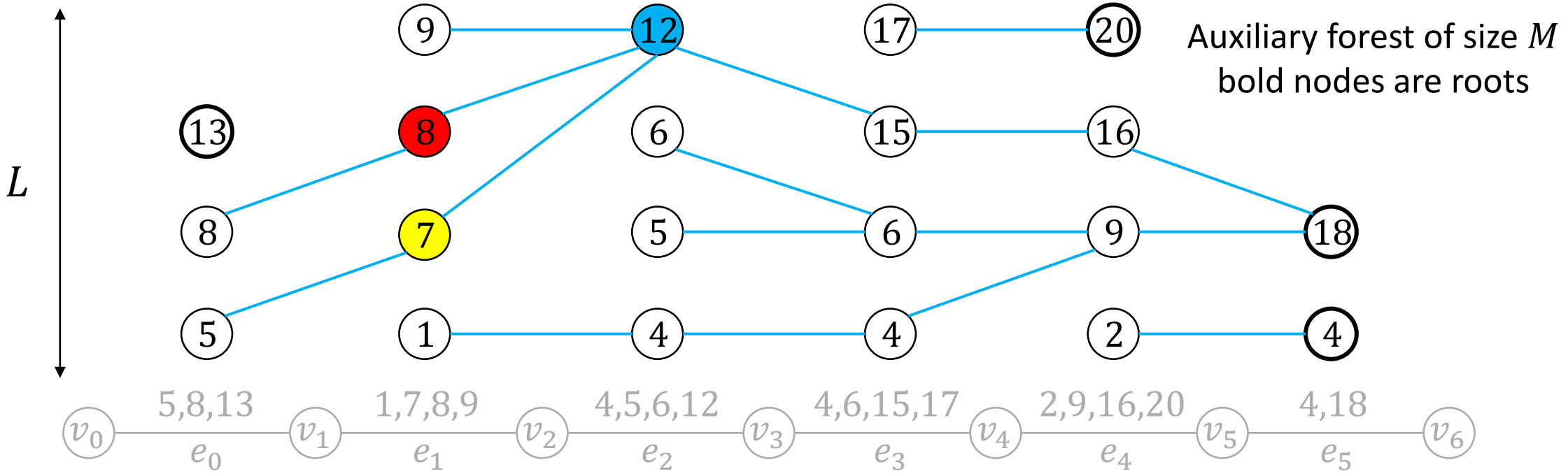
IDEA 2: Reduce the degree of nodes to a constant.



How to reduce node degrees

IDEA 2: Reduce the degree of nodes to a constant.

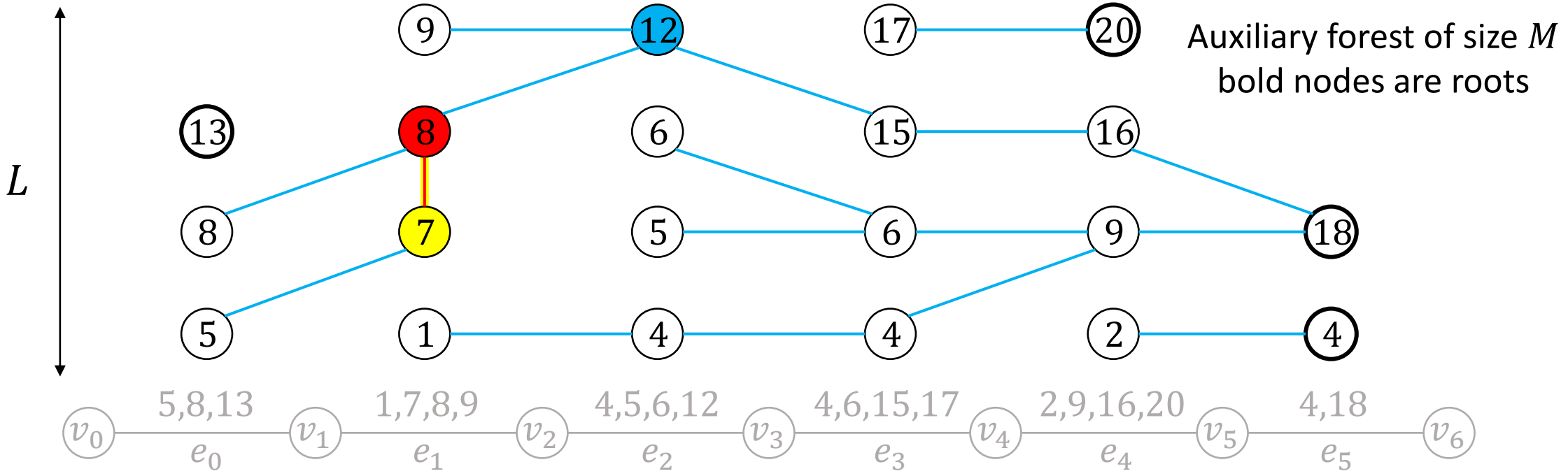
Parent of a node in column i is node of smallest label between
its non-strict successor in column $i+1$ and its strict successor in column i (wins in case of a tie)



How to reduce node degrees

IDEA 2: Reduce the degree of nodes to a constant.

