Computing Parameters of Sequence-based Dynamic Graphs

Ralf Klasing

LaBRI, CNRS, University of Bordeaux, France

**This is a joint work with Arnaud Casteigts, Yessin M. Neggaz, and Joseph G. Peters.

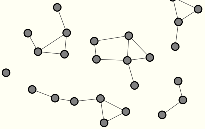
Dynamic Networks

- Highly dynamic networks?

Dynamic Networks

- Highly dynamic networks?

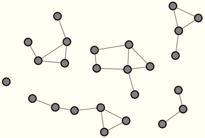
- How changes are perceived?
  - ~~Faults and Failures?~~
  - Nature of the system
  - Change is normal

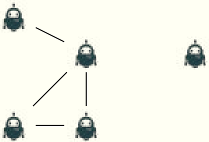


Dynamic Networks

- Highly dynamic networks?

- How changes are perceived?
  - ~~Faults and Failures?~~
  - Nature of the system
  - Change is normal

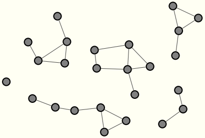


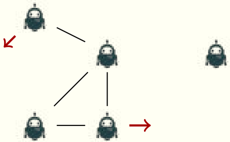


Dynamic Networks

- Highly dynamic networks?

- How changes are perceived?
  - ~~Faults and Failures?~~
  - Nature of the system
  - Change is normal

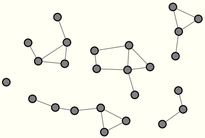


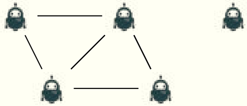


Dynamic Networks

- Highly dynamic networks?

- How changes are perceived?
  - ~~Faults and Failures?~~
  - Nature of the system
  - Change is normal

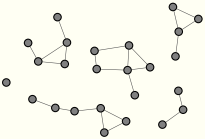


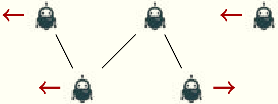


Dynamic Networks

- Highly dynamic networks?

- How changes are perceived?
  - ~~Faults and Failures?~~
  - Nature of the system
  - Change is normal

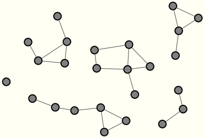


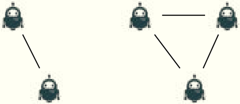


Dynamic Networks

- Highly dynamic networks?

- How changes are perceived?
  - ~~Faults and Failures?~~
  - Nature of the system
  - Change is normal

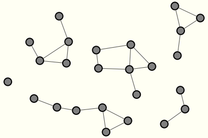


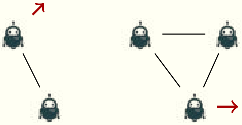


Dynamic Networks

- Highly dynamic networks?

- How changes are perceived?
  - ~~Faults and Failures?~~
  - Nature of the system
  - Change is normal

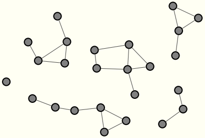


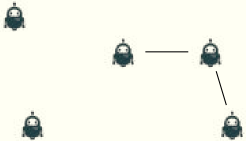


Dynamic Networks

- Highly dynamic networks?

- How changes are perceived?
  - ~~Faults and Failures?~~
  - Nature of the system
  - Change is normal

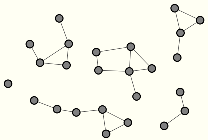


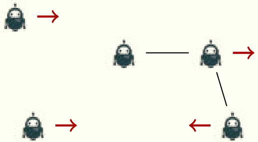


Dynamic Networks

- Highly dynamic networks?

- How changes are perceived?
  - ~~Faults and Failures?~~
  - Nature of the system
  - Change is normal

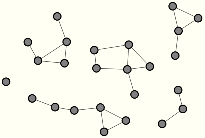


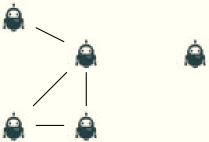


Dynamic Networks

- Highly dynamic networks?

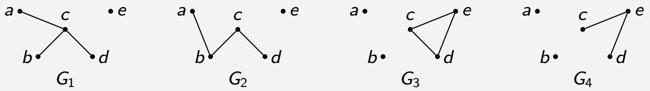
- How changes are perceived?
  - ~~Faults and Failures?~~
  - Nature of the system
  - Change is normal





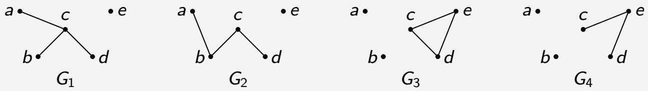
Dynamic Graphs

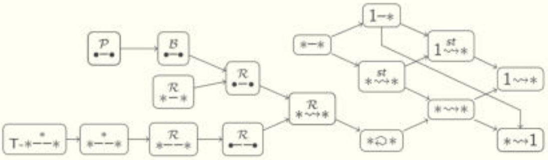
Dynamic graphs: Various forms: TVG, Evolving graphs



Dynamic Graphs

Dynamic graphs: Various forms: TVG, Evolving graphs



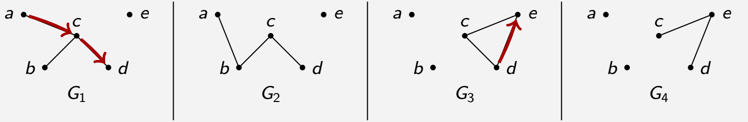


Dynamic graphs classes: [Casteigts, Flocchini, Quattrociocchi et Santoro, 2011]

Temporal Connectivity

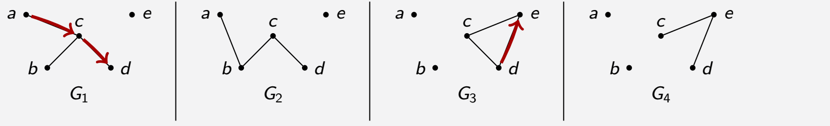
Temporal Connectivity

$\mathcal{G} = (V, E_i)$



Temporal Connectivity

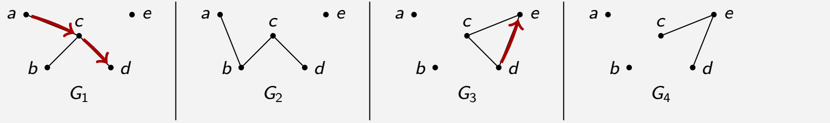
$\mathcal{G} = (V, E_i)$



- **Temporal connectivity** $\iff \forall u, v \in V, u \rightsquigarrow v$.

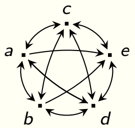
Temporal Connectivity

$\mathcal{G} = (V, E_i)$



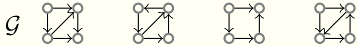
- **Temporal connectivity** $\iff \forall u, v \in V, u \rightsquigarrow v$.
- **Transitive closure** of the journeys: reachability over time *[Bhadra and Ferreira, 2003]*

$\mathcal{G}^*$



- $\mathcal{G}$ is **temporally connected** $\Leftrightarrow$ Transitive closure $\mathcal{G}^*$ is **complete**

High-level Strategy



High-level Strategy

$G_{(1,4)}$

- High-level strategies that work directly at the graph level
- Elementary graph-level operations

$\mathcal{G}$

G_1 G_2 G_3 G_4

High-level Strategy

$G_{(1,4)}$

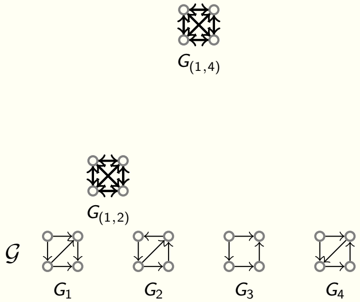
- High-level strategies that work directly at the graph level
- Elementary graph-level operations

$\mathcal{G}$ G_1 G_2 G_3 G_4

TEMPORAL-DIAMETER

Finding the *temporal diameter* of a given dynamic graph $\mathcal{G}$, i.e. the smallest duration in which there exists a journey from any node to all other nodes.

