Object of study: **directed graphs**

**Static vs. dynamic:**
Do the arcs change over time?

**Reachability question**
Is there a path from $u$ to $v$?

**k-reachingability question**
How many arc-disjoint paths from $u$ to $v$?
### My contributions

1) Fully dynamic reachability for graphs of partial functions
- special class of graphs
- simple bookkeeping data structure

2) All-Pairs 2-Reachability in $O(n^{\omega \log n})$ time
- any graph
- encoding for DAGs
- auxiliary graph for SC

3) All-Pairs $k$-Reachability on DAGs in $O(n^{\omega + o(1)})$ for small $k$
- DAGs only
- structure of cuts
- a lot of encoding tricks
1) Fully dynamic reachability for graphs of partial functions

joint work with Timon Gehr
Compiler Motivation: Semantic Analysis

Are all variables in the code well-defined?

- compile-time code evaluation makes it tricky
- AST nodes and dependencies vary over time
- node evaluation might be blocked by at most one other node → detect cyclic dependencies
Each vertex has at most one out-neighbour

- graph = cycles + trees hanging off of them
Book keeping: remember arc that closed the cycle separately

- rest = rooted forests to be stored in link-cut trees

Algorithm \text{QUERY}(u, v)

1: if $D'.\text{Root}(u) \neq D'.\text{Root}(v)$ then
2: \quad return false
3: end if
4: if $D'.\text{LCA}(u, v) = v$ then
5: \quad return true
6: end if
7: $x \leftarrow D'.\text{Root}(u)$
8: if $A_C.\text{FIND}(x) = \text{false}$ then
9: \quad return false
10: end if
11: $(x, y) \leftarrow A_C.\text{FIND}(x)$
12: if $D'.\text{LCA}(v, y) = v$ then
13: \quad return true
14: else
15: \quad return false
16: end if
2) All-Pairs 2-Reachability in $O(n^{\omega \log n})$

joint work with Loukas Georgiadis, Giuseppe F. Italiano, Nikos Parotsidis, and Przemysław Uznański, ICALP 2017
Our Problem: All-Pairs 2-Reachability

Given: directed graph \( G = (V, A) \)
with \( n \) vertices, \( m \) arcs

**For all** \( u, v \in V \), **decide if**

- many arc-disjoint \( u-v \)-paths
- some arc \( a \) on every \( u-v \)-path
- no \( u-v \)-path at all

Goal: prepare for constant query time
Improving 2-Reachability

Main Result

All-pairs 2-reachability with witnesses in $O(n^\omega \log n)$

Previously fastest: single source in $O(m)$, all pairs in $O(n \cdot m) \subseteq O(n^3)$

Algorithm:

- acyclic
- strongly connected
- combine

[Alstrup, Harel, Lauridsen, Thorup 1990] non-trivial combinatorics, dominator trees

[Georgiadis, Graf, Italiano, Parotsidis, Uznański, ICALP 2018]

$\omega < 2.373$
DAGs

Dynamic Programming

State: $L[u,v] = \begin{cases} 
\perp & \text{no } u\text{-}v\text{-path} \\
\top & \text{multiple disjoint } u\text{-}v\text{-paths} \\
a & \text{first common arc of all } u\text{-}v\text{-paths} 
\end{cases}$

\[ L[u,p] \]

\[ L[u,q] \]

\[ L[u,v] \]

Time: $O(n \cdot m) \subseteq O(n^3)$
Arc split: \[ A = A_1 \cup A_2 \] once you leave \( A_1 \), you can never go back
Path families:

Extend DP-cases to: \( \text{Any}[u, v] = \bigoplus_w (L[u, w] \otimes R[w, v]) \)
### DAGs

