Algorithms for Algebraic Path Properties in Concurrent Systems of Constant Treewidth Components

Krishnendu Chatterjee, Amir Kafshdar Goharshady, Rasmus Ibsen-Jensen, Andreas Pavlogiannis

POPL 2016
A typical paradigm in program analysis is to reduce the problem to a standard graph problem $P$:

**Input:** Program

1. Extract control flow graph $G$
2. Annotate $G$
3. Run best general graph algorithm for $P$ on $G$
A typical paradigm in program analysis is to reduce the problem to a standard graph problem \( P \):

**Input:** Program

1. Extract control flow graph \( G \)
2. Annotate \( G \)
3. **Run best general graph algorithm for** \( P \) **on** \( G \)
   - by exploiting special structure of CFGs
Static dataflow/quantitative analysis of concurrent systems
System consists of CFGs of local threads + sync vars
A node of the concurrent system specifies the local state of each thread (+ sync)
System transitions wrt interleaving semantics
Concurrent system annotated with a (complete, closed) semiring.

Variety of properties expressible

- (Generalized) reachability
- Distributive dataflow analysis problems
- Quantitative problems (quality measures / quantitative verification)
- Algebraic relaxations for interprocedural analysis
Concurrent system annotated with a (complete, closed) semiring.

Variety of properties expressible

- (Generalized) reachability
- Distributive dataflow analysis problems
- Quantitative problems (quality measures / quantitative verification)
- Algebraic relaxations for interprocedural analysis
Formal Setting

- Complete, closed \textbf{semiring} \( S = (\Sigma, \oplus, \otimes, \overline{0}, \overline{1}) \)
- \( k \) local graphs \( G_i = (V_i, E_i), |V_i| \leq n \)
- Compose a \textbf{concurrent system} \( G = (V, E, \text{wt}) \)
  - Nodes of the form \( v = \langle v_1, \ldots, v_k \rangle \)
  - \( E \subseteq \text{product}(E_1, \ldots, E_k) \)
    - i.e. \( (\langle u_1, \ldots, u_k \rangle, \langle v_1, \ldots, v_k \rangle) \in E \)
  - Global weight function \( \text{wt} : E \to \Sigma \)
- \textbf{Weight} of a path \( P : x_1, x_2, \ldots, x_m : \)

\[ \otimes(P) = \text{wt}(x_1, x_2) \otimes \text{wt}(x_2, x_3) \cdots \otimes \text{wt}(x_{k-1}, x_m) \]
Formal Setting

- Complete, closed semiring $S = (\Sigma, \oplus, \otimes, \overline{0}, \overline{1})$
- $k$ local graphs $G_i = (V_i, E_i), \; |V_i| \leq n$
- Compose a concurrent system $G = (V, E, wt)$
  - Nodes of the form $v = \langle v_1, \ldots, v_k \rangle$
  - $E \subseteq \text{product}(E_1, \ldots, E_k)$
    - i.e. $(\langle u_1, \ldots, u_k \rangle, \langle v_1, \ldots, v_k \rangle) \in E$
  - Global weight function $wt : E \rightarrow \Sigma$

Weight of a path $P : x_1, x_2, \ldots, x_m$:

$$\otimes(P) = wt(x_1, x_2) \otimes wt(x_2, x_3) \cdots \otimes wt(x_{k-1}, x_m)$$
Formal Setting

- Complete, closed **semiring** $S = (\Sigma, \oplus, \otimes, \overline{0}, \overline{1})$
- $k$ local graphs $G_i = (V_i, E_i)$, $|V_i| \leq n$
- Compose a **concurrent system** $G = (V, E, wt)$
  - Nodes of the form $v = \langle v_1, \ldots, v_k \rangle$
  - $E \subseteq \text{product}(E_1, \ldots, E_k)$
    - i.e. $(\langle u_1, \ldots, u_k \rangle, \langle v_1, \ldots, v_k \rangle) \in E$
  - Global weight function $wt : E \to \Sigma$
- **Weight** of a path $P : x_1, x_2, \ldots, x_m$:

$$\otimes(P) = wt(x_1, x_2) \otimes wt(x_2, x_3) \cdots \otimes wt(x_{k-1}, x_m)$$
Graph Problem: Semiring Distances