High-level Strategy

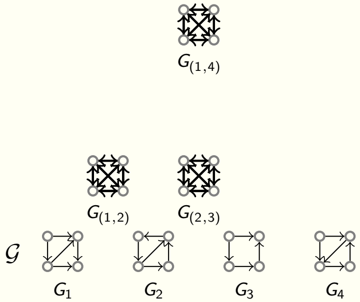


- High-level strategies that work directly at the graph level
- Elementary graph-level operations

TEMPORAL-DIAMETER

Finding the *temporal diameter* of a given dynamic graph $\mathcal{G}$, i.e. the smallest duration in which there exists a journey from any node to all other nodes.

High-level Strategy

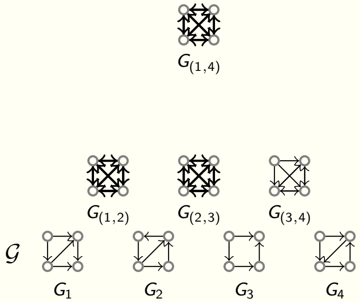


- High-level strategies that work directly at the graph level
- Elementary graph-level operations

TEMPORAL-DIAMETER

Finding the *temporal diameter* of a given dynamic graph $\mathcal{G}$, i.e. the smallest duration in which there exists a journey from any node to all other nodes.

High-level Strategy

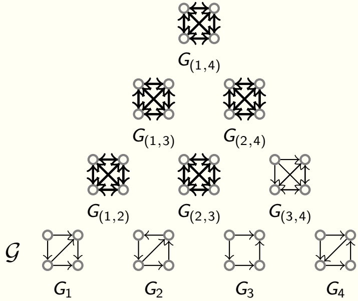


TEMPORAL-DIAMETER

Finding the *temporal diameter* of a given dynamic graph $\mathcal{G}$, i.e. the smallest duration in which there exists a journey from any node to all other nodes.

High-level Strategy

- Transitive closures
- Completeness test

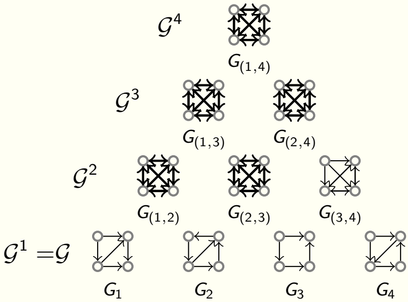


Temporal-Diameter

Finding the *temporal diameter* of a given dynamic graph $\mathcal{G}$, i.e. the smallest duration in which there exists a journey from any node to all other nodes.

High-level Strategy

- Transitive closures
- Completeness test

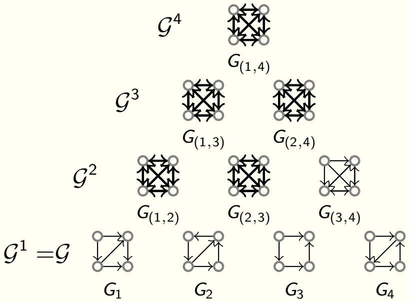


TEMPORAL-DIAMETER

Finding the *temporal diameter* of a given dynamic graph $\mathcal{G}$, i.e. the smallest duration in which there exists a journey from any node to all other nodes.

High-level Strategy

- Transitive closures
- Completeness test



TEMPORAL-DIAMETER

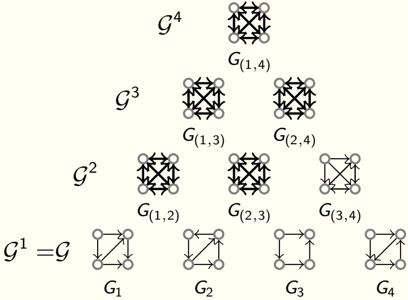
Finding the *temporal diameter* of a given dynamic graph $\mathcal{G}$, i.e. the smallest duration in which there exists a journey from any node to all other nodes.

$$\Longleftrightarrow$$

Finding the smallest d such that every super node in row $\mathcal{G}^d$ is a complete graph (i.e. every subsequence of length d is temporally connected).

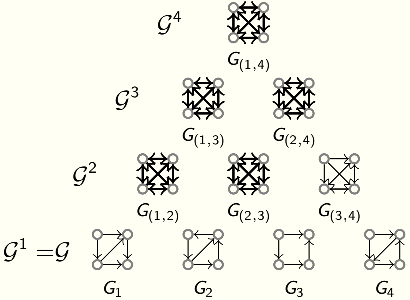
High-level Strategy

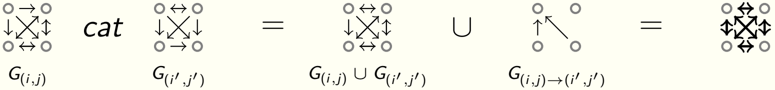
- Transitive closures
- Completeness test
- Transitive closures concatenation



High-level Strategy

- Transitive closures
- Completeness test
- Transitive closures concatenation

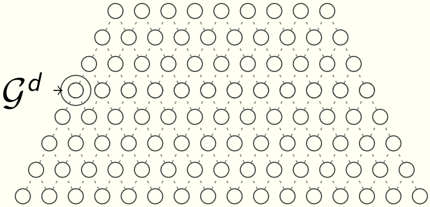




Temporal Diameter Computation

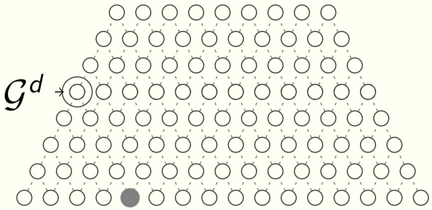
Temporal Diameter Computation

Decision version (given d)



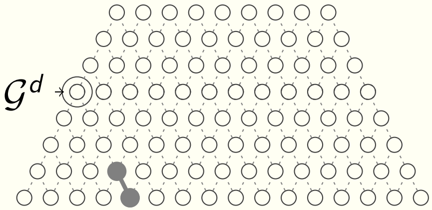
Temporal Diameter Computation

Decision version (given d)



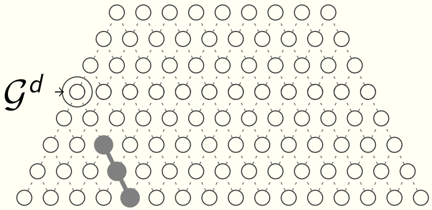
Temporal Diameter Computation

Decision version (given d)



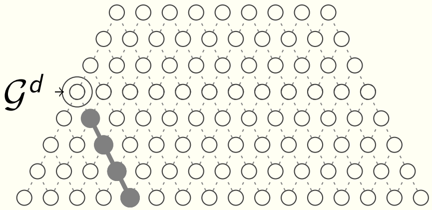
Temporal Diameter Computation

Decision version (given d)



Temporal Diameter Computation

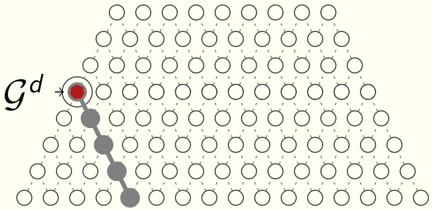
Decision version (given d)



Temporal Diameter Computation

Decision version (given d)

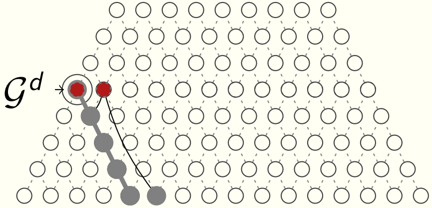
- A ladder of length l costs $l - 1$ concatenation



Temporal Diameter Computation

Decision version (given d)

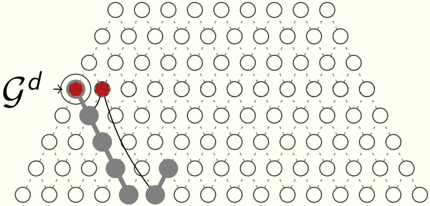
- A ladder of length l costs $l-1$ concatenation



Temporal Diameter Computation

Decision version (given d)

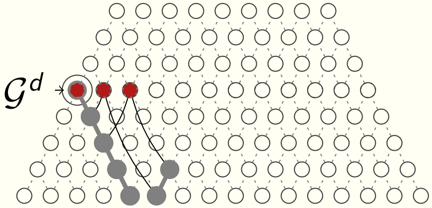
- A ladder of length l costs $l-1$ concatenation



