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?

Images of various objects including a smartphone, car, satellite, robot, drone, and network symbol.
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

<table>
<thead>
<tr>
<th>G₁</th>
<th>G₂</th>
<th>G₃</th>
<th>G₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>b</td>
<td>d</td>
<td>b</td>
<td>d</td>
</tr>
<tr>
<td>e</td>
<td>e</td>
<td>e</td>
<td>e</td>
</tr>
</tbody>
</table>

Dynamic graphs classes: [Casteigts, Flocchini, Quattrociocchi et Santoro, 2011]
Dynamic graphs: Various forms: TVG, Evolving graphs

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

\[ G = (V, E_i) \]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td>e</td>
<td>d</td>
<td>b</td>
</tr>
</tbody>
</table>

\[ G_1 \]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td>e</td>
<td>d</td>
<td>b</td>
</tr>
</tbody>
</table>

\[ G_2 \]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td>e</td>
<td>d</td>
<td>b</td>
</tr>
</tbody>
</table>

\[ G_3 \]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>e</td>
<td>c</td>
<td>d</td>
<td>b</td>
</tr>
</tbody>
</table>

\[ G_4 \]

\[ G \text{ is temporally connected} \iff \text{Transitive closure } G^* \text{ is complete} \]

Transitive closure of the journeys: reachability over time [Bhadra and Ferreira, 2003]
Temporal Connectivity

\[ \mathcal{G} = (V, E_i) \]

\[ \begin{align*}
G_1 & : & a & \rightarrow c \rightarrow e
\end{align*} \]

\[ \begin{align*}
G_2 & : & a & \rightarrow c \rightarrow d \rightarrow e
\end{align*} \]

\[ \begin{align*}
G_3 & : & a & \rightarrow c \rightarrow d \\
G_4 & : & a & \rightarrow b \rightarrow d
\end{align*} \]

**Temporal connectivity** ⇔ ∀u, v ∈ V, u ∼ v.
Temporal Connectivity

\[ \mathcal{G} = (V, E_i) \]

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

\[ \mathcal{G}^* \]

\[ \mathcal{G} \text{ is temporally connected} \iff \text{Transitive closure } \mathcal{G}^* \text{ is complete} \]
High-level Strategy

Temporal Connectivity

$G_{(1,4)}$
High-level Strategy

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

\[ G = G_1 \cup G_2 \cup G_3 \cup G_4 \]
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 $G$, 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 $G$, 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 $G$, 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 $G$, 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 $G$, 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 $G$, 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 $G$, i.e. the smallest duration in which there exists a journey from any node to all other nodes.

$\iff$

Finding the smallest $d$ such that every super node in row $G^d$ is a complete graph (i.e. every subsequence of length $d$ is temporally connected).
High-level Strategy

- Transitive closures
- Completeness test
- Transitive closures concatenation

$G^1 = G$

$G^2$

$G^3$

$G^4$

$G_1$

$G_2$

$G_3$

$G_4$

$G_{(1,2)}$

$G_{(1,3)}$

$G_{(2,3)}$

$G_{(3,4)}$

$G_{(1,4)}$
High-level Strategy

- Transitive closures
- Completeness test
- Transitive closures concatenation

\[
g^1 = g = G^4 \cup G^3 \cup G^2 \cup G^1 = G
\]

\[
G_{i,j} \quad cat \quad G_{i',j'} = G_{i,j} \cup G_{i',j'} \quad G_{i,j} \rightarrow (i',j')
\]
Temporal Diameter Computation

Decision version (given $d$)

$G^d$
Decision version (given d)
Decision version (given $d$)
Temporal Diameter Computation

Decision version (given $d$)

$$G^d_d(O)$$
Decision version (given $d$)
Temporal Diameter Computation

Decision version (given d)

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

$G_d^d$
Decision version (given $d$)

- A ladder of length $l$ costs $l - 1$ concatenation
Decision version (given d)

- A ladder of length \( l \) costs \( l - 1 \) concatenation
Decision version (given $d$)

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

$G^d$
Decision version (given $d$)

- A ladder of length $l$ costs $l - 1$ concatenation
- Use left and right ladders
Decision version (given $d$)

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

$G^d$
Decision version (given d)

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

$$G^d$$
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
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
Minimization version (find the temporal diameter $d$)
Minimization version (find the temporal diameter $d$)

Temporal-Diameter is solvable with $O(\delta)$ elementary operations
Minimization version (find the temporal diameter $d$)

Disjointness property: $\text{cat}(G(i,j), G(i',j')) = G(i, j')$

If $G(i,j)$ is complete, then $G(i',j')$ is complete, for all $i' \leq i$ and $j' \geq j$
Minimization version (find the temporal diameter \(d\))

Temporal-Diameter is solvable with \(O(\delta)\) elementary operations
Minimization version (find the temporal diameter $d$)
Minimization version (find the temporal diameter $d$)
Minimization version (find the temporal diameter $d$)
Minimization version (find the temporal diameter $d$)

- Strategy: ascending walk
Minimization version (find the temporal diameter \( d \))

- Strategy: ascending walk

Temporal-Diameter is solvable with \( O(\delta) \) elementary operations
Minimization version (find the temporal diameter $d$)

