All-Pairs Reachability
a.k.a.
Transitive Closure

Fully Dynamic Transitive Closure

directed graph +
sequence of operations:
queries \( s \rightarrow t \)
edge insertions & deletions
<table>
<thead>
<tr>
<th>Query &amp; Update Times (in $O$)</th>
<th>2 Very Large Studies [FMNZ01, KZ08]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>$\sqrt{n}$</td>
</tr>
<tr>
<td>$1$</td>
<td>$n^2$</td>
</tr>
<tr>
<td>$\sqrt{n}$</td>
<td>$m^{\sqrt{n}}$</td>
</tr>
<tr>
<td>$m^{0.43}$</td>
<td>$m^{0.58}$</td>
</tr>
<tr>
<td>$n^{0.58}$</td>
<td>$n^{1.58}$</td>
</tr>
<tr>
<td>$n^{1.495}$</td>
<td>$n^{1.495}$</td>
</tr>
<tr>
<td>$n$</td>
<td>$m + n \log n$</td>
</tr>
<tr>
<td>$n^{1.407}$</td>
<td>$n^{1.407}$</td>
</tr>
</tbody>
</table>

**Our Idea**

Engineering algorithms that . . .

- use single-source reachability
- don’t maintain SCCs
- profit from SCCs
Supportive Vertices

Observations

Let $v, s, t$ be vertices.

$R^+(v)/R^-(v)$: vertices reachable from/to that can reach $v$

Consider query $s \leadsto t$:

$v$ is a supportive vertex: $R^+(v)/R^-(v)$ can help to answer $s \leadsto t$
Supportive Vertices Algorithms

General Outline

- Store list of supportive vertices $L_{SV}$
  $\forall v \in L_{SV}$: maintain $R^+(v)$ and $R^-(v)$ via SSR; algorithms

- Updates (edge insertions & deletions):
  forward to SSR algorithms

- Query:
  $\forall v \in L_{SV}$: try to answer via (O1), (O2), (O3)
  fallback to static graph traversal

single-source/
single-sink
reachability

(O1)  (O2)  (O3)
## Supportive Vertices Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SV($k$)</td>
<td>pick $k$ vertices as supportive vertices uniformly at random</td>
</tr>
<tr>
<td>SVA($k$, $c$)</td>
<td>with adjustments</td>
</tr>
<tr>
<td>SVC($z$, $c$)</td>
<td>with SCC cover</td>
</tr>
</tbody>
</table>

### Initialization:
- **SV($k$):**
  - pick $k$ vertices as supportive vertices uniformly at random
- **SVA($k$, $c$):**
  - compute SCCs \( \{ S_0, \ldots, S_\ell \} \)
  - if \( |S_i| \geq z \): pick supportive vertex for \( S_i \) as representative
- **SVC($z$, $c$):**
  - map: vertex \( \rightarrow \) representative

### Updates:
- re-initialize every $c$ updates

### Queries:
- try supportive vertices in order of \( L_{SV} \)
- fallback: static graph traversal
- lookup & use representatives
- remove invalid entries from map
- fallback: mode of SV/SVA
Single-Source Reachability Subalgorithms

Extended Simple Incremental Algorithm (SI)
Maintains reachability tree:

*Insertions:* extend tree via BFS
*Deletions:* reconstruct tree via backward/forward BFS
*Queries:* $O(1)$ time

Simplified Extended Even-Shiloach Trees (SES)
Maintains BFS tree:

*Insertions:* update BFS levels
*Deletions:* simplified ES tree routine
*Queries:* $O(1)$ time
Experiments

All algorithms implemented in C++17 as part of the open-source algorithms library Algora.