Temporal Diameter Computation

Decision version (given d)

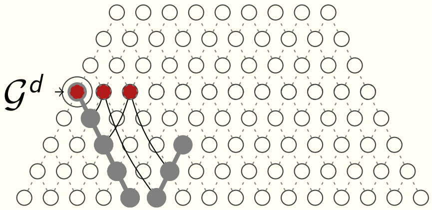
- A ladder of length l costs $l-1$ concatenation
- Use left and right ladders



Temporal Diameter Computation

Decision version (given d)

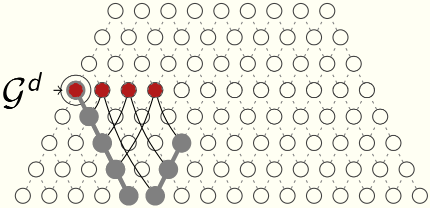
- A ladder of length l costs $l - 1$ concatenation
- Use left and right ladders



Temporal Diameter Computation

Decision version (given d)

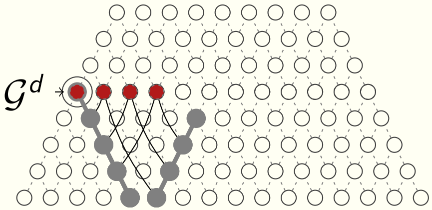
- A ladder of length l costs $l-1$ concatenation
- Use left and right ladders



Temporal Diameter Computation

Decision version (given d)

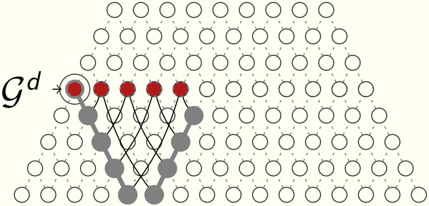
- A ladder of length l costs $l-1$ concatenation
- Use left and right ladders



Temporal Diameter Computation

Decision version (given d)

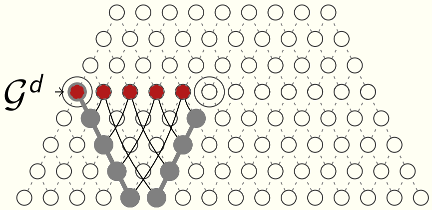
- A ladder of length l costs $l-1$ concatenation
- Use left and right ladders
- Any graph “between” two ladders (red graphs) can be computed by a single binary concatenation



Temporal Diameter Computation

Decision version (given d)

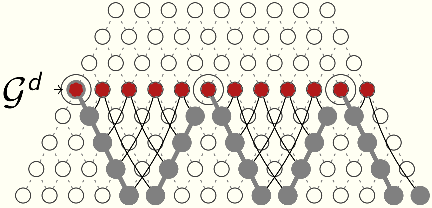
- A ladder of length l costs $l - 1$ concatenation
- Use left and right ladders
- Any graph "between" two ladders (red graphs) can be computed by a single binary concatenation



Temporal Diameter Computation

Decision version (given d)

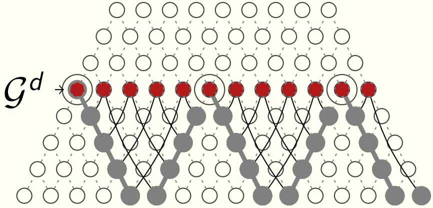
- A ladder of length l costs $l-1$ concatenation
- Use left and right ladders
- Any graph "between" two ladders (red graphs) can be computed by a single binary concatenation



Temporal Diameter Computation

Decision version (given d)

- A ladder of length l costs $l-1$ concatenation
- Use left and right ladders
- Any graph "between" two ladders (red graphs) can be computed by a single binary concatenation



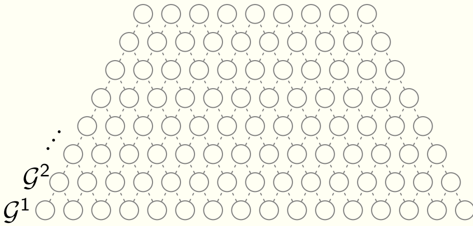
$O(\delta)$ **elementary operations per row**

Temporal Diameter Computation

Minimization version (find the temporal diameter d)

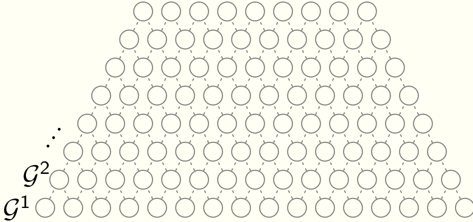
Transitive Closures Computation

Minimization version (find the temporal diameter d)



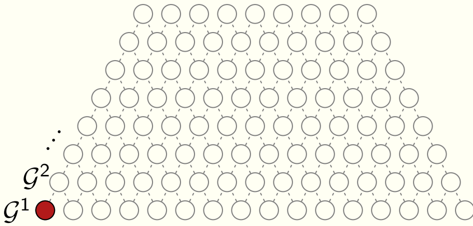
Transitive Closures Computation

Minimization version (find the temporal diameter d)



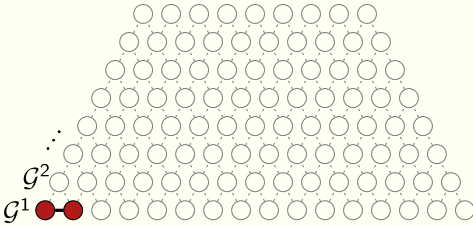
Transitive Closures Computation

Minimization version (find the temporal diameter d)



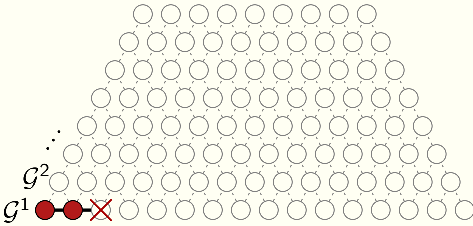
Transitive Closures Computation

Minimization version (find the temporal diameter d)



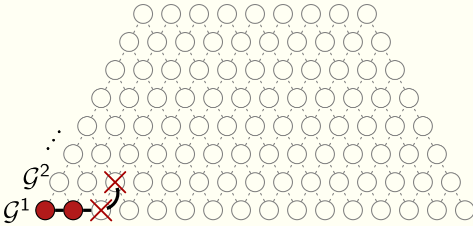
Transitive Closures Computation

Minimization version (find the temporal diameter d)



Transitive Closures Computation

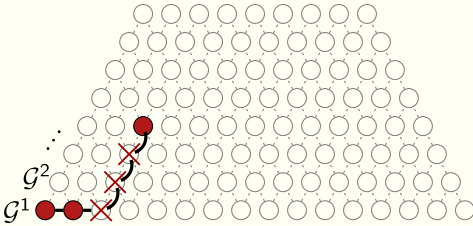
Minimization version (find the temporal diameter d)



Transitive Closures Computation

Minimization version (find the temporal diameter d)

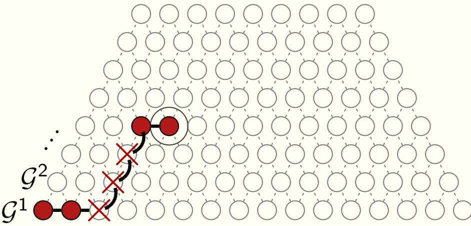
- Strategy: ascending walk



Transitive Closures Computation

Minimization version (find the temporal diameter d)

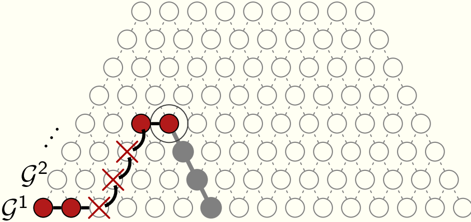
- Strategy: ascending walk



Transitive Closures Computation

Minimization version (find the temporal diameter d)

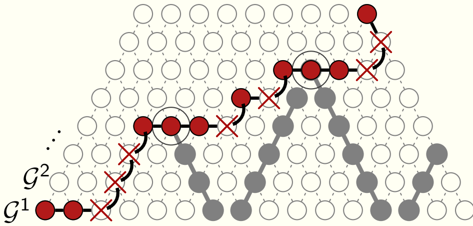
- Strategy: ascending walk



Transitive Closures Computation

Minimization version (find the temporal diameter d)

