Fine-Grained Complexity and Algorithm Design for Graph Reachability and Distance Problems

Karl Bringmann

“Fine-Grained Complexity, Logic, and Query Evaluation”
@ Simons Institute
September 28, 2023
Talk Outline

Hung’s invitation:

- survey-ish talk about **recursive query evaluation** from algorithms perspective
- reachability problems (connected components, transitive closure, ...)
- distance problems (shortest paths, diameter, ...)

Many Problem Variants

Input: graph $G = (V, E)$

What type of graph?
- undirected vs directed
- weighted vs unweighted
- encoding of weights, negative cycles?, ... 

Which parameters for measuring time?
- $n =$ number of nodes
- $m =$ number of edges
- output size, range of weights, ...
Reachability
Single-Source Reachability

given a node $s$, compute all nodes that are reachable from $s$

Classic optimal algorithm:

Run depth-first-search from $s$

linear time $O(n + m)$
All-Pairs Reachability

compute for all nodes \( u, v \) whether \( u \) can reach \( v \)

Undirected graphs \( \rightarrow \) connected components

Run depth-first-search from every unexplored node

linear time \( O(n + m) \)
All-Pairs Reachability

*compute for all nodes \( u, v \) whether \( u \) can reach \( v \)*

Directed graphs \( \rightarrow \) transitive closure, parameter \( m \):

Run single-source reachability from every node

\[ \text{time } O(nm) \leq O(m^2) \]

optimal since output size can be up to \( \Omega(m^2) \)
All-Pairs Reachability

compute for all nodes $u, v$ whether $u$ can reach $v$

Directed graphs $\rightarrow$ transitive closure, parameter $n$:

Run single-source reachability from every node

time $O(nm) \leq O(n^3)$

equivalent to Boolean matrix multiplication
All-Pairs Reachability

compute for all nodes $u, v$ whether $u$ can reach $v$

Directed graphs $\rightarrow$ transitive closure, parameter $n$:

**Transitive Closure**
- given directed $n$-node graph,
- compute for all nodes $u, v$
- whether $u$ can reach $v$

**Boolean Matrix Mult, BMM**
- given $n \times n$ matrices $A, B$,
- compute matrix $C$ with
  $C[i, j] = \bigvee_k A[i, k] \land B[k, j]$

$A :=$ adjacency matrix plus selfloops
for $i = 1, \ldots, \log n$:
  $A :=$ Boolean matrix product $A \ast A$

$\rightarrow$ compute transitive closure in time $\tilde{O}(n^\omega)$
All-Pairs Reachability

compute for all nodes $u, v$ whether $u$ can reach $v$

Directed graphs $\rightarrow$ transitive closure, parameter $n$:

Transitive Closure

given directed $n$-node graph, compute for all nodes $u, v$ whether $u$ can reach $v$

Boolean Matrix Mult, BMM

given $n \times n$ matrices $A, B$, compute matrix $C$ with

$$C[i, j] = \bigvee_k A[i, k] \land B[k, j]$$

From the transitive closure of this graph we can read off the Boolean matrix product $A \ast B$
All-Pairs Reachability

compute for all nodes \( u, v \) whether \( u \) can reach \( v \)

Directed graphs \( \rightarrow \) transitive closure, parameter \( n \):

Transitive Closure

given directed \( n \)-node graph, compute for all nodes \( u, v \) whether \( u \) can reach \( v \)

\[ \tilde{O}(n^\omega) \]

Boolean Matrix Mult, BMM

given \( n \times n \) matrices \( A, B \), compute matrix \( C \) with

\[ C[i, j] = \lor_k A[i, k] \land B[k, j] \]

\[ \tilde{O}(n^\omega) \]
All-Pairs Reachability

*compute for all nodes \( u, v \) whether \( u \) can reach \( v \)*

Directed graphs \( \rightarrow \) transitive closure, parameter \( m \):

optimal time \( O(m^2) \), by output size bound

Directed graphs \( \rightarrow \) transitive closure, parameter \( n \):

optimal time \( \tilde{O}(n^\omega) \), by equivalence with Boolean matrix product

parameter \textit{out} = number of edges in transitive closure?
All-Pairs Reachability

compute for all nodes $u, v$ whether $u$ can reach $v$

Directed graphs $\rightarrow$ transitive closure, parameter $\text{out}$:

Transitive Closure

\[
\text{out} = \text{number of edges in transitive closure}
\]

$\tilde{O}(\text{out}^c)$

Boolean Matrix Mult, BMM

\[
\begin{align*}
\text{in} & = \text{number of nonzero entries in input matrices} \\
\text{out} & = \text{number of nonzero entries in product matrix}
\end{align*}
\]