Code available publicly on Gitlab & Github:

Algorithms

- BFS, DFS, DBFS (DFS-BFS hybrid)
- BiBFS (bidirectional BFS)
- SV with $k = 1$, $k = 2$, $k = 3$ *
- SVA with $k = 1$ and $c = 1k$, $c = 10k$, $c = 100k$ *
- SVC with $z = 25$ or $z = 50$ and $c = 10k$, $c = 100k$ *

* Fallback: BiBFS; SSR algorithms: SES, SI [HHS20]
Experiments: Instances

Random dynamic instances

ER graphs:
\[ n = 100k \text{ and } n = 10m, \ m_{\text{init}} = d \cdot n, \ d \in [1.25 \ldots 50] \]
\[ \sigma = 100k, \text{ different ratios of insertions/deletions/queries} \]

Stochastic Kronecker graphs with random update sequences:
\[ n \approx 130k \text{ and } n \approx 30 \ldots 130k, \ m_{\text{avg}} = d \cdot n, \ d = 0.7 \ldots 16.5 \]
\[ \sigma_{\pm} = 1.6m \ldots 702m \text{ and } \sigma_{\pm} = 282k \ldots 82m \text{ (updates only)} \]

Real-world dynamic instances

... with real-world update sequences:
\[ n = 100k \ldots 2.2m, \ m_{\text{avg}} = d \cdot n, \ d = 5.4 \ldots 7.8 \]
\[ \sigma_{\pm} = 1.6m \ldots 86.2m \text{ (updates only)} \]

... with randomized update sequences:
\[ n = 31k \ldots 2.2m, \ m_{\text{avg}} = d \cdot n, \ d = 4.7 \ldots 10.4 \]
\[ \sigma_{\pm} = 1.4m \ldots 76.4m \text{ (updates only)} \]
Experiments: Random Instances

\[ n = \sigma = 100k, \ \rho_{IDQ} = 1 : 1 : 1 \]
Experiments: Random Instances

\( n = \sigma = 100k, \ \rho_{IDQ} = 1 : 1 : 2 \ | \ n = 10m, \ \sigma = 100k, \ \rho_{IDQ} = 1 : 1 : 1 \)
Experiments: Kronecker Instances

\[ n \approx 130k, \sigma_{\pm} = 1.6m \ldots 702m, \rho_{UQ} = 2 : 1 \]

**Fastest:** \( SV(1), SV(2), SVC(25, 100k) \)

**BFS, DFS, DBFS:** \( > 6 \text{ h on } \geq 13/20 \) instances

similar picture for \( n \approx 30 \ldots 130k \)
Experiments: Real-World Instances

\[ n = 31k \ldots 2.2m, \sigma_{\pm} = 1.6m \ldots 86.2m, \rho_{UQ} = 2 : 1 \]

Fastest: SV(1), SV(2)

BFS, DFS, DBFS: \( \approx 6\% \) in 24 h on DE instance

similar picture on set with randomized updates
Conclusion

+ more stable query time
+ fast on sparse instances
– doubled update time
– considerably increased update time
?? recompute less often

Slower by several orders of magnitude: BFS, DFS, DBFS, BiBFS

Kathrin Hanauer
Faster Fully Dynamic Transitive Closure in Practice
Supportive Vertices

Observations
Let \( v, s, t \) be vertices.

\( R^+(v)/R^-(v) \): vertices reachable from/that can reach \( v \)

Consider query \( s \sim t \):

\( v \) is a supportive vertex: \( R^+(v)/R^-(v) \) can help to answer \( s \sim t \)

---

Experiments: Random Instances
\( n = \sigma = 100k, \rho_{IDQ} = 1:1:1 \)

\( 1.25 2.5 5 10 20 40 \)

\( 0.1 \)

\( 1 \)

\( 10 \)

\( 100 \)

\( 1 000 \)

\( 10 000 \)

\( \text{SVC}(25, \infty) \text{ SV}(1) \)

\( \text{Mean total operation time (s)} \)

\( \text{relative to } \text{SVC}(25, \infty) \)

---

Conclusion

1
2 3

\( \text{SV}(1) \)

\( \text{SV}(2) \)

\( \text{SVC}(*, 100k) \)

+ more stable query time

– doubled update time

+ more stable query time

+ fast on sparse instances

– considerably increased update time

?? recompute less often

---

Dynamic instances & source code:

https://dyreach.taa.univie.ac.at/transitive-closure

---