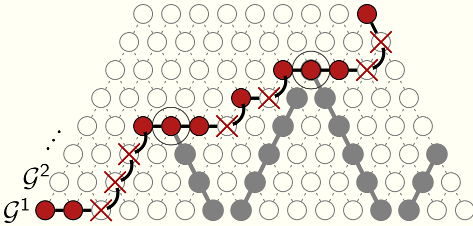
- Strategy: ascending walk



Transitive Closures Computation

Minimization version (find the temporal diameter d)

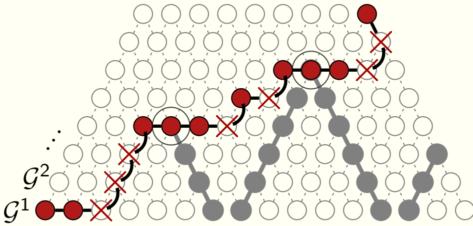
- Strategy: ascending walk
- The total length of the ladders is $O(\delta)$
- At most $O(\delta)$ binary concatenation and completeness tests



Transitive Closures Computation

Minimization version (find the temporal diameter d)

- Strategy: ascending walk
- The total length of the ladders is $O(\delta)$
- At most $O(\delta)$ binary concatenation and completeness tests

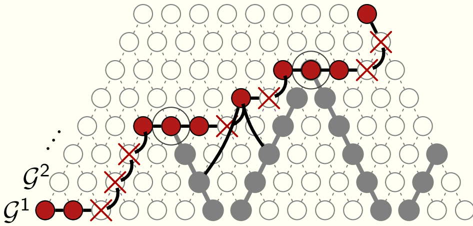


- **Disjointness property:** $cat(G_{(i,j)}, G_{(i',j')}) = G_{(i,j')}$

Transitive Closures Computation

Minimization version (find the temporal diameter d)

- Strategy: ascending walk
- The total length of the ladders is $O(\delta)$
- At most $O(\delta)$ binary concatenation and completeness tests

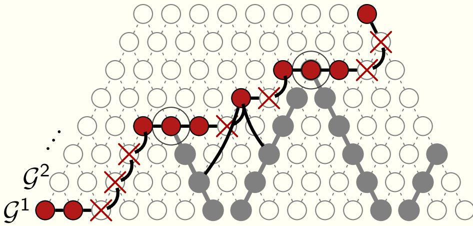


- **Disjointness property:** $cat(G_{(i,j)}, G_{(i',j')}) = G_{(i,j')}$

Transitive Closures Computation

Minimization version (find the temporal diameter d)

- Strategy: ascending walk
- The total length of the ladders is $O(\delta)$
- At most $O(\delta)$ binary concatenation and completeness tests

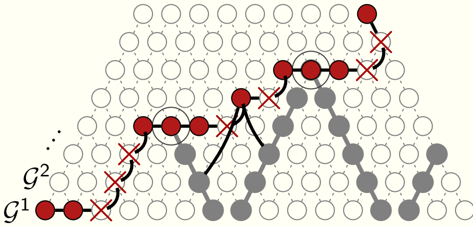


- **Disjointness property:** $cat(G_{(i,j)}, G_{(i',j')}) = G_{(i,j')}$
- **If** $G_{(i,j)}$**is complete, then** $G_{(i',j')}$ **is complete, for all** $i' \leq i$ **and** $j' \geq j$

Transitive Closures Computation

Minimization version (find the temporal diameter d)

- Strategy: ascending walk
- The total length of the ladders is $O(\delta)$
- At most $O(\delta)$ binary concatenation and completeness tests



- **Disjointness property:** $cat(G_{(i,j)}, G_{(i',j')}) = G_{(i,j')}$
- **If** $G_{(i,j)}$ **is complete, then** $G_{(i',j')}$ **is complete, for all** $i' \leq i$ **and** $j' \geq j$

`Temporal-Diameter` **is solvable with** $O(\delta)$ **elementary operations**

Online Algorithms

Online Algorithms

- The optimal algorithms can be adapted to an **online setting**
- The sequence of graphs $G_1, G_2, G_3, \ldots$ of $\mathcal{G}$ is processed in the order of reception
- **Amortized cost of** $O(1)$ elementary operations per graph received
- Dynamic version: consider only the recent history

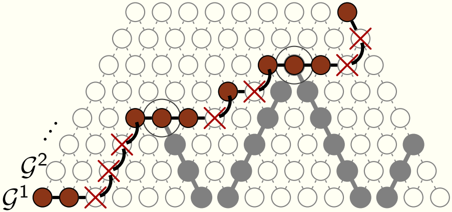
Generic Framework

A Generic Framework

- Solve other problems using the same framework

A Generic Framework

- Solve other problems using the same framework

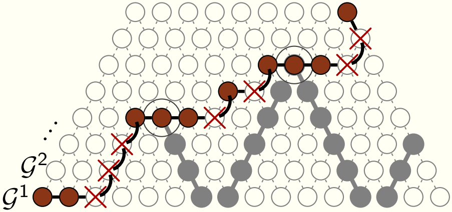


Framework generalization

- Transitive closures concatenation
- Completeness test
- Transitive closure

A Generic Framework

- Solve other problems using the same framework

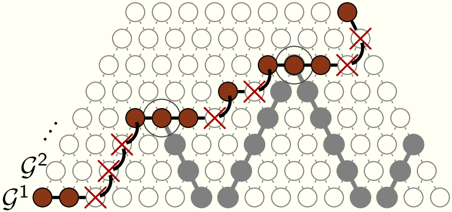


Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems

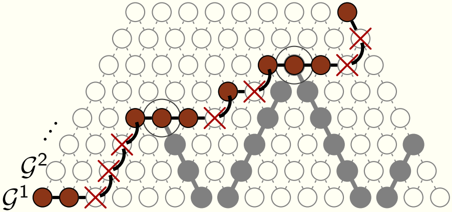
Find the **smallest** value

Framework generalization

- Transitive closures concatenation $\rightarrow$ **Composition operation**
- Completeness test $\rightarrow$ **Test operation**
- Transitive closure $\rightarrow$ **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

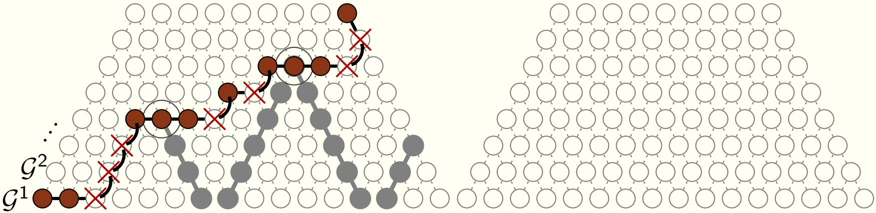
Find the **smallest** value | Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

Find the **smallest** value

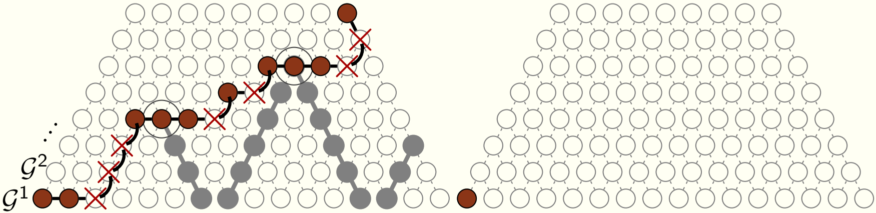
Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

Find the **smallest** value

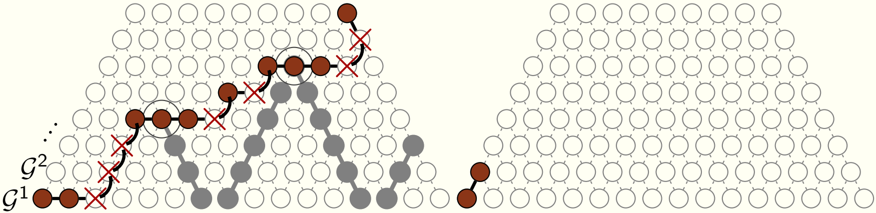
Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