Weight of a path $P : x_1, x_2, \ldots, x_m$:

$$\otimes(P) = \text{wt}(x_1, x_2) \otimes \text{wt}(x_2, x_3) \cdots \otimes \text{wt}(x_{k-1}, x_m)$$

**Semiring distance** from $u$ to $v$:

$$d(u, v) = \bigoplus_{P : u \rightsquigarrow v} \otimes(P)$$
Method 1: Thread 1

1 while 1 do
2   if turn = −1 then
3     lock(ℓ)
4     turn ← my_id
5     unlock(ℓ)
6   if turn = my_id then
7     /* do stuff */
8     turn ← −1
9 end

Method 2: Thread 2

1 while 1 do
2   if turn = −1 then
3     lock(ℓ)
4     turn ← my_id
5     unlock(ℓ)
6   if turn = my_id then
7     /* do stuff */
8     turn ← −1
9 end
Method 1: Thread 1
1 while 1 do
2     if turn = −1 then
3         lock(ℓ)
4         turn ← my_id
5         unlock(ℓ)
6     end
7     if turn = my_id then /* do stuff */
8         turn ← −1
9 end

Method 2: Thread 2
1 while 1 do
2     if turn = −1 then
3         lock(ℓ)
4         turn ← my_id
5         unlock(ℓ)
6     end
7     if turn = my_id then /* do stuff */
8         turn ← −1
9 end

Nodes V: (2, 7)
Nodes $V$: $\langle 2, 7 \rangle$

Edges $E$: $(\langle 2, 7 \rangle, \langle 6, 8 \rangle)$

Method 1: Thread 1

1 while 1 do
2 if turn = $-1$ then
3 lock($\ell$)
4 turn $\leftarrow$ my_id
5 unlock($\ell$)
6 if turn = my_id then /* do stuff */
7 turn $\leftarrow -1$
8 end
9 end

Method 2: Thread 2

1 while 1 do
2 if turn = $-1$ then
3 lock($\ell$)
4 turn $\leftarrow$ my_id
5 unlock($\ell$)
6 if turn = my_id then /* do stuff */
7 turn $\leftarrow -1$
8 end
9 end
**Method 1: Thread 1**

1. while 1 do
2.   if turn = $-1$ then
3.     lock($\ell$)
4.     turn $\leftarrow$ my_id
5.     unlock($\ell$)
6.   if turn = my_id then
7.     /* do stuff */
8.   turn $\leftarrow -1$
9. end

**Method 2: Thread 2**

1. while 1 do
2.   if turn = $-1$ then
3.     lock($\ell$)
4.     turn $\leftarrow$ my_id
5.     unlock($\ell$)
6.   if turn = my_id then
7.     /* do stuff */
8.   turn $\leftarrow -1$
9. end

Nodes $V$: $\langle 2, 7 \rangle$

Edges $E$: $(\langle 2, 7 \rangle, \langle 6, 8 \rangle)$

Weights $wt : E \rightarrow \Sigma : wt(\langle 2, 7 \rangle, \langle 6, 8 \rangle) = \alpha$
Nodes $V$: $\langle 2, 7 \rangle$

Edges $E$: $(\langle 2, 7 \rangle, \langle 6, 8 \rangle)$

Weights $wt : E \to \Sigma : wt(\langle 2, 7 \rangle, \langle 6, 8 \rangle) = \alpha$

Pair query: $d(\langle 1, 1 \rangle, \langle 7, 7 \rangle)$
Existing Algorithmic Approach

