Plan

- Query-update time tradeoffs
- Other conjectures
- Unconditional lower bounds
- Partially-dynamic algorithms
Part 1

Query-Update Time Tradeoffs
Motivation

• So far, we focuses on outputting something small (yes/no, numbers) after each update.
• More realistic: output when users want.
• Also: Users may just want part of the (large) output.
Example

Single-Source Reachability with queries
How should we define single-source reachability?

Option 1: Output list of reachable nodes
  • $\Omega(n)$ is an obvious update time lower bound
  • ... not so interesting

Option 2: Answer query “Can s reach u?”
  • Possible to get polylog update time in this case?
  • Let’s look into this
Single Source Reachability (ss-Reach)

1. Preprocess
Single Source Reachability (#ss-Reach)

1. Preprocess

2. Updates/Queries

Reach(1)?
Insert(4,5)
Delete(s,2)
Reach(5)?
Delete(3,4)
Reach(5)?
...
...
Single Source Reachability (ss-Reach)

1. Preprocess

2. Updates/Queries

Reach(1)?
Insert(4,5)
Delete(s,2)
Reach(5)?
Delete(3,4)
Reach(5)?
...
...
1. Preprocess

2. Updates/Queries

Reach(1)?
Insert(4,5)
Delete(s,2)
Reach(5)?
Delete(3,4)
Reach(5)?
...
...

Single Source Reachability (ss-Reach)
1. Preprocess

2. Updates/Queries

Reach(1)?
Insert(4,5)
Delete(s,2)
Reach(5)?
Delete(3,4)
Reach(5)?

...
Single Source Reachability (ss-Reach)

1. Preprocess

2. Updates/Queries

Reach(1)?
Insert(4,5)
delete(s,2)
Reach(5)?
Delete(3,4)
Reach(5)?
...
...

YES
A Naïve Algorithm for (fully dynamic) ss-Reach:

<table>
<thead>
<tr>
<th></th>
<th>Update time</th>
<th>Query time</th>
</tr>
</thead>
<tbody>
<tr>
<td>BFS from $s$ when update</td>
<td>$m$</td>
<td>1</td>
</tr>
</tbody>
</table>

Can we improve update time

$m \Rightarrow m^{1-\varepsilon}$? (maybe amortized)

$m = \max \text{ #edges}$
$n = \#\text{nodes}$
Reduction from $\gamma$-OuMv to ss-Reach
(sketched)

**Sketch:**
1. For each $(u_i, v_i)$: $n^\gamma$ updates and $n$ queries

2. $\gamma$-OuMv implies that amortized time over $n^\gamma$ updates and $n$ queries cannot be $O(n^{1+\gamma-\epsilon})$ for any $\epsilon > 0$.

3. If query time is $n^{o(1)}$, then update time cannot be $O(n^{1-\epsilon})$ for any $\epsilon > 0$.

4. For any $\epsilon' > 0$, update time of $O(m^{1-\epsilon'})$ implies amortized time of $O(n^{(1+\gamma)(1-\epsilon')})$.

5. ... which is $O(n^{1-\epsilon})$ for some $\epsilon > 0$ for small enough $\gamma$. 
Bounds for \textit{ss-Reach} via OMv

Forbidden by OMv

Forbidden by OMv

BFS when query

BFS when update

Sankowski'04

\((n^{1.495}, n^{1.495})\)
Further notes

- **OMv** (in fact \(\gamma\)-OuMv) gives tight lower bounds of query time and update-query tradeoffs for many problems.
Open: Close bounds for Subgraph Connectivity via OMv
Part 2

Other Conjectures?
As an algorithm designer, I’m not sure I should give up when I see lower bounds from other conjectures.

But they sometimes guide to good directions.
Example: st-Reach

- Bounds hold only for small preprocessing time
- Time smaller than OMv
- Bounds from BMM is only for "combinatorial" algorithms
  - They were broken by algorithms based on fast matrix multiplication
Should I make new conjectures?

Our own study case: **st-reach**

- After failing to further improve our algebraic algorithms for st-reach and related problems. We made three conjectures. One of them:

  **v-hinted OMv (informal)**

  **Input:** Phase 1: Boolean matrix $M$, Phase 2: a Boolean matrix $V$, Phase 3: index $i$.
  **Output** the matrix-vector product $M V_i$, where $V_i$ is the i-th column of $V$.

  **Naïve algorithm:** Compute $MV$ in phase 2 or $MV_i$ in phase 3.

  **Conjecture:