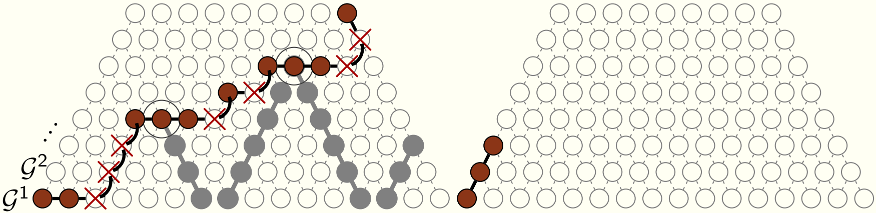
Find the **smallest** value — Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

Find the **smallest** value

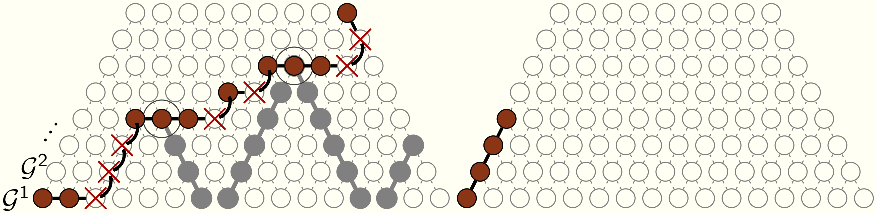
Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

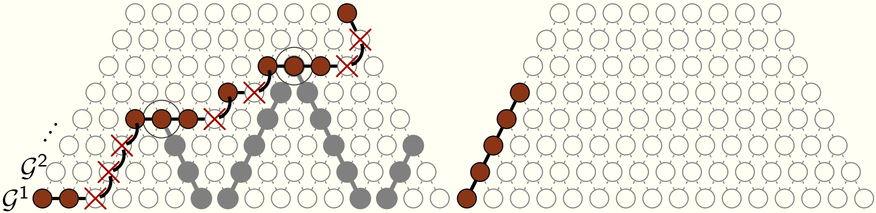
Find the **smallest** value — Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

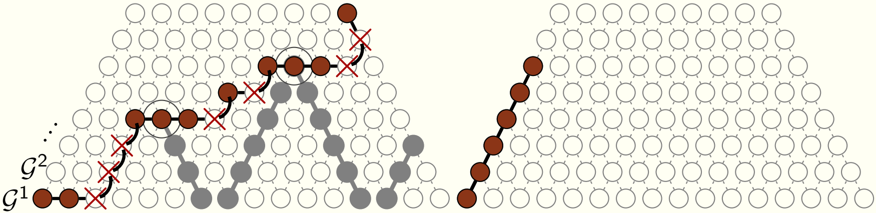
Find the **smallest** value | Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

Find the **smallest** value

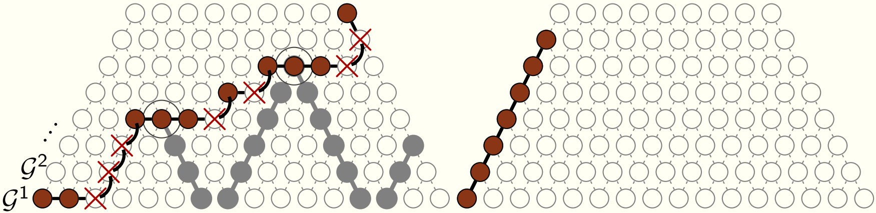
Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

Find the **smallest** value

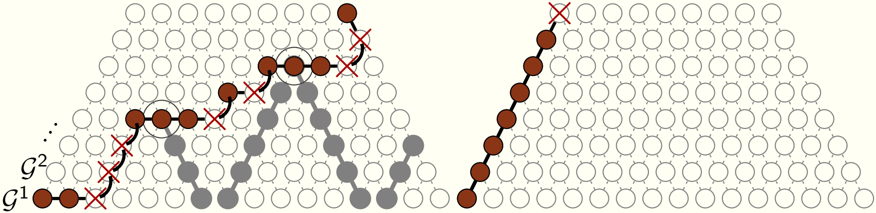
Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

Find the **smallest** value

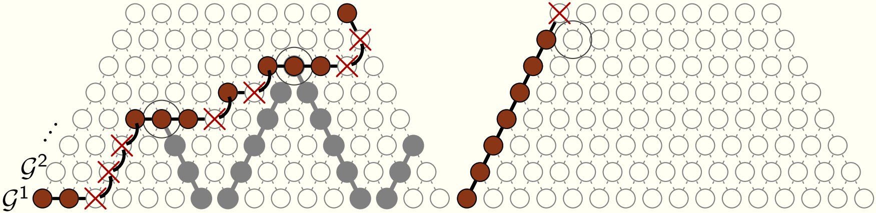
Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

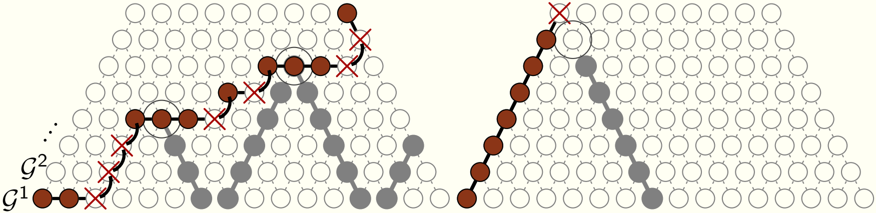
Find the **smallest** value — Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

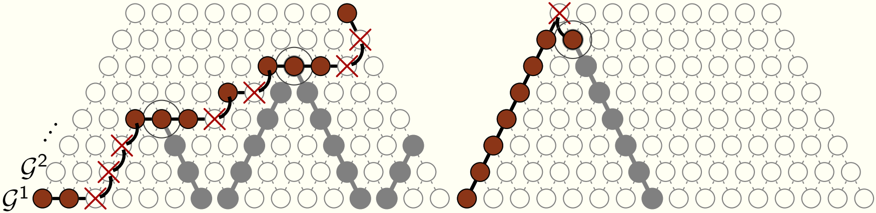
Find the **smallest** value — Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

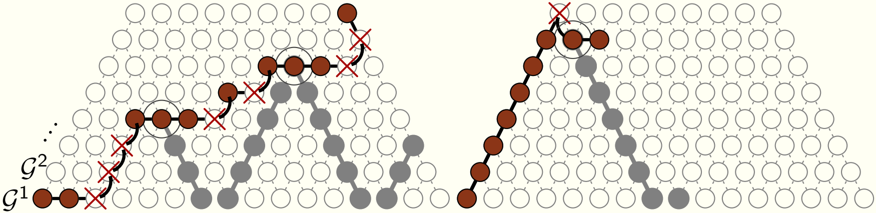
Find the **smallest** value — Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

Find the **smallest** value

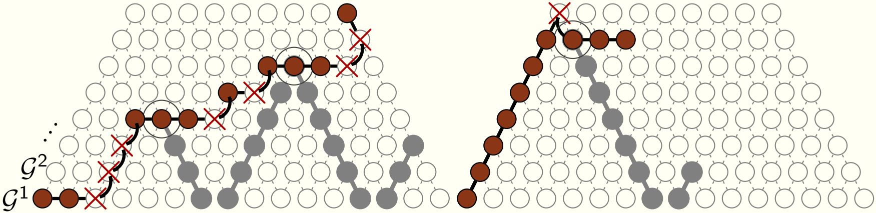
Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

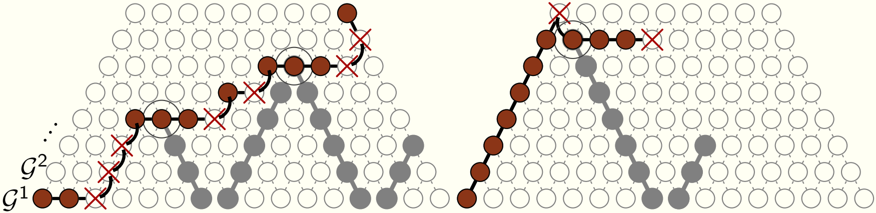
Find the **smallest** value — Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