- Construct the product graph $G$ from the local components $G_1, G_2$
- Compute transitive closure (all-pairs) on $G$
  - Warshall-Floyd-Kleene algorithm, cubic complexity
  - $O\left( (n^2)^3 \right) = O\left( n^6 \right)$
Existing Algorithmic Approach

- Construct the product graph $G$ from the local components $G_1, G_2$
- Compute transitive closure (all-pairs) on $G$
  - Warshall-Floyd-Kleene algorithm, cubic complexity
  - $O \left( (n^2)^3 \right) = O(n^6)$
Component graphs have special structure, i.e., low-treewidth graphs

- Measures similarity of a graph to a tree
- Well-known property of control-flow graphs

A faster algorithm for the transitive closure $O(n^6) \rightarrow O(n^{4+\epsilon})$
Our Improvements for Semiring Distances

- Component graphs have special structure, i.e., low-treewidth graphs
  - Measures similarity of a graph to a tree
  - Well-known property of control-flow graphs

\[ O(n^6) \rightarrow O(n^{4+\epsilon}) \]
Our Improvements for Semiring Distances

- On demand analysis: *preprocess vs query*

Preprocessing spectrum

- No Preprocessing
- Transitive Closure

- Answering a few queries does not require the transitive closure
- On demand analysis: preprocess more only if expecting many queries
Our Improvements for Semiring Distances

- On demand analysis: \textit{preprocess} vs \textit{query}

Preprocessing spectrum

- No Preprocessing
- Transitive Closure

- Answering a few queries does not require the transitive closure
- On demand analysis: preprocess more only if expecting many queries
Our Improvements for Semiring Distances

- 2 components: product size $n \times n$, transitive closure has $n^4$ entries
- Existing $O(n^6)$ time for transitive closure

Preprocessing spectrum

No Preprocessing

Transitive Closure

- (1): Transitive closure almost optimal
- (3): Conditionally optimal
Our Improvements for Semiring Distances

- 2 components: product size $n \times n$, transitive closure has $n^4$ entries
- Existing $O(n^6)$ time for transitive closure

Preprocessing spectrum

<table>
<thead>
<tr>
<th>No Preprocessing</th>
<th>Transitive Closure</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(1) \ n^{4+\epsilon} + i$</td>
<td>$n^{3+\epsilon}$</td>
</tr>
</tbody>
</table>

- (1): Transitive closure almost optimal
- (3): Conditionally optimal
Our Improvements for Semiring Distances

- 2 components: product size $n \times n$, transitive closure has $n^4$ entries
- Existing $O(n^6)$ time for transitive closure

Preprocessing spectrum

\[
\begin{array}{c|c|c}
& (2) & (1) \\
\hline
n^3 + \epsilon + i \cdot n & n^4 + \epsilon + i \\
\hline
n^{1+\epsilon} & n^{3+\epsilon} \\
\end{array}
\]

No Preprocessing  \hspace{2cm}  Transitive Closure

- (1): Transitive closure almost optimal
- (3): Conditionally optimal
Our Improvements for Semiring Distances

- 2 components: product size $n \times n$, transitive closure has $n^4$ entries
- Existing $O(n^6)$ time for transitive closure

Preprocessing spectrum

<table>
<thead>
<tr>
<th>Preprocessing</th>
<th>Spectrum</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Preprocessing</td>
<td>$n^{1+\epsilon}$</td>
</tr>
<tr>
<td>Transitive Closure</td>
<td>$n^{3+\epsilon}$</td>
</tr>
</tbody>
</table>

(1) $n^{4+\epsilon} + i$
(2) $n^{3+\epsilon} + i \cdot n$
(3) $n^3 + i \cdot n^2$

- (1): Transitive closure almost optimal
- (3): Conditionally optimal
Our Improvements for Semiring Distances

- 2 components: product size $n \times n$, transitive closure has $n^4$ entries
- Existing $O(n^6)$ time for transitive closure