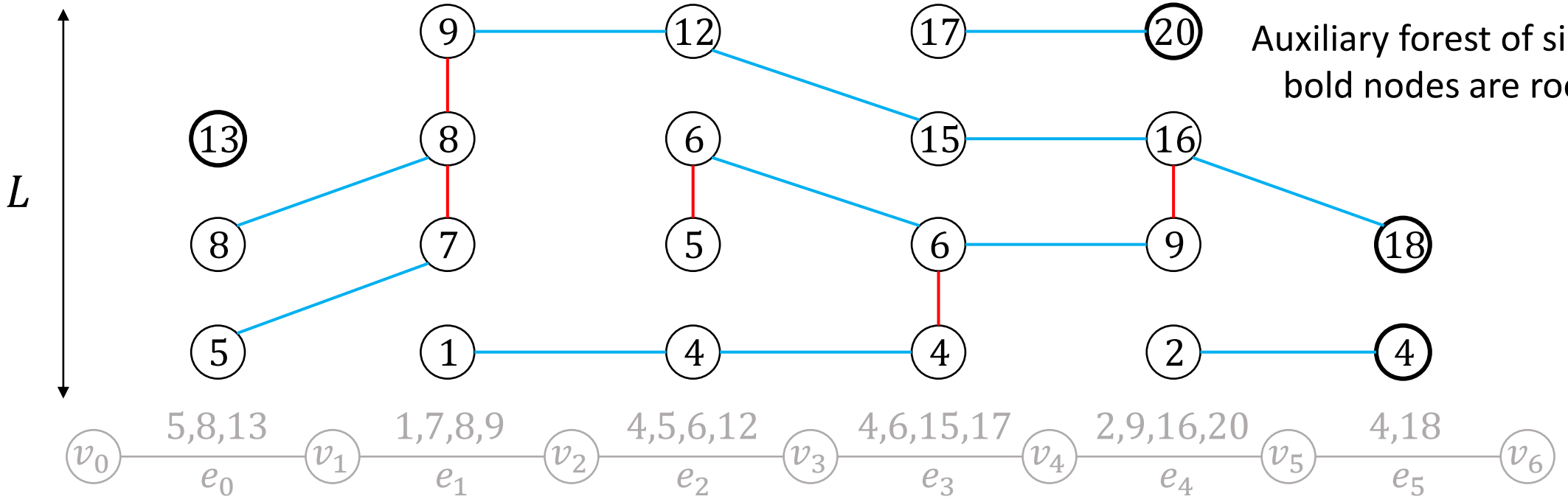
Parent of a node in column i is node of smallest label between
its non-strict successor in column $i+1$ and its strict successor in column i (wins in case of a tie)



How to reduce node degrees

IDEA 2: Reduce the degree of nodes to a constant.

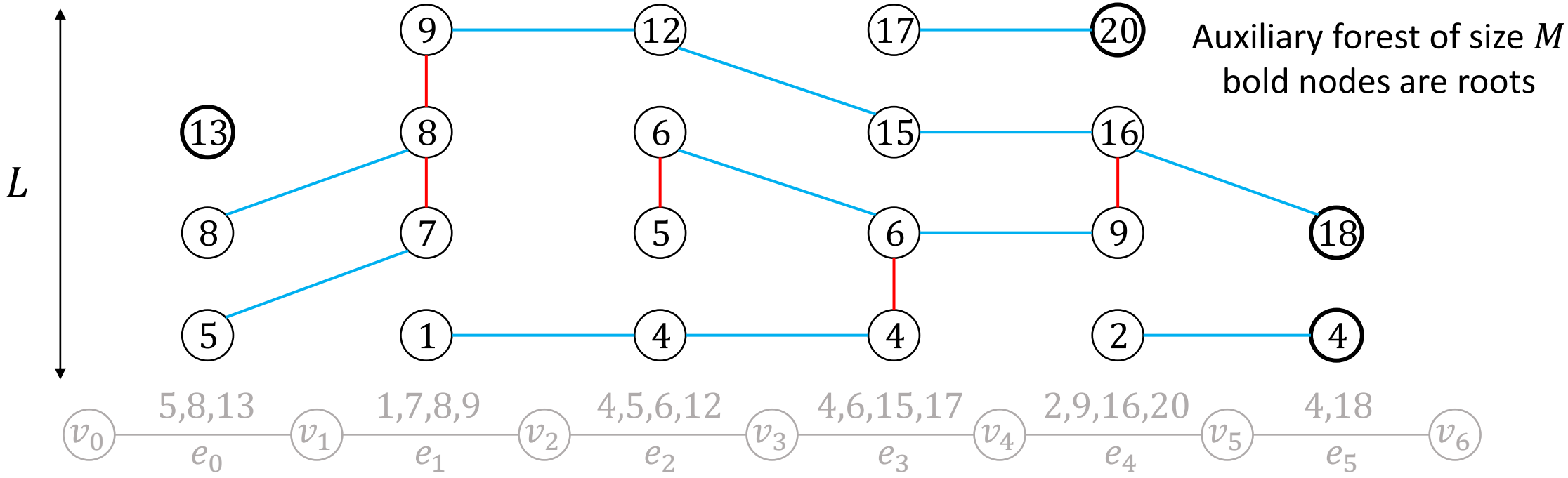
Parent of a node in column i is node of smallest label between
its non-strict successor in column $i+1$ and its strict successor in column i (wins in case of a tie)