Find the **smallest** value

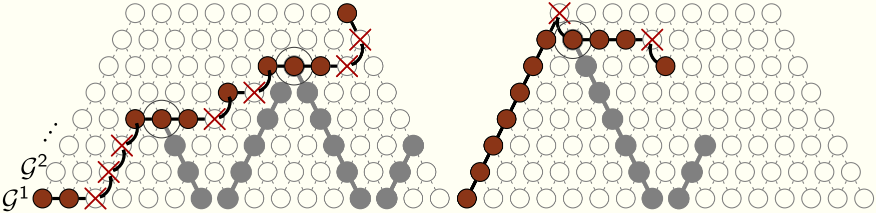
Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

Find the **smallest** value

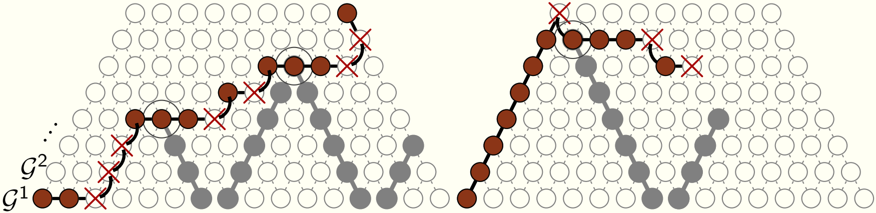
Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

Find the **smallest** value

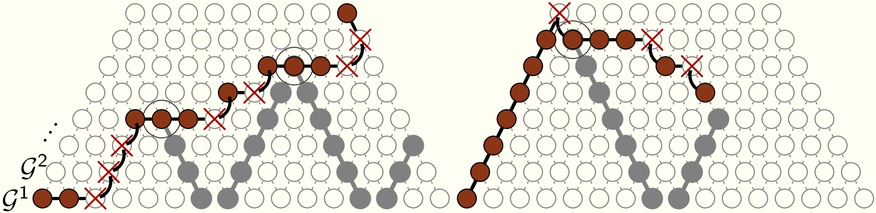
Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

Find the **smallest** value

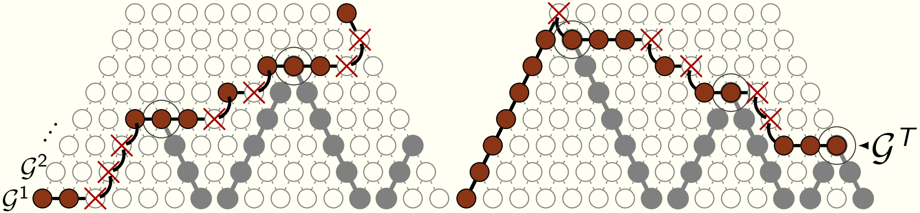
Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

Find the **smallest** value

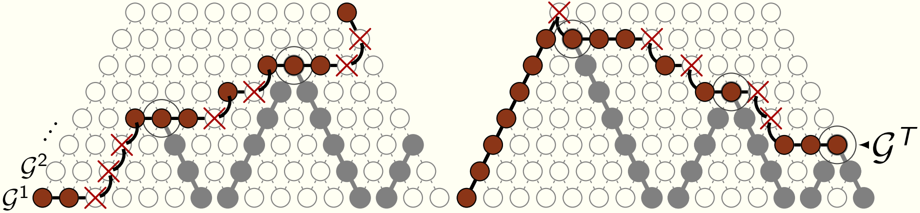
Find the **largest** value

Framework generalization

- Transitive closures concatenation → **Composition operation**
- Completeness test → **Test operation**
- Transitive closure → **Super node**

A Generic Framework

- Solve other problems using the same framework



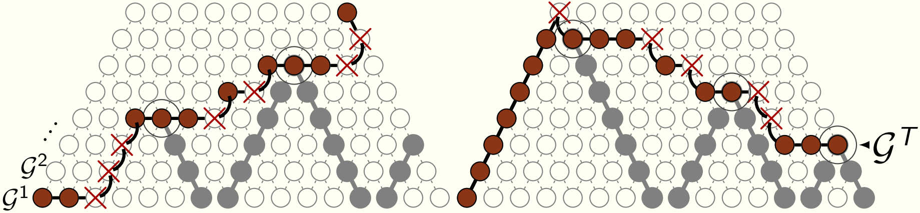
Minimization problems V.S **Maximization problems**

Find the **smallest** value

Find the **largest** value

A Generic Framework

- Solve other problems using the same framework



Minimization problems V.S **Maximization problems**

Find the **smallest** value

Find the **largest** value

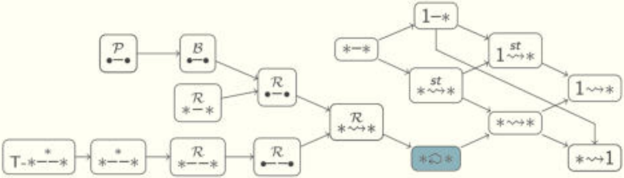
Requirements

- $test(G_{(i,j)}) = true \Leftrightarrow \{G_i, G_{i+1}, \ldots, G_j\}$ satisfies the property P
- The composition operation is associative
- Only minimization: If $test(G_{(i,j)}) = true$ then $test(G_{(i',j')}) = true, \forall i' \leq i, j' \geq j$
- Only maximization: If $test(G_{(i,j)}) = true$ then $test(G_{(i',j')}) = true, \forall i' \geq i, j' \leq j$

Round-trip Temporal Connectivity

Round-trip Temporal Connectivity

A dynamic graph $\mathcal{G}$ is round-trip temporal connected if and only if a back-and-forth journey exists from any node to all other nodes.



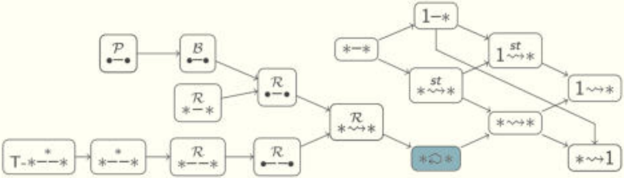
Round-trip Temporal Connectivity

Round-trip Temporal Connectivity

A dynamic graph $\mathcal{G}$ is round-trip temporal connected if and only if a back-and-forth journey exists from any node to all other nodes.

$\textsc{Round-Trip-Temporal-Diameter}$(minimization)

Finding the smallest duration in which there exists a back-and-forth journey from any node to all other nodes.



Round-trip Temporal Connectivity

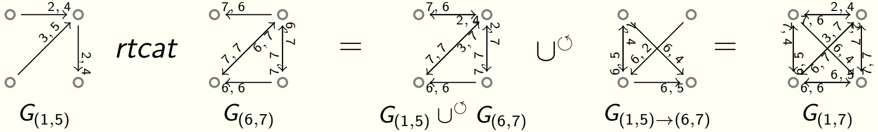
Round-trip Temporal Connectivity

A dynamic graph $\mathcal{G}$ is round-trip temporal connectivity if and only if a back-and-forth journey exists from any node to all other nodes.

Round-Trip-Temporal-Diameter(minimization)

Finding the smallest duration in which there exists a back-and-forth journey from any node to all other nodes.

- **Super node:** Round-trip transitive closure
- **Composition operation:** Round-trip transitive closure concatenation

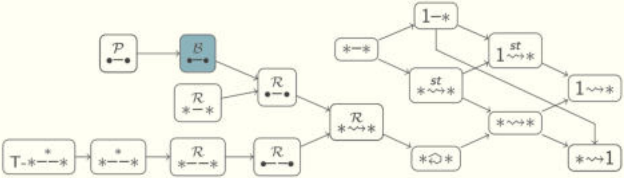


- **Test operation:** Round-trip completeness

Bounded Realization of the footprint

Time-bounded edge reappearance

A dynamic graph $\mathcal{G}$ has a time-bounded edge reappearance with a bound b if the time between two appearances of the same edge is at most b.



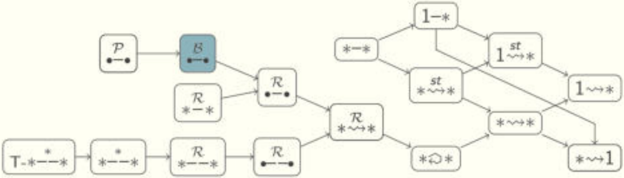
Bounded Realization of the footprint

Time-bounded edge reappearance

A dynamic graph $\mathcal{G}$ has a time-bounded edge reappearance with a bound b if the time between two appearances of the same edge is at most b.