$\tilde{O}((\text{in} + \text{out})^c)$
## Fully-Sparse BMM

Solve BMM in time $\tilde{O}((in + out)^c)$ where $in / out = \# \text{nonzeros in input / output}$

With current $\omega$:

- $c \geq \omega / 2$ \quad \geq 1.18 \quad \geq 1$
- $c \leq 1.5$ \quad \leq 1.5 \quad \leq 1.5
- $c \leq \frac{2\omega}{\omega + 1}$ \quad \leq 1.41 \quad \leq 4/3
- $c \leq 1 + \frac{\mu}{1 + \mu}$ \quad \leq 1.3459 \quad \leq 4/3

Assuming $\omega = 2$:

- $c \geq \omega / 2$ \quad \geq 1
- $c \leq 1.5$ \quad \leq 1.5
- $c \leq \frac{2\omega}{\omega + 1}$ \quad \leq 1.41
- $c \leq 1 + \frac{\mu}{1 + \mu}$ \quad \leq 1.3459

Where $\omega(\mu, 1, 1) = 2\mu + 1$

$0.5 \leq \mu \leq 0.5286$

[van Gucht, Williams, Woodruff, Zhang '15]
[Amossen, Pagh '09]
[Abboud, B, Fischer, Künemann '23+]
Fully-Sparse BMM

solve BMM in time $\tilde{O}((\text{in} + \text{out})^c)$ where $\text{in} / \text{out} = \# \text{nonzeros in input / output}$

with current $\omega$:  assuming $\omega = 2$:

\[
\begin{align*}
  c &\geq \omega / 2 & \geq 1.18 & \geq 1 \\
  c &\leq 1.5 & \leq 1.5 & \leq 1.5 \\
  c &\leq \frac{2\omega}{\omega + 1} & \leq 1.41 & \leq 4/3 \\
  c &\leq 1 + \frac{\mu}{1 + \mu} & \leq 1.3459 & \leq 4/3
\end{align*}
\]

where $\omega(\mu, 1, 1) = 2\mu + 1$

$0.5 \leq \mu \leq 0.5286$

deterministic algorithm for BMM

also works for integer matrix mult, but randomized

[van Gucht, Williams, Woodruff, Zhang ’15]
[Amossen, Pagh ’09]
[Abboud, B, Fischer, Künemann ’23+]
Fully-Sparse BMM – Further Improvements?

solve BMM in time $\tilde{O}\left((\text{in} + \text{out})^c\right)$ where $\text{in} / \text{out} = \# \text{nonzeros in input/output}$

**BMM** has algorithm with exponent $c < 1 + \frac{\mu}{1 + \mu}$

$\iff$

**AllEdgesTriangle**($n^\mu, n, n; n^{1+\mu}$) can be solved in time $O(n^{1+2\mu-\varepsilon})$ for $\varepsilon > 0$

[Abboud, B, Fischer, Künnemann ‘23+]

0.5 ≤ $\mu$ ≤ 0.5286

for each edge: decide whether it is in a triangle
Fully-Sparse BMM – General Tradeoff

Our bound $\tilde{O}\left((in + out)^{1.3459}\right)$ is optimized for $out \approx in$

General setting: $out \approx in^r$ for some $r \in [0,2]$

With current $\omega$:

Near-linear time $\tilde{O}(out)$

If $out \geq in^{1.762}$

$\tilde{O}(in\sqrt{out})$

[van Gucht, Williams, Woodruff, Zhang '15]

[Amossen, Pagh ‘09]

[Abboud, B, Fischer, Künnemann ‘23+]

$\tilde{O}(in \cdot out^{0.3459} + in^{0.8002} out^{0.5457} + out)$
Fully-Sparse BMM – General Tradeoff

our bound $\tilde{O}\left((\text{in} + \text{out})^{1.3459}\right)$ is optimized for $\text{out} \approx \text{in}$

general setting: $\text{out} \approx \text{in}^r$ for some $r \in [0,2]$

with current $\omega$:

$\text{time } \tilde{O}\left(\text{in}^{1.762} + \text{out}\right)$

$v$an Gucht, Williams, Woodruff, Zhang ‘15]

$\tilde{O}\left(\frac{\text{in}}{\sqrt{\text{out}}}\right)$

[Amossen, Pagh ‘09]

[Abboud, B, Fischer, Künemann ‘23+]

$\tilde{O}\left(\text{in} \cdot \text{out}^{0.3459} + \text{in}^{0.8002} \text{out}^{0.5457} + \text{out}\right)$
Fully-Sparse BMM – General Tradeoff

our bound $\tilde{O}((in + out)^{1.3459})$ is optimized for $out \approx in$

general setting: $out \approx in^r$ for some $r \in [0,2]$

assuming $\omega = 2$:

\[
\tilde{O}(in^{1.5} + out)
\]