Fixing EA queries

IDEA 3: Add edge weights.

Blue edges have weight 1
Red edges have weight 0

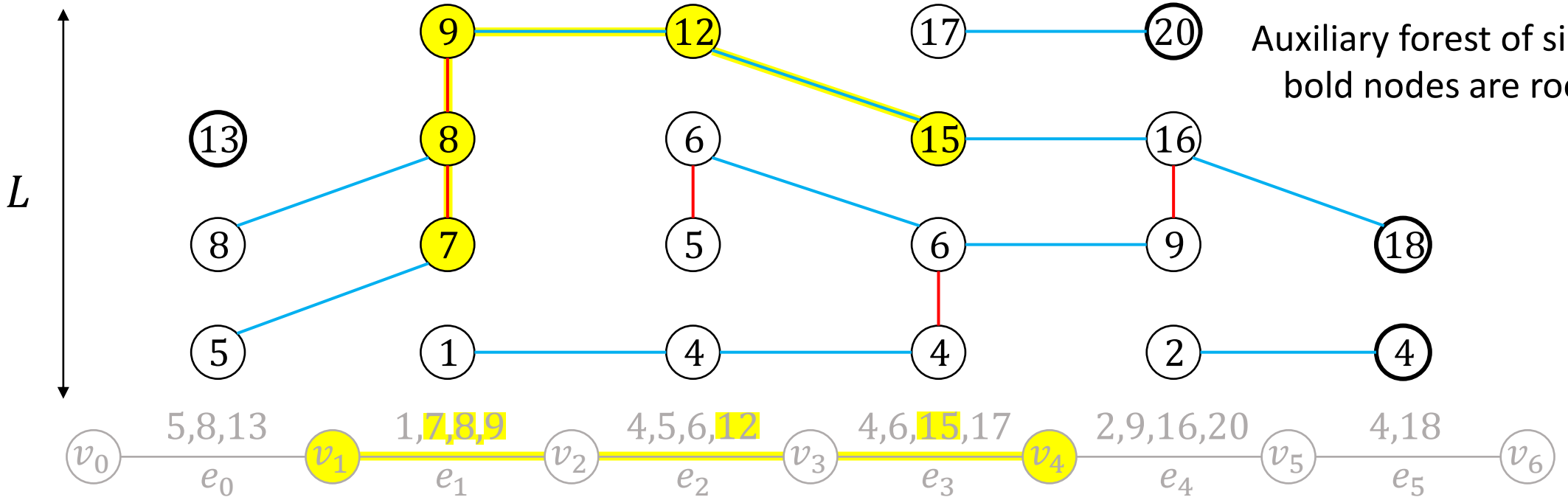


Fixing EA queries

IDEA 3: Add edge weights.