**Divide & Conquer**

**Details**

<table>
<thead>
<tr>
<th>Recursion</th>
<th>Path Algebra</th>
<th>Bit representations</th>
</tr>
</thead>
<tbody>
<tr>
<td>arc splits</td>
<td>serial paths</td>
<td>simulate $\otimes, \oplus$ with $\land, \lor$</td>
</tr>
<tr>
<td>combine C, D</td>
<td>$a \otimes \bot = \bot$</td>
<td>${\bot, \top} \cup A \rightarrow {0, 1}^{\Theta(\log n)}$</td>
</tr>
<tr>
<td>combine B, CD</td>
<td>$a \otimes \top = a$</td>
<td>$\bot \rightarrow 0 \ldots 0</td>
</tr>
<tr>
<td></td>
<td>$a \otimes a' = a \text{ or } a'$</td>
<td>$\top \rightarrow 1 \ldots 1</td>
</tr>
<tr>
<td></td>
<td>parallel paths</td>
<td>$a_1 \rightarrow b_{a_1} \overline{b_{a_1}}</td>
</tr>
<tr>
<td></td>
<td>$\bot \oplus a = a$</td>
<td>$a_2 \rightarrow 1 \ldots 1</td>
</tr>
<tr>
<td></td>
<td>$\top \oplus a = \top$</td>
<td>allows 0, 1, 2+ counting</td>
</tr>
<tr>
<td></td>
<td>$a \oplus a' = \top$</td>
<td>overall: $\tilde{O}(n^\omega)$</td>
</tr>
</tbody>
</table>
Main Result: All-pairs 2-reachability with witnesses in $O(n^\omega \log n)$

[Georgiadis, Graf, Italiano, Parotsidis, Uznański, ICALP 2018]

Algorithm: acyclic + strongly connected + combine
Testing 2-reachability after breaking symmetry

Claim: If some $a$ disconnects $u$ from $v$, then so does $a_u$ or $a_v$. 
Claim: If some $a$ disconnects $u$ from $v$, then so does $a_u$ or $a_v$.

Proof:

Consider: $G \setminus a_u$ and $G \setminus a_v$

If $a$ disconnects $u$ from $v$, then $a$ also disconnects $u$ from $s$ or $s$ from $v$.

If $a$ disconnects $u$ from $s$, $a_u$ also lies on all $u$-$a$-paths and also all $u$-$v$-paths.
**Query**

$u$-$v$-path in $G \backslash a_u$ and $u$-$v$-path in $G \backslash a_v$?

**Algorithm:**

- build two auxiliary graphs in $O(n^2)$
- take transitive closure in $O(n^\omega)$
- answer queries in $O(1)$
2-Reachability

Main Result
All-pairs 2-reachability with witnesses in $O(n^{\omega \log n})$

Algorithm:
acyclic + strongly connected + combine

[Georgiadis, Graf, Italiano, Parotsidis, Uznański, ICALP 2018]
First: compute SCCs and any topological order in $O(n + m)$

Is there a giant in the middle?

- Yes: $T(n) \leq T(n/3) + T(n/3) + O(n^\omega) + O(n^\omega \log n)$
- No: $T(n) \leq T(2n/3) + T(n/3) + O(n^\omega \log n) + O(n^\omega \log n)$

Dominating: $O(n^\omega \log n)$
3) All-Pairs $k$-Reachability on DAGs in $O(n^{\omega+o(1)})$ for small $k$

joint work with Loukas Georgiadis, Giuseppe F. Italiano, Nikos Parotsidis, and Przemysław Uznański
### Problem: bounded Min-Cut-Size with Witnesses

For all \(s\) and \(t\), find all \(s-t\)-earliest and \(s-t\)-latest \(\leq k\)-cuts \(\mathcal{E}_{s,t}^{\leq k}\) and \(\mathcal{F}_{s,t}^{\leq k}\).

- All earlier cuts are larger.
- All later cuts are larger.
- Cuts with at most \(k\) arcs.

#### Example:

- Report \(M_1\) as \(s-t\)-earliest 2-cut.
- Report \(M_3\) as \(s-t\)-latest 2-cut.
- Report \(M_4\) as \(s-t\)-latest 3-cut.
- Do not report \(M_2\) (not \(s-t\)-earliest/latest).
Structure of Cuts

Useful properties

• Earliest and latest min-cuts are unique
  [Ford, Fulkerson 1962] (proof by residual graph of max flow)

• $|\mathcal{E}_{s,t}^\leq k|, |\mathcal{F}_{s,t}^\leq k| \leq 4^k$
  [Cygan et al. 2015] (proof by arc replacement process)
First Idea: Combining Min-Cuts

This does not work:

- Looking only at pairs of min-cuts

s-t-min-cuts are not contained in any union of two s-v-, v-t-, s-w-, w-t-min-cuts
Working Idea: Combining $\leq k$-Cuts

This does work:

- For one $s$-$t$-min-cut $M$ and every $v$, $M$ contains an $s$-$v$-earliest or a $v$-$t$-latest cut.