[van Gucht, Williams, Woodruff, Zhang '15]
\[
\tilde{O}(in\sqrt{out})
\]

[Amossen, Pagh '09]
\[
\tilde{O}(in \cdot out^{1/3} + out)
\]

[Abboud, B, Fischer, Künemann '23+]
Fully-Sparse BMM – Algorithm Overview

$A$ is $x \times y$-matrix, $B$ is $y \times z$-matrix

1. **Output Densification:**
   use hashing / sparse recovery to reduce outer dimensions to $x \cdot z = O(out)$

2. **High-degree/low-degree:**
   split $y$’s into degree higher than $\Delta$ or lower than $\Delta$
   low degree: enumerate all 2-paths in time $O(in \cdot \Delta)$
   high degree: matrix multiplication in time $MM(x, y_H, z)$

   \[
   \leq MM\left(x, \frac{in}{\Delta}, \frac{out}{x}\right) \quad \Delta \leq x \leq \frac{out}{\Delta}
   \]

   \[
   \leq MM\left(\Delta, \frac{in}{\Delta}, \frac{out}{\Delta}\right)
   \]

   use bounds on MM to bound both terms and balance their sum
All-Pairs Reachability

compute for all nodes \( u, v \) whether \( u \) can reach \( v \)

Directed graphs \( \rightarrow \) transitive closure, parameter \textit{out}: 

Transitive Closure

\[
\text{\textit{out}} = \text{number of edges in transitive closure}
\]

\[\tilde{O}(\text{\textit{out}}^{1.3459})\]

Boolean Matrix Mult, BMM

\[
\text{\textit{in}} = \text{number of nonzero entries in input matrices}
\]

\[
\text{\textit{out}} = \text{number of nonzero entries in product matrix}
\]

\[\tilde{O}((\text{\textit{in}} + \text{\textit{out}})^{1.3459})\]

[Abboud, B, Fischer, K{"u}nnemann ‘23+]

Q: What is the optimal exponent?
Distances
Weight Encoding

*each edge e has a weight/length w(e)*

**RAM model:** each edge weight fits into a machine cell
- arithmetic operations on two machine cells in time $O(1)$

1. **integer weights** in $\{-W, \ldots, W\}$
   - 1.1. **near-constant weights:** $W$ factors in running time are okay
   - 1.2. **polynomial weights:** $W \leq n^{O(1)}$, $\log W$ factors hidden by $\tilde{O}$
   - 1.3. **mildly superpolynomial weights:** $\log W$ factors are okay
   - 1.4. **strongly polynomial algorithms:** running time independent of $W$

2. **real weights**
   - 2.1. **RealRAM:** arithmetic operations on reals in constant time
   - 2.2. **floating point** approximation, e.g. $O(\log(n/\varepsilon))$-bit mantissa and exponent
Single-Source Shortest Paths

given a node s, compute distances from s to all other nodes

nonnegative edge weights:
Dijkstra‘s algorithm: \( \tilde{O}(m) = O(m + n \log n) \)

general edge weights:
Bellman-Ford algorithm: \( O(mn) \)  \[\text{[Ford’ 56, Bellman ‘58]}\]
Single-Source Shortest Paths

given a node $s$, compute distances from $s$ to all other nodes

**nonnegative edge weights:**

Dijkstra’s algorithm: $\tilde{O}(m) = O(m + n \log n)$

**general edge weights:**

Bellman-Ford algorithm: $O(mn)$

scaling-based algorithms: $O(m\sqrt{n} \log W)$

[Gabow ’83, Gabow, Tarjan ‘89, Goldberg ‘95]

recent breakthrough: $\tilde{O}(m \log W) = O(m \log^8 n \log W)$

[Bernstein, Nanongkai, Wulff-Nilsen FOCS’22 best paper]

further improvements: $O((m + n \log \log n) \log^2 n \log(nW))$

[B, Cassis, Fischer FOCS’23]

Q: Can the log $W$ factor be removed?
All-Pairs Shortest Paths

compute all pairwise distances in a graph

negative edge weights can be removed in time $O(nm)$ [Johnson’77]

**parameter $m$:**

Run single-source shortest paths from every node

time $\tilde{O}(nm) \leq \tilde{O}(m^2)$, optimal by output size

**parameter $n$:**

time $\tilde{O}(nm) \leq \tilde{O}(n^3)$

equivalent to MinPlusProduct
All-Pairs Shortest Paths

compute all pairwise distances in a graph

All-Pairs Shortest Paths

given a directed graph, compute for all nodes $u, v$
the distance from $u$ to $v$
$	ilde{O}(n^3)$

MinPlusProduct

given $n \times n$ matrices $A, B$, compute matrix $C$ with
$C[i, j] = \min_k A[i, k] + B[k, j]$
$	ilde{O}(n^3)$