BOUNDED-REALIZATION-OF-THE-FOOTPRINT(minimization)

Finding the smallest b such that in every subsequence of length b in the sequence $\mathcal{G}$, all the edges of the footprint appear at least once.



Bounded Realization of the footprint

Time-bounded edge reappearance

A dynamic graph $\mathcal{G}$ has a time-bounded edge reappearance with a bound b if the time between two appearances of the same edge is at most b.

Bounded-Realization-of-the-Footprint(minimization)

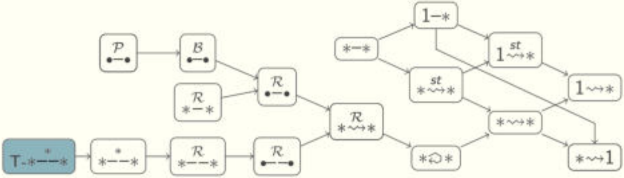
Finding the smallest b such that in every subsequence of length b in the sequence $\mathcal{G}$, all the edges of the footprint appear at least once.

- **Super node:** Union graphs
- **Composition operation:** Union
- **Test operation:** Equality to the footprint

T-interval Connectivity

Definition: T-interval connectivity

A dynamic graph $\mathcal{G}$ is T-interval connected if and only if every T length sequence of graphs has a common connected spanning sub-graph.



T-interval Connectivity

Definition: T-interval connectivity

A dynamic graph $\mathcal{G}$ is T-interval connected if and only if every T length sequence of graphs has a common connected spanning sub-graph.

T-interval Connectivity

Definition: T-interval connectivity

A dynamic graph $\mathcal{G}$ is T-interval connected if and only if every T length sequence of graphs has a common connected spanning sub-graph.

T-INTERVAL-CONNECTIVITY (maximization)

Finding the largest T for which the graph is T-interval connected.

$\mathcal{G}$ 

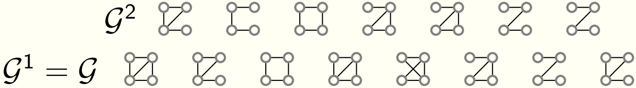
T-interval Connectivity

Definition: T-interval connectivity

A dynamic graph $\mathcal{G}$ is T-interval connected if and only if every T length sequence of graphs has a common connected spanning sub-graph.

T-INTERVAL-CONNECTIVITY (maximization)

Finding the largest T for which the graph is T-interval connected.



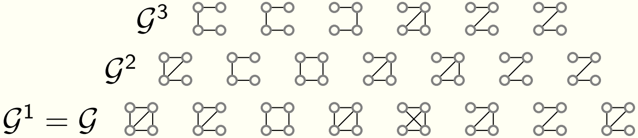
T-interval Connectivity

Definition: T-interval connectivity

A dynamic graph $\mathcal{G}$ is T-interval connected if and only if every T length sequence of graphs has a common connected spanning sub-graph.

T-INTERVAL-CONNECTIVITY (maximization)

Finding the largest T for which the graph is T-interval connected.



- **Super node:** Intersection graph
- **Composition operation:** Intersection
- **Test operation:** Connectivity test

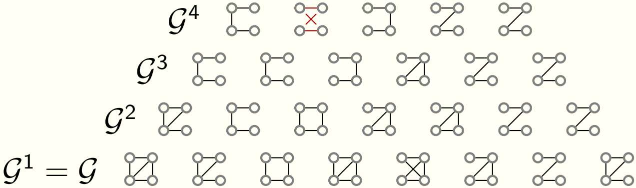
T-interval Connectivity

Definition: T-interval connectivity

A dynamic graph $\mathcal{G}$ is T-interval connected if and only if every T length sequence of graphs has a common connected spanning sub-graph.

T-INTERVAL-CONNECTIVITY (maximization)

Finding the largest T for which the graph is T-interval connected.



- **Super node:** Intersection graph
- **Composition operation:** Intersection
- **Test operation:** Connectivity test

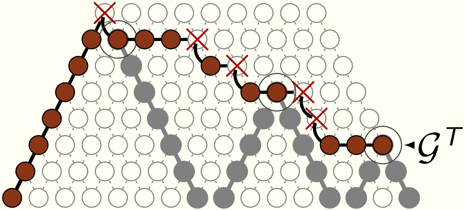
T-interval Connectivity

Definition: T-interval connectivity

A dynamic graph $\mathcal{G}$ is T-interval connected if and only if every T length sequence of graphs has a common connected spanning sub-graph.

T-Interval-Connectivity (maximization)

Finding the largest T for which the graph is T-interval connected.



Symmetric Problems

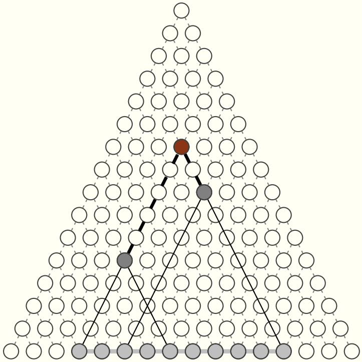
Symmetric Problems

A minimization or maximization problem is symmetric if:
for all $i, j, i', j' \leq \delta$, $i \leq i' \leq j$, $composition(G_{(i,j)}, G_{(i',j')}) = G_{(i,j')}$.

Symmetric Problems

Symmetric Problems

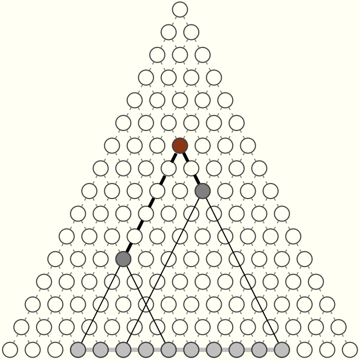
A minimization or maximization problem is symmetric if:
for all $i, j, i', j' \leq \delta$, $i \leq i' \leq j$, *composition*$(G_{(i,j)}, G_{(i',j')}) = G_{(i,j')}$.



Symmetric Problems

Symmetric Problems

A minimization or maximization problem is symmetric if:
for all $i, j, i', j' \leq \delta$, $i \leq i' \leq j$, $composition(G_{(i,j)}, G_{(i',j')}) = G_{(i,j')}$.

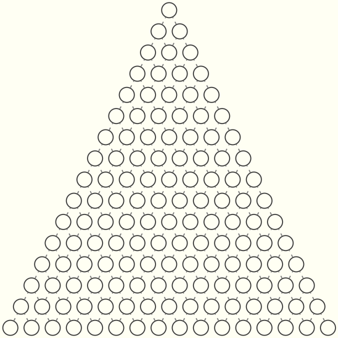


- e.g T-Interval-Connectivity and Bounded-Realization-of-the-Footprint

Row-Based Strategy

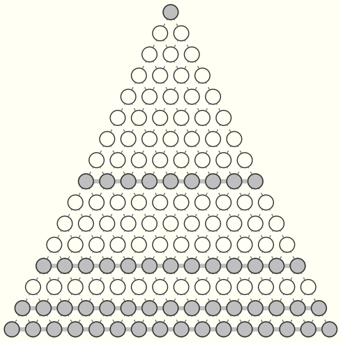
Row-Based Strategy

Symmetric problems (maximization)



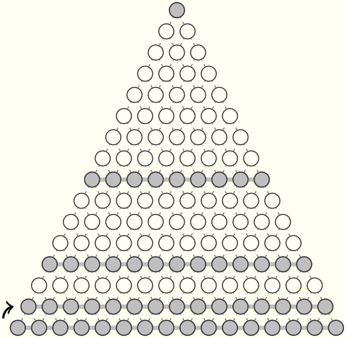
Row-Based Strategy

Symmetric problems (maximization)



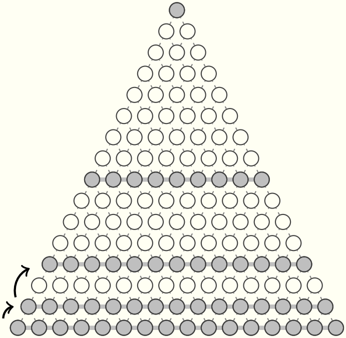
Row-Based Strategy

Symmetric problems (maximization)