Some $s$-$t$-min-cut contains $s$-$v$-earliest and $v$-$t$-latest cuts for all $v$. 
## Abstract Problem

### Witness Superset Problem

**Input**
- \( c \) set families \( \mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_c \)
- with \( \mathcal{F}_i = \{F_{i,1}, F_{i,2}, \ldots, F_{i,K}\} \)
- with \( F_{i,j} \subseteq U \) and \( |F_{i,j}| \leq k \)

**Output**
- all witnesses \( \mathcal{W} \) of size \( |\mathcal{W}| \leq k \)
- with \( \forall i \exists \mathcal{W}_i \subseteq \mathcal{W} \text{ s.t. } \mathcal{W}_i \in \mathcal{F}_i \)
- no subset of \( \mathcal{W} \) has this property

Example: \( \mathcal{F}_1 = \{\{2\}, \{1, 5\}\}, \mathcal{F}_2 = \{\{1,3\}, \{4\}\}, \mathcal{F}_3 = \{\{4\}, \{2, 4\}\} \)

for \( k=2 \), \( \mathcal{W}=\{2,4\} \) is the only witness
Algorithm 1: One-by-one

Process vertices one-by-one in reverse topological order:

For $v_i$ and $v_j$ look at:

\[
\{(v_i, v_{i_1})\} \cup \mathcal{F}_{v_{i_1}, v_j}^{\leq k},
\{(v_i, v_{i_2})\} \cup \mathcal{F}_{v_{i_2}, v_j}^{\leq k}, \ldots
\{(v_i, v_{i_k})\} \cup \mathcal{F}_{v_{i_k}, v_j}^{\leq k}
\]

After some filtering, gives $\mathcal{F}_{v_i, v_j}^{\leq k}$ in time $O(c \cdot 2^{O(k^2)})$

Result: For all $v_i, v_j$: time $O(mn \cdot 2^{O(k^2)}) = O(mn^{1+o(1)})$ for $k = o\left(\sqrt{\log n}\right)$
Algorithm 2: Divide-and-Conquer

Process first half and second half of topological order recursively:

For $v_i$ and $v_j$ look at:

- $E \leq k_{v_i, v_1} \cup F \leq k_{v_1, v_j}$
- $E \leq k_{v_i, v_2} \cup F \leq k_{v_2, v_j}$, \ldots
- $E \leq k_{v_i, v_c} \cup F \leq k_{v_c, v_j}$

Bottleneck is building the $n^2$ Witness Superset instances using encodings.

Result: Overall time $O(n^\omega \cdot (k \log n)^{4k+o(k)}) = O(n^{\omega+o(1)})$ for $k = o(\log \log n)$.
Encoding for Witness Superset

Encode the set $F_{i,j}$ into a $k$-superimposed codeword $C(F_{i,j})$ of size $\mathcal{O}(\text{poly}(k \cdot \log n))$.

Encode the family $\mathcal{F}_i$ as the tensor product $C^\otimes(\mathcal{F}_i) = C(F_{i,1}) \times C(F_{i,2}) \times \cdots \times C(F_{i,k})$.

Encode the set of families $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_c$ as the union $C^\otimes(\mathcal{F}_1) \cup C^\otimes(\mathcal{F}_2) \cup \cdots \cup C^\otimes(\mathcal{F}_c)$.

If we set $\mathcal{F}_\ell = \mathcal{E}_{v_i, v_\ell}^{\leq k} \cup \mathcal{F}_{v_\ell, v_j}^{\leq k}$ we get an encoding of the witness superset problem we want to solve.
## Conclusion

### 1) Fully dynamic reachability for graphs of partial functions
- special class of graphs
- simple bookkeeping data structure

### 2) All-Pairs 2-Reachability in $O(n^{\omega \log n})$
- any graph
- encoding for DAGs
- auxiliary graph for SC

### 3) All-Pairs $k$-Reachability on DAGs in $O(n^{\omega + o(1)})$ for small $k$
- DAGs only
- structure of cuts
- a lot of encoding tricks