Preprocessing spectrum

| (3) $n^3 + i \cdot n^2$ | (2) $n^{3+\epsilon} + i \cdot n$ | (1) $n^{4+\epsilon} + i$ |

- $n^{1+\epsilon}$
- $n^{3+\epsilon}$

No Preprocessing | Transitive Closure

- (1): Transitive closure almost optimal
- (3): Conditionally optimal
Our Improvements for Semiring Distances

- 2 components: product size $n \times n$, transitive closure has $n^4$ entries
- Existing $O(n^6)$ time for transitive closure

Preprocessing spectrum

<table>
<thead>
<tr>
<th>(3) $n^3 + i \cdot n^2$</th>
<th>(2) $n^{3+\epsilon} + i \cdot n$</th>
<th>(1) $n^{4+\epsilon} + i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n^{1+\epsilon}$</td>
<td>$n^{3+\epsilon}$</td>
<td></td>
</tr>
</tbody>
</table>

No Preprocessing  
Transitive Closure

- (1): Transitive closure almost optimal
- (3): Conditionally optimal
Outline

- Tree decompositions
- Treewidth of the concurrent system
- On demand analysis on the concurrent tree-decomposition
- Experimental results
Definition (Tree decomposition)

Given a graph \( G = (V, E) \), a **tree-decomposition** \( \text{Tree}(G) = (V_T, E_T) \) is a tree of bags \( B_i \subseteq V \).
Given a graph $G = (V, E)$, a **tree-decomposition** $\text{Tree}(G) = (V_T, E_T)$ is a tree of bags $B_i \subseteq V$.
Tree Decompositions

Definition (Tree decomposition)

Given a graph $G = (V, E)$, a tree-decomposition $\text{Tree}(G) = (V_T, E_T)$ is a tree of bags $B_i \subseteq V$. 

![Diagram of a graph and its tree decomposition](https://via.placeholder.com/150)
Definition (Tree decomposition)

Given a graph $G = (V, E)$, a tree-decomposition $\text{Tree}(G) = (V_T, E_T)$ is a tree of bags $B_i \subseteq V$. 

$\text{G}$

$\text{Tree}(G)$
Definition (Tree decomposition)

Given a graph $G = (V, E)$, a **tree-decomposition** $\text{Tree}(G) = (V_T, E_T)$ is a tree of bags $B_i \subseteq V$.

\[ \exists x_1 \in B_1 \cap B_2 \]
\[ \exists x_2 \in B_2 \cap B_3 \]

\[ d(10,6) = d(10, x_1) \otimes d(x_1, x_2) \otimes d(x_2, 6) \]
Definition (Tree decomposition)

Given a graph $G = (V, E)$, a **tree-decomposition** $\text{Tree}(G) = (V_T, E_T)$ is a tree of bags $B_i \subseteq V$

$\exists x_1 \in B_1 \cap B_2$
$\exists x_2 \in B_2 \cap B_3$

$d(10, 6) = d(10, x_1) \otimes d(x_1, x_2) \otimes d(x_2, 6)$

Semiring distances reduce to:
1. Tree decomposition
2. Local Distances
CFGs of typical imperative programs have tree-decompositions of small sized bags

- Theoretically, for goto-free programs
  - Pascal $\leq 4$
  - C $\leq 7$
- In practice small in imperative programs (e.g. Java $\leq 8$)
CFGs of typical imperative programs have tree-decompositions of small sized bags

- Theoretically, for goto-free programs
  - Pascal $\leq 4$
  - C $\leq 7$
- In practice small in imperative programs (e.g. Java $\leq 8$)

**Theorem (Tree decomposition)**

*For constant treewidth graphs, $\text{Tree}(G)$ can be constructed in $O(n)$ time.*
Treewidth of the Concurrent System

Components of small treewidth can yield a concurrent system $G$ of very large treewidth!