Blue edges have weight 1
Red edges have weight 0

$EA(v_i, v_j, d)$: Find non-strict successor of d in col i; return its deepest ancestor at distance of $j - i - 1$



$EA(v_1, v_4, 5) = 15$

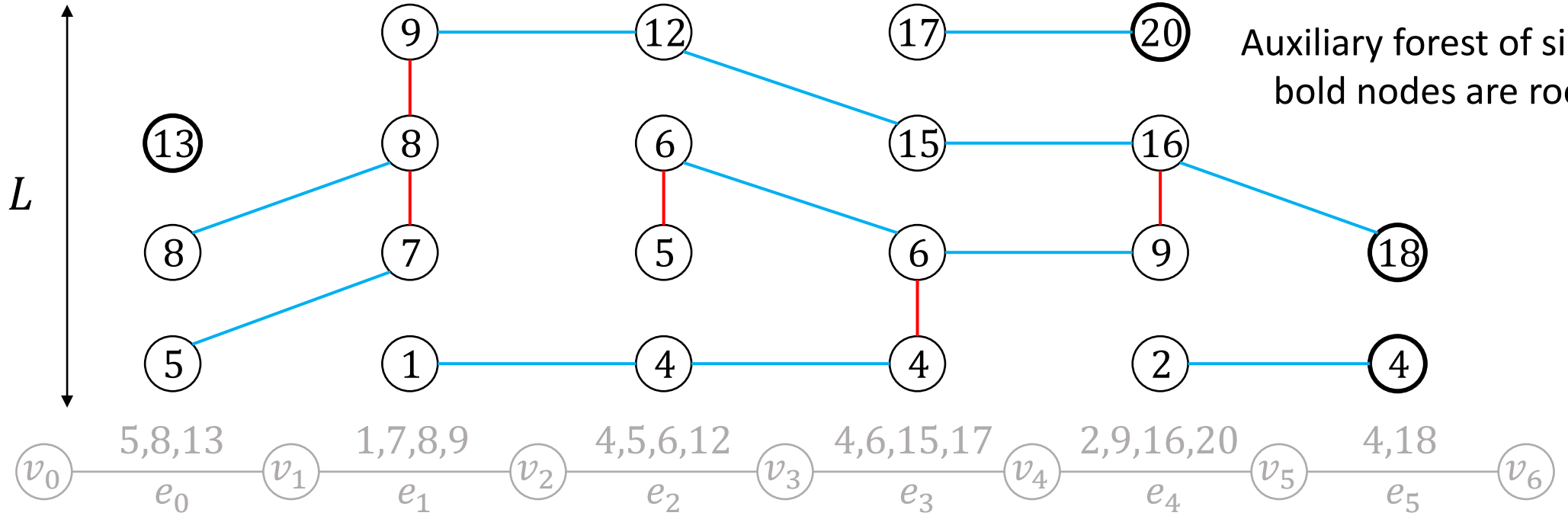
The oracle

A dictionary for each e_i to store its time labels

- total size is $O(M)$
- total building time is $O(M \log L)$
- insertions/deletions in $O(\log L)$ time
- predecessor/successor queries in $O(\log L)$ time

A top tree to store the auxiliary forest

- size is $O(M)$
- building time is $O(M)$
- node/edge insertions/deletions in $O(\log M)$ time
- weighted level-ancestor queries in $O(\log M)$ time



Results presented in this talk

- $M = \sum_e |\lambda(e)|$
- $L = \max_e |\lambda(e)|$

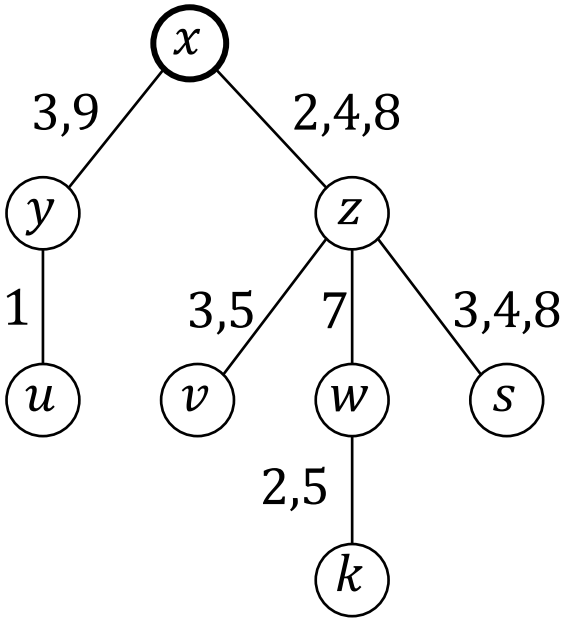
	Size	Time of supported queries			Time of supported updates (both insertions and deletions)		
		TR	EA/LD	MD	Time-labels	Singletons	Edge
(STATIC) Trees [1]	Choose $t > 0$ $O(M + M^2/t^2)$	NO	NO	$O(t \log(1 + L))$	NO	NO	NO
Paths [2]	$O(n + M)$	$O(\log M)$	$O(\log M)$	NO	$O(\log M)$	NO	NO
Forests of rooted trees [2]	$O(n + M)$	$O(\log M)$	$O(\log L \cdot \log M)$	NO	$O(\log M)$	$O(\log M)$	$O(\log M)$