- Strategy: ascending walk
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

Temporal-Diameter is solvable with $O(\delta)$ elementary operations
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:** $\text{cat}(G_{(i,j)}, G_{(i',j')}) = G_{(i,j')}$
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:** $cat(G_{(i,j)}, G_{(i',j')}) = G_{(i,j')}$
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**: $\text{cat}(G_{(i,j)}, G_{(i',j')}) = G_{(i,j')}$
- 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:** $\text{cat}(G(i,j), G(i',j')) = G(i,j')$
- 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
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
Solve other problems using the same framework
Solve other problems using the same framework

Framework generalization

- Transitive closures concatenation
- Completeness test
- Transitive closure
Solve other problems using the same framework

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
Find the \textit{smallest} value

Framework generalization

| Transitive closures concatenation | → | Composition operation |
| Complement test | → | Test operation |
| Transitive closure | → | Super node |
A Generic Framework

- Solve other problems using the same framework

Minimization problems
Find the smallest value

Maximization problems
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
Find the smallest value

Maximization problems
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 (Find the smallest value) V.S Maximization problems (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 vs. Maximization problems

Find the smallest value vs. 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
Find the smallest value

Maximization problems
Find the largest value

Framework generalization

- Transitive closures concatenation → Composition operation
- Completeness test → Test operation
- Transitive closure → Super node
Solve other problems using the same framework

Minimization problems
Find the smallest value

Maximization problems
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
Find the smallest value

Maximization problems
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
Find the smallest value

Maximization problems
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

- Find the smallest value

Maximization problems

- Find the largest value

Framework generalization

- Transitive closures concatenation → Composition operation
- Completeness test → Test operation
- Transitive closure → Super node
Solve other problems using the same framework

Minimization problems
Find the smallest value

Maximization problems
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
Find the **smallest** value

Maximization problems
Find the **largest** 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
Find the smallest value

V.S

Maximization problems
Find the largest value

Framework generalization

- Transitive closures concatenation → Composition operation
- Completeness test → Test operation
- Transitive closure → Super node
Solve other problems using the same framework

Minimization problems
Find the smallest value

Maximization problems
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 vs Maximization problems

- Minimization problems: Find the smallest value
- Maximization problems: Find the largest 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
Find the \textit{smallest} value

Maximization problems
Find the \textit{largest} 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
Find the \textit{smallest} value

Maximization problems
Find the \textit{largest} value

Framework generalization

- Transitive closures concatenation \rightarrow \textit{Composition operation}
- Completeness test \rightarrow \textit{Test operation}
- Transitive closure \rightarrow \textit{Super node}
A Generic Framework

- Solve other problems using the same framework

Minimization problems  \( \text{V.S} \)  Maximization problems
Find the \textit{smallest} value  Find the \textit{largest} 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
- Find the smallest value

Maximization problems
- 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: Find the smallest value

Maximization problems: Find the largest value
A Generic Framework

- Solve other problems using the same framework

Minimization problems
Find the **smallest** value

Maximization problems
Find the **largest** value

**Requirements**

- \( test(G_{(i,j)}) = true \iff \{G_i, G_{i+1}, \ldots, G_j\} \) 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

A dynamic graph $G$ 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

A dynamic graph $G$ 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-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

A dynamic graph $G$ 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
Time-bounded edge reappearance

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

A dynamic graph $G$ 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 $G$, all the edges of the footprint appear at least once.
Bounded Realization of the footprint

Time-bounded edge reappearance

A dynamic graph $G$ 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 $G$, all the edges of the footprint appear at least once.

- **Super node:** Union graphs
- **Composition operation:** Union
- **Test operation:** Equality to the footprint
Definition: \( T \)-interval connectivity

A dynamic graph \( G \) is \( T \)-interval connected if and only if every \( T \) length sequence of graphs has a common connected spanning sub-graph.
Definition: $T$-interval connectivity

A dynamic graph $G$ is $T$-interval connected if and only if every $T$ length sequence of graphs has a common connected spanning sub-graph.
Definition: \( T \)-interval connectivity

A dynamic graph \( G \) 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.

\[ G = \{ G_1, G_2, G_3, G_4 \} \]

Super node: Intersection graph

Composition operation: Intersection

Test operation: Connectivity test
**Definition: T-interval connectivity**

A dynamic graph $G$ 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 $G$ 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
**Definition: T-interval connectivity**

A dynamic graph $G$ 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.

$G^4$  $G^3$  $G^2$  $G^1 = G$

- **Super node:** Intersection graph
- **Composition operation:** Intersection
- **Test operation:** Connectivity test
**Definition: T-interval connectivity**

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

For example, T-Interval-Connectivity and Bounded-Realization-of-the-Footprint.
A minimization or maximization problem is symmetric if:
for all \( i, j, i', j' \leq \delta \), \( i \leq i' \leq j \), \( \text{composition}(G(i,j), G(i',j')) = G(i,j') \).
A minimization or maximization problem is symmetric if:
for all \(i, j, i', j' \leq \delta\), \(i \leq i' \leq j\),
\(\text{composition}(G_{(i,j)}, G_{(i',j')}) = G_{(i,j')}\).

\(\text{e.g. T-Interval-Connectivity and Bounded-Realization-of-the-Footprint}\)
Row-Based Strategy
Row-Based Strategy

Symmetric problems (maximization)
Row-Based Strategy

Symmetric problems (maximization)
Row-Based Strategy

Symmetric problems (maximization)
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
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
Symmetric problems are solvable in $O(\log_2 \delta)$ on an EREW PRAM with $O(\delta)$ processors.
Parallel Version

- On EREW PRAM
On EREW PRAM

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