Computing an optimal tree decomposition of $G$ is intractable (NP-C)
Strongly Balanced Tree Decompositions

Convert the local tree decompositions to

- **Binary**: every bag has two children
- **Strongly balanced**: most bags have two subtrees of approximately equal size
State the local tree decompositions to

- **Binary:** every bag has two children
- **Strongly balanced:** most bags have two subtrees of approximately equal size
Convert the local tree decompositions to

- **Binary**: every bag has two children
- **Strongly balanced**: most bags have two subtrees of approximately equal size
Strongly Balanced Tree Decompositions

Convert the local tree decompositions to

- **Binary**: every bag has two children
- **Strongly balanced**: most bags have two subtrees of approximately equal size

\[
\text{Tree}(G_1) \quad \rightarrow \quad \text{Strongly Balanced Tree}(G_1)
\]
Tree Decomposition of the Concurrent System

$t = O(1)$

Tree($G_1$)

Tree($G_2$)

Tree($G$)
For every two nodes $u, v$ appearing in a bag, compute $d(u, v)$

Tree($G$)

- Transitive closure on the bags instead of the whole system
- Cost decreases geometrically on the levels
  - Transitive closure on the root dominates
  - $O(n^6) \rightarrow O(n^3)$
For every two nodes $u, v$ appearing in a bag, compute $d(u, v)$

Tree($G$)

- Transitive closure on the bags instead of the whole system
- Cost decreases geometrically on the levels
  - Transitive closure on the root dominates
  - $O(n^6) \rightarrow O(n^3)$
For every two nodes $u, v$ appearing in a bag, compute $d(u, v)$

- Transitive closure on the bags instead of the whole system
- Cost decreases geometrically on the levels
  - Transitive closure on the root dominates
  - $O(n^6) \rightarrow O(n^3)$
For every two nodes $u, v$ appearing in a bag, compute $d(u, v)$

Tree($G$)

- Transitive closure on the bags instead of the whole system
- Cost decreases geometrically on the levels
  - Transitive closure on the root dominates
  - $O(n^6) \rightarrow O(n^3)$
For every two nodes $u, v$ appearing in a bag, compute $d(u, v)$

Tree($G$)

- Transitive closure on the bags instead of the whole system
- Cost decreases geometrically on the levels
  - Transitive closure on the root dominates
  - $O(n^6) \rightarrow O(n^3)$
For every two nodes $u, v$ appearing in a bag, compute $d(u, v)$

$$\text{Tree}(G)$$

- Transitive closure on the bags instead of the whole system
- Cost decreases geometrically on the levels
  - Transitive closure on the root dominates
  - $O(n^6) \rightarrow O(n^3)$
For every two nodes $u, v$ appearing in a bag, compute $d(u, v)$

Tree($G$)

- Transitive closure on the bags instead of the whole system
- Cost decreases geometrically on the levels
  - Transitive closure on the root dominates
  - $O(n^6) \rightarrow O(n^3)$
For every two nodes $u, v$ appearing in a bag, compute $d(u, v)$

Tree($G$)

- Transitive closure on the bags instead of the whole system
- Cost decreases geometrically on the levels
  - Transitive closure on the root dominates
  - $O(n^6) \rightarrow O(n^3)$
For every two nodes \(u, v\) appearing in a bag, compute \(d(u, v)\)

Tree(\(G\))

- Transitive closure on the bags instead of the whole system
- Cost decreases geometrically on the levels
  - Transitive closure on the root dominates
  - \(O(n^6) \rightarrow O(n^3)\)
For every two nodes $u, v$ appearing in a bag, compute $d(u, v)$

Tree($G$)

- Transitive closure on the bags instead of the whole system
- Cost decreases geometrically on the levels
  - Transitive closure on the root dominates
  - $O(n^6) \rightarrow O(n^3)$
### Tradeoffs