$A :=$ weighted adjacency matrix plus 0-weight selfloops
for $i = 1, \ldots, \log n$:

$A :=$ MinPlus matrix product $A \ast A$
All-Pairs Shortest Paths

compute all pairwise distances in a graph

**All-Pairs Shortest Paths**

given a directed graph, compute for all nodes $u, v$ the distance from $u$ to $v$

$\tilde{O}(n^3)$

**MinPlusProduct**

given $n \times n$ matrices $A, B$, compute matrix $C$ with

$C[i, j] = \min_k A[i, k] + B[k, j]$

$\tilde{O}(n^3)$

From the pairwise distances in this graph we can read off the MinPlus matrix product $A \cdot B$
All-Pairs Shortest Paths

compute all pairwise distances in a graph

All-Pairs Shortest Paths

given a directed graph,
compute for all nodes $u, v$
the distance from $u$ to $v$

$\tilde{O}(n^3)$

MinPlusProduct

given $n \times n$ matrices $A, B$,
compute matrix $C$ with
$C[i, j] = \min_k A[i, k] + B[k, j]$

$\tilde{O}(n^3)$

APSP Hypothesis:

These problems cannot
be solved in time $O(n^{3-\delta})$

Negative Triangle

given an edge-weighted graph,
are there nodes $x, y, z$ with
$w(x, y) + w(y, z) + w(z, x) < 0$?
All-Pairs Shortest Paths

compute all pairwise distances in a graph

**parameter $m$:**

Run single-source shortest paths from every node

time $\tilde{O}(nm) \leq \tilde{O}(m^2)$, optimal by output size

**parameter $n$:**

time $\tilde{O}(nm) \leq \tilde{O}(n^3)$
equivalent to MinPlusProduct

optimality is the APSP hypothesis

$n^3 / 2^{\Omega(\sqrt{\log n})}$ [Williams ‘14]
Approximate All-Pairs Shortest Paths

compute $\alpha$-approximation of all pairwise distances in a graph

directed graph OR undirected graph and $\alpha < 2$:

time $\Omega(n^\omega)$, since at least as hard as BMM
Approximate All-Pairs Shortest Paths

compute $\alpha$-approximation of all pairwise distances in a graph

directed graph OR undirected graph and $\alpha < 2$:

time $\Omega(n^\omega)$, since at least as hard as BMM

$(1 + \varepsilon)$-approximation: time $\tilde{O}\left(\frac{n^\omega}{\varepsilon \log W}\right)$

Is $\log W$ factor necessary?
Approximate All-Pairs Shortest Paths

compute $\alpha$-approximation of all pairwise distances in a graph

directed graph OR undirected graph and $\alpha < 2$:

\[
\Omega(n^\omega), \text{ since at least as hard as BMM}
\]

\[(1 + \varepsilon)\text{-approximation:} \quad \text{time } \tilde{O}\left(\frac{n^\omega}{\varepsilon} \log W\right) \quad [\text{Zwick '02}]
\]

.. in undirected graphs: \quad \text{time } \tilde{O}\left(\frac{n^\omega}{\varepsilon}\right) \quad [\text{B, Künnemann, Wegrzycki STOC'19}]

.. in directed graphs: \quad \text{time } \tilde{O}\left(\frac{n^{(3+\omega)/2}}{\varepsilon}\right) \quad [\text{B, Künnemann, Wegrzycki STOC'19}]

equivalent to exact MinMaxProduct,
for which best known time is $\tilde{O}(n^{(3+\omega)/2})$
Approximate All-Pairs Shortest Paths

compute $\alpha$-approximation of all pairwise distances in a graph

directed graph OR undirected graph and $\alpha < 2$:

time $\Omega(n^\omega)$, since at least as hard as BMM

$O(1)$-approximation in undirected graphs:

preprocess given graph in time $O(mn^{1/k})$, $k = O(1)$ in [Thorup, Zwick ‘05]

then query($u, v$) returns a $(2k - 1)$-approximation of $\text{dist}(u, v)$
in query time $O(1)$

Under 3SUM, in the same preprocessing time and $n^{o(1)}$ query time we
cannot compute a $< k$-approximation → hardness of approximation in $P$

[Abboud, B, Khoury, Zamir STOC’22] [Abboud, B, Fischer STOC’23] [Jin, Xu STOC’23]
Conclusion

Graph reachability and distance problems:

**single-source**: mostly in near-linear time

**all-pairs**: mostly equivalent (up to logfactors) to an appropriate matrix product

Many, many more directions:

centrality measures: diameter, radius, eccentricities, girth, ...

additive approximation, small weights, ...

dynamic graphs, failing edges (replacement paths), spanners, ...

... a huge, active research area

Thank you!