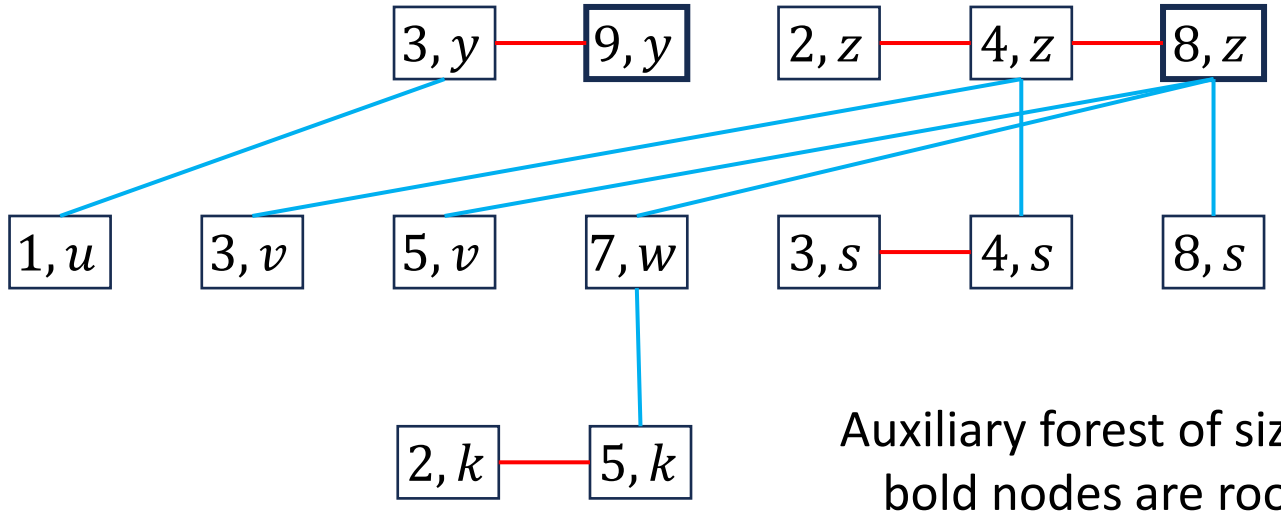
- Tradeoffs in [1] are asymptotically tight, up to polylog factors, under the STRONG SET DISJOINTNESS CONJECTURE

Oracle for dynamic temporal forests of rooted trees

RECYCLED IDEA: Model all useful *upwards* EA temporal paths.

It does not work as we create high-degree nodes.



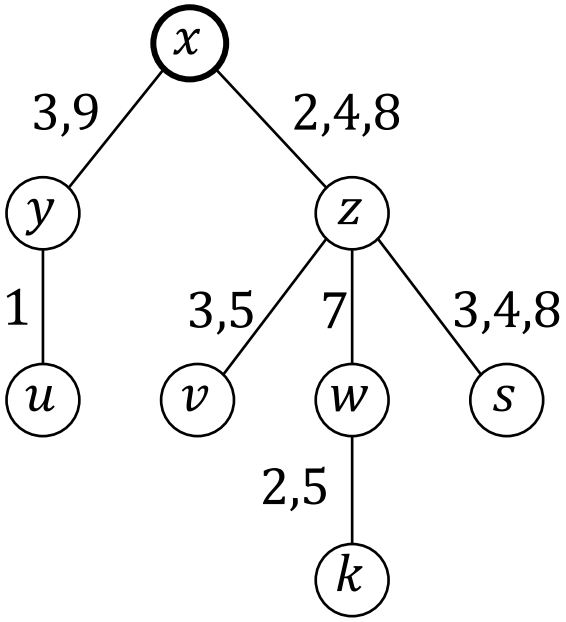


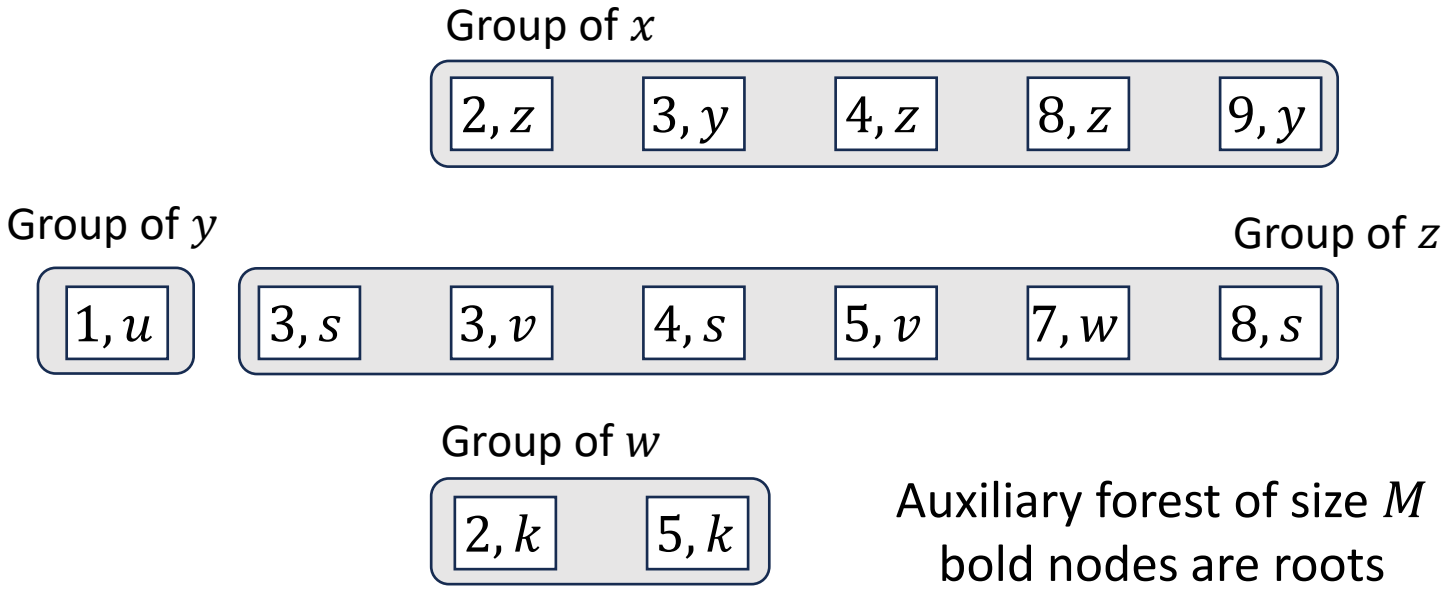
Auxiliary forest of size M
bold nodes are roots

Oracle for dynamic temporal forests of rooted trees

IDEA 2: Reduce the degree of nodes to a constant.

For each non-leaf v, group all nodes created by all edges branching from v





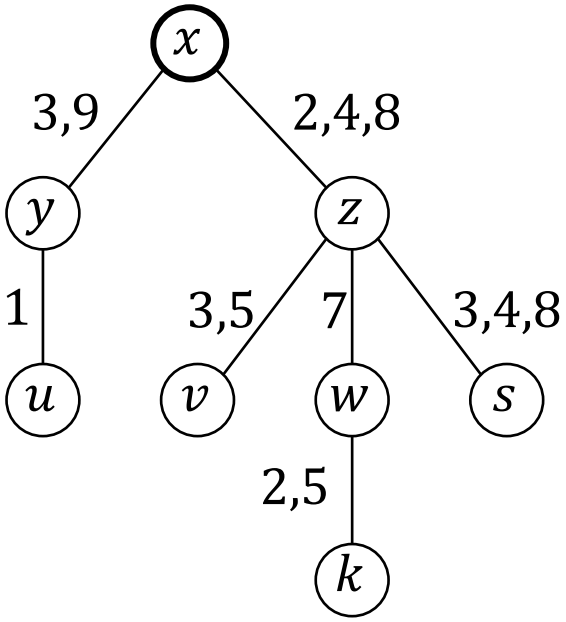
Auxiliary forest of size M
bold nodes are roots

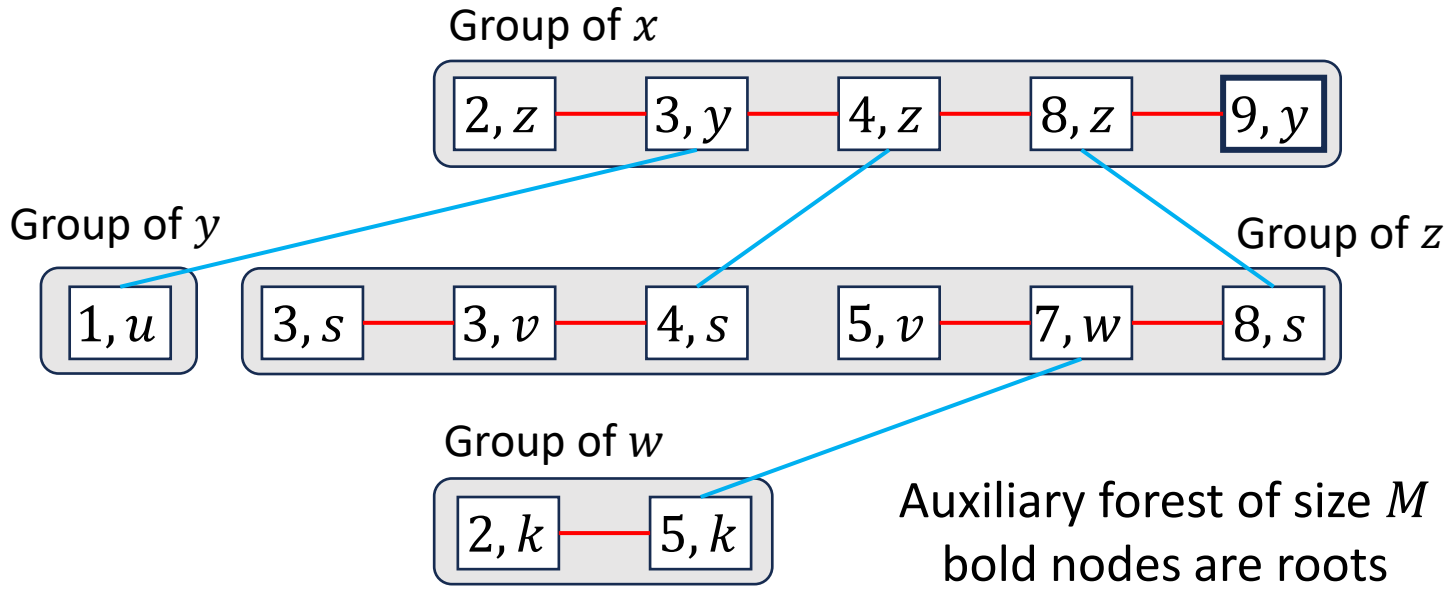
Oracle for dynamic temporal forests of rooted trees

IDEA 2: Reduce the degree of nodes to a constant.

For each non-leaf v, group all nodes created by all edges branching from v

Candidate red parent of a node is computed in its group.

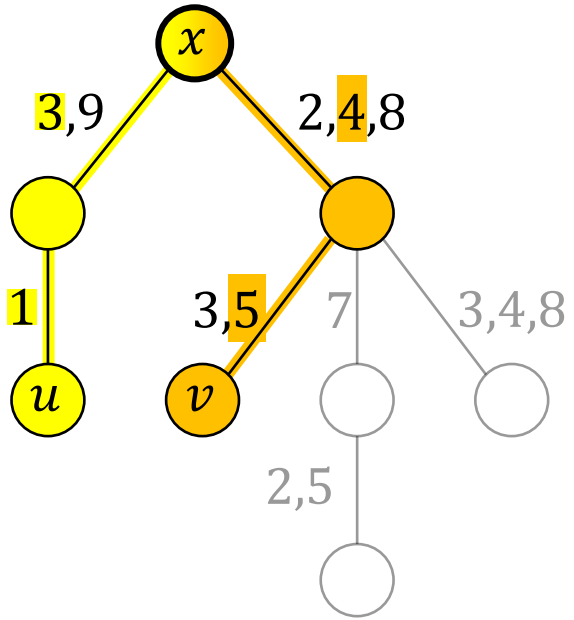




EA queries in $O(\log L \cdot \log M)$ time

$EA(u, v, d)$

1. $x = LCA(u, v)$
2. $d_x = EA(u, x, d)$ // upward query in $O(\log M)$ time
3. Return $EA(x, v, d_x)$ // downward query in $O(\log L \cdot \log M)$ time



$EA(u, v, 1) = 5$

- Upward query $EA(u, x, 1) = 3$
- Downward query $EA(x, v, 3) = 5$

Results presented in this talk

- $M = \sum_e |\lambda(e)|$
- $L = \max_e |\lambda(e)|$

	Size	Time of supported queries			Time of supported updates (both insertions and deletions)		
		TR	EA/LD	MD	Time-labels	Singletons	Edge
(STATIC) Trees [1]	Choose $t > 0$ $O(M + M^2/t^2)$	NO	NO	$O(t \log(1 + L))$	NO	NO	NO
Paths [2]	$O(n + M)$	$O(\log M)$	$O(\log M)$	NO	$O(\log M)$	NO	NO
Forests of rooted trees [2]	$O(n + M)$	$O(\log M)$	$O(\log L \cdot \log M)$	NO	$O(\log M)$	$O(\log M)$	$O(\log M)$

- Tradeoffs in [1] are asymptotically tight, up to polylog factors, under the STRONG SET DISJOINTNESS CONJECTURE

Summary and Open Problems

- Prior to our work, results known for directed graphs supporting TR queries and time-label insertions

Summary and Open Problems

- Prior to our work, results known for directed graphs supporting TR queries and time-label insertions
- We designed oracles with good tradeoffs for
  - STATIC TREES supporting MD queries
    - Asymptotic tight bounds, up to polylog factors, even for trees of max degree 3

Summary and Open Problems

- Prior to our work, results known for directed graphs supporting TR queries and time-label insertions
- We designed oracles with good tradeoffs for
  - STATIC TREES supporting MD queries
    - Asymptotic tight bounds, up to polylog factors, even for trees of max degree 3
  - DYNAMIC temporal FOREST of ROOTED TREES supporting TR/EA/LD queries
    - same amortized update times if we add latencies, but only in the incremental/decremental scenarios

Summary and Open Problems

- Prior to our work, results known for directed graphs supporting TR queries and time-label insertions
- We designed oracles with good tradeoffs for
  - STATIC TREES supporting MD queries
    - Asymptotic tight bounds, up to polylog factors, even for trees of max degree 3
  - DYNAMIC temporal FOREST of ROOTED TREES supporting TR/EA/LD queries
    - same amortized update times if we add latencies, but only in the incremental/decremental scenarios

OPEN PROBLEM 1: Oracles for fully dynamic temporal forests with latencies supporting TR/EA/LD queries

Summary and Open Problems

- Prior to our work, results known for directed graphs supporting TR queries and time-label insertions
- We designed oracles with good tradeoffs for
  - STATIC TREES supporting MD queries
    - Asymptotic tight bounds, up to polylog factors, even for trees of max degree 3
  - DYNAMIC temporal FOREST of ROOTED TREES supporting TR/EA/LD queries
    - same amortized update times if we add latencies, but only in the incremental/decremental scenarios

OPEN PROBLEM 1: Oracles for fully dynamic temporal forests with latencies supporting TR/EA/LD queries

OPEN PROBLEM 2: Better oracles for static paths supporting MD queries

Summary and Open Problems

- Prior to our work, results known for directed graphs supporting TR queries and time-label insertions
- We designed oracles with good tradeoffs for
  - STATIC TREES supporting MD queries
    - Asymptotic tight bounds, up to polylog factors, even for trees of max degree 3
  - DYNAMIC temporal FOREST of ROOTED TREES supporting TR/EA/LD queries
    - same amortized update times if we add latencies, but only in the incremental/decremental scenarios

OPEN PROBLEM 1: Oracles for fully dynamic temporal forests with latencies supporting TR/EA/LD queries

OPEN PROBLEM 2: Better oracles for static paths supporting MD queries

OPEN PROBLEM 3: Oracles for dynamic trees supporting MD queries and time-label updates

Summary and Open Problems

- Prior to our work, results known for directed graphs supporting TR queries and time-label insertions
- We designed oracles with good tradeoffs for
  - STATIC TREES supporting MD queries
    - Asymptotic tight bounds, up to polylog factors, even for trees of max degree 3
  - DYNAMIC temporal FOREST of ROOTED TREES supporting TR/EA/LD queries
    - same amortized update times if we add latencies, but only in the incremental/decremental scenarios

OPEN PROBLEM 1: Oracles for fully dynamic temporal forests with latencies supporting TR/EA/LD queries

OPEN PROBLEM 2: Better oracles for static paths supporting MD queries

OPEN PROBLEM 3: Oracles for dynamic trees supporting MD queries and time-label updates

OPEN PROBLEM 4: Oracles for temporal graphs
(possibly with some useful restrictions like simple temporal graphs, or bounded treewidth)

Summary and Open Problems

- Prior to our work, results known for directed graphs supporting TR queries and time-label insertions
- We designed oracles with good tradeoffs for
  - STATIC TREES supporting MD queries
    - Asymptotic tight bounds, up to polylog factors, even for trees of max degree 3
  - DYNAMIC temporal FOREST of ROOTED TREES supporting TR/EA/LD queries
    - same amortized update times if we add latencies, but only in the incremental/decremental scenarios

OPEN PROBLEM 1: Oracles for fully dynamic temporal forests with latencies supporting TR/EA/LD queries

OPEN PROBLEM 2: Better oracles for static paths supporting MD queries

OPEN PROBLEM 3: Oracles for dynamic trees supporting MD queries and time-label updates

OPEN PROBLEM 4: Oracles for temporal graphs
(possibly with some useful restrictions like simple temporal graphs, or bounded treewidth)

OPEN PROBLEM 5: Oracles supporting other types of queries (e.g., min travel time, min waiting time, etc.)

Thanks for your attention