#### Preprocessing spectrum

<table>
<thead>
<tr>
<th>No Preprocessing</th>
<th>Transitive Closure</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3) $n^3 + i \cdot n^2$</td>
<td>(1) $n^{4+\varepsilon} + i$</td>
</tr>
<tr>
<td>$n^{1+\varepsilon}$</td>
<td>$n^{3+\varepsilon}$</td>
</tr>
</tbody>
</table>
Experiments

- Control-flow graphs (CFGs) of methods from java libraries and benchmarks
- Focus on algorithmic comparison with standard transitive closure algorithms
- Used the tropical min-plus semiring on $\mathbb{R} \cup \{\infty\}$
Experiments 1

- CFGs from the DaCapo suit
- $n$ nodes each CFG
- 2-self product: size $n \times n$
- Random weights in $[-10^3, 10^3]$
- Compute the transitive closure
- Baseline: Bellman-Ford
Experiments 1

- CFGs from the DaCapo suit
- $n$ nodes each CFG
- 2-self product: size $n \times n$

- Random weights in $[-10^3, 10^3]$
- Compute the transitive closure
- Baseline: Bellman-Ford
Experiments 1

- CFGs from the DaCapo suit
- $n$ nodes each CFG
- 2-self product: size $n \times n$
- Random weights in $[-10^3, 10^3]$
- Compute the transitive closure
- Baseline: Bellman-Ford
Experiments 2

- CFGs from container methods in java.util.concurrent
- Bloated with values of locks
- 2-self product: size $n \times n$

- Random weights in $[-10^3, 10^3]$
- Transitive closure
Experiments 2

- CFGs from container methods in java.util.concurrent
- Bloated with values of locks
- 2-self product: size $n \times n$
- Random weights in $[-10^3, 10^3]$
- Transitive closure
Experiments 2

- CFGs from container methods in `java.util.concurrent`
- Bloated with values of locks
- 2-self product: size $n \times n$
- Random weights in $[-10^3, 10^3]$
- Transitive closure
Thank you!
Questions?
Experiments 2

- CFGs from container methods in java.util.concurrent
- Bloated with values of locks
- 2-self product: size $n \times n$
- Random weights in $[-10^3, 10^3]$
- Transitive closure times:
  - $T_o(s)$: our algorithm
  - $T_b(s)$: baseline (Bellman-Ford)

<table>
<thead>
<tr>
<th>Java method</th>
<th>n</th>
<th>$T_o(s)$</th>
<th>$T_b(s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArrayBlockingQueue: poll</td>
<td>19</td>
<td>19</td>
<td>60</td>
</tr>
<tr>
<td>ArrayBlockingQueue: peek</td>
<td>20</td>
<td>20</td>
<td>81</td>
</tr>
<tr>
<td>LinkedBlockingDeque: advance</td>
<td>25</td>
<td>29</td>
<td>195</td>
</tr>
<tr>
<td>PriorityBlockingQueue: removeEQ</td>
<td>25</td>
<td>32</td>
<td>176</td>
</tr>
<tr>
<td>ArrayBlockingQueue: init</td>
<td>26</td>
<td>47</td>
<td>249</td>
</tr>
<tr>
<td>LinkedBlockingDeque: remove</td>
<td>26</td>
<td>49</td>
<td>290</td>
</tr>
<tr>
<td>ArrayBlockingQueue: offer</td>
<td>26</td>
<td>56</td>
<td>304</td>
</tr>
<tr>
<td>ArrayBlockingQueue: clear</td>
<td>28</td>
<td>33</td>
<td>389</td>
</tr>
<tr>
<td>ArrayBlockingQueue: contains</td>
<td>32</td>
<td>205</td>
<td>881</td>
</tr>
<tr>
<td>DelayQueue: remove</td>
<td>42</td>
<td>267</td>
<td>3792</td>
</tr>
<tr>
<td>ConcurrentHashMap: scanAndLockForPut</td>
<td>46</td>
<td>375</td>
<td>2176</td>
</tr>
<tr>
<td>ArrayBlockingQueue: next</td>
<td>46</td>
<td>407</td>
<td>3915</td>
</tr>
<tr>
<td>ConcurrentHashMap: put</td>
<td>72</td>
<td>1895</td>
<td>$&gt; 8$ h</td>
</tr>
</tbody>
</table>
### Results