Row-Based Strategy

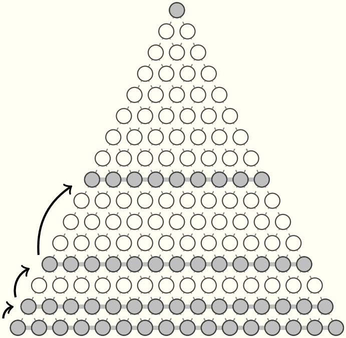
Symmetric problems (maximization)



Row-Based Strategy

Symmetric problems (maximization)

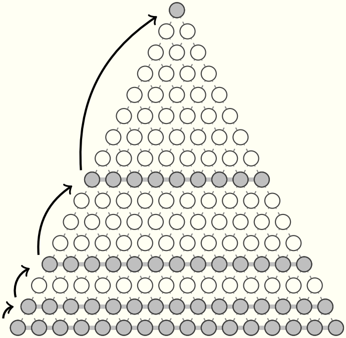
- $O(\delta)$ composition per row
- $O(\delta)$ tests per row



Row-Based Strategy

Symmetric problems (maximization)

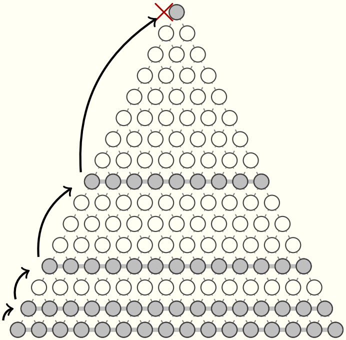
- $O(\delta)$ composition per row
- $O(\delta)$ tests per row



Row-Based Strategy

Symmetric problems (maximization)

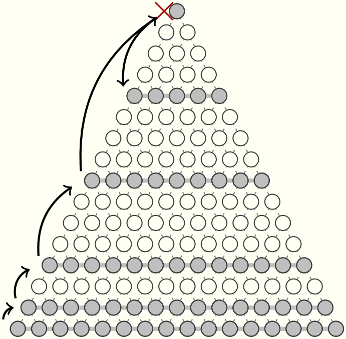
- $O(\delta)$ composition per row
- $O(\delta)$ tests per row



Row-Based Strategy

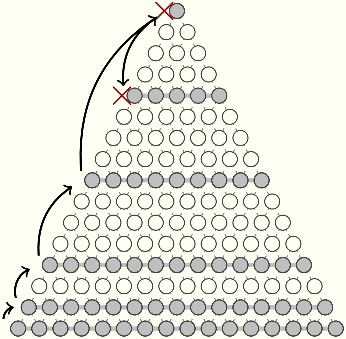
Symmetric problems (maximization)

- $O(\delta)$ composition per row
- $O(\delta)$ tests per row



Row-Based Strategy

Symmetric problems (maximization)

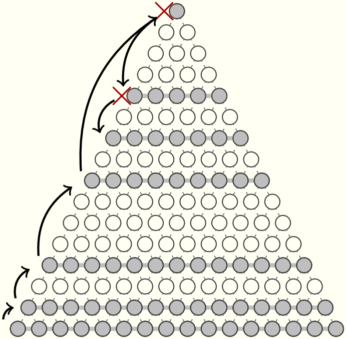


- $O(\delta)$ composition per row
- $O(\delta)$ tests per row

Row-Based Strategy

Symmetric problems (maximization)

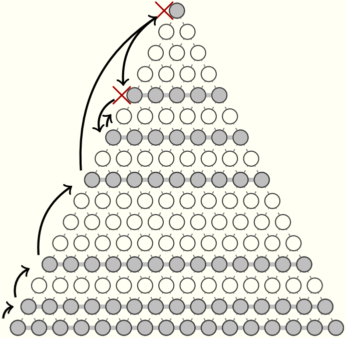
- $O(\delta)$ composition per row
- $O(\delta)$ tests per row



Row-Based Strategy

Symmetric problems (maximization)

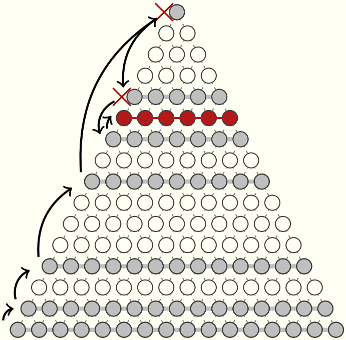
- $O(\delta)$ composition per row
- $O(\delta)$ tests per row



Row-Based Strategy

Symmetric problems (maximization)

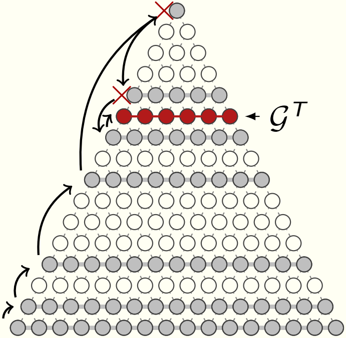
- $O(\delta)$ composition per row
- $O(\delta)$ tests per row



Row-Based Strategy

Symmetric problems (maximization)

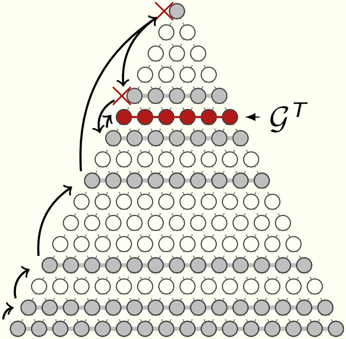
- $O(\delta)$ composition per row
- $O(\delta)$ tests per row



Row-Based Strategy

Symmetric problems (maximization)

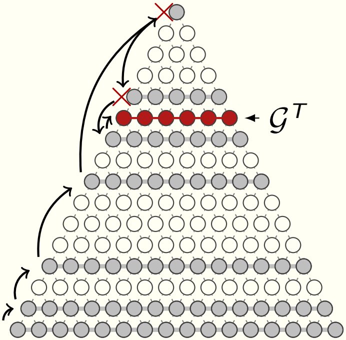
- $O(\delta)$ composition per row
- $O(\delta)$ tests per row
- $O(\log \delta)$ rows



Row-Based Strategy

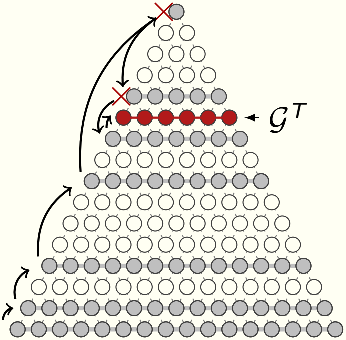
Symmetric problems (maximization)
$O(\delta \log \delta)$ **elementary operations**

- $O(\delta)$ composition per row
- $O(\delta)$ tests per row
- $O(\log \delta)$ rows



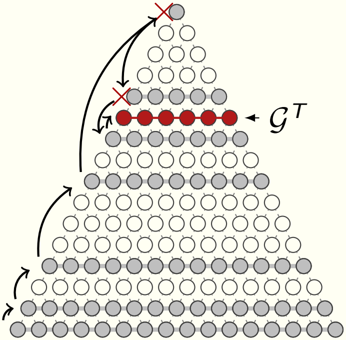
Parallel Version

- On EREW PRAM



Parallel Version

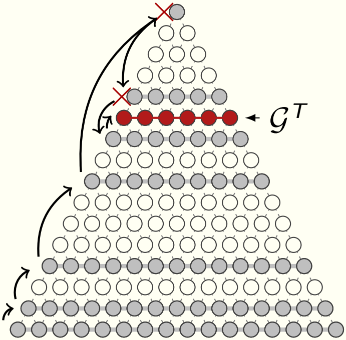
- On EREW PRAM



Parallel Version

- On EREW PRAM

  - Symmetric problems are solvable in $O(\log^2 \delta)$ on an **EREW PRAM** with $O(\delta)$ processors



Conclusion

- Conclusion
  - High-level strategies for computing minimization and maximization parameters
  - Algorithms that use only $O(\delta)$ elementary operations
  - Parallel versions on PRAM (in Nick's class)
  - Online algorithms with amortized cost of $O(1)$ elementary operations per graph received
- Perspectives
  - How about other classes?
  - Generic Framework
    - What if the evolution of the dynamic graph is constrained?

Thank you !