#### 2 components

<table>
<thead>
<tr>
<th></th>
<th>Preprocess</th>
<th>Query time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Space</td>
</tr>
<tr>
<td>Existing</td>
<td>$O(n^6)$</td>
<td>$O(n^4)$</td>
</tr>
<tr>
<td><strong>Our result</strong></td>
<td>$O(n^3)$</td>
<td>$O(n^{2+\epsilon})$</td>
</tr>
<tr>
<td><strong>Our result</strong></td>
<td>$O(n^{3+\epsilon})$</td>
<td>$O(n^3)$</td>
</tr>
<tr>
<td><strong>Our result</strong></td>
<td>$O(n^{4+\epsilon})$</td>
<td>$O(n^4)$</td>
</tr>
</tbody>
</table>

#### $k \geq 3$ components

<table>
<thead>
<tr>
<th></th>
<th>Preprocess</th>
<th>Query time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Space</td>
</tr>
<tr>
<td>Existing</td>
<td>$O(n^{3k})$</td>
<td>$O(n^{2k})$</td>
</tr>
<tr>
<td><strong>Our result</strong></td>
<td>$O(n^{3k-3})$</td>
<td>$O(n^{2k-1})$</td>
</tr>
<tr>
<td><strong>Our result</strong></td>
<td>$O(n^{3k-2})$</td>
<td>$O(n^{2k})$</td>
</tr>
</tbody>
</table>

Chatterjee, Kafshdar Goharshady, Ibsen-Jensen, Pavlogiannis
2 components: product size $n \times n$, transitive closure has $n^4$ entries

Existing $O(n^6)$ time for transitive closure
Tree Decompositions

Definition (Tree decomposition)

Given a graph $G = (V, E)$, a **tree-decomposition** $\text{Tree}(G) = (V_T, E_T)$ is a **tree of bags** $B_i \subseteq V$ such that:

1. Every node of $G$ is contained in a bag
2. Every edge of $G$ is contained in a bag
3. Every node of $G$ appears in a contiguous subtree of $\text{Tree}(G)$.

\[
\begin{align*}
G & \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad
\end{align*}
\]
Definition (Tree decomposition)

Given a graph $G = (V, E)$, a **tree-decomposition** $\text{Tree}(G) = (V_T, E_T)$ is a tree of bags $B_i \subseteq V$ such that:

1. Every node of $G$ is contained in a bag
2. Every edge of $G$ is contained in a bag
3. Every node of $G$ appears in a contiguous subtree of $\text{Tree}(G)$. 

![Diagram of a graph $G$ and its tree decomposition $\text{Tree}(G)$]
Definition (Tree decomposition)

Given a graph $G = (V, E)$, a **tree-decomposition** $\text{Tree}(G) = (V_T, E_T)$ is a **tree of bags** $B_i \subseteq V$ such that:

1. Every node of $G$ is contained in a bag
2. Every edge of $G$ is contained in a bag
3. Every node of $G$ appears in a contiguous subtree of $\text{Tree}(G)$.
Definition (Tree decomposition)

Given a graph $G = (V, E)$, a **tree-decomposition** $\text{Tree}(G) = (V_T, E_T)$ is a tree of bags $B_i \subseteq V$ such that:

1. Every node of $G$ is contained in a bag
2. Every edge of $G$ is contained in a bag
3. Every node of $G$ appears in a contiguous subtree of $\text{Tree}(G)$.
Any graph $G$ can be constructed as the product of two constant-treewidth graphs $G_1$, $G_2$

Semiring distance on the product of $G_1$, $G_2$ as hard as on $G$

Widely conjectured $\Omega(n^3)$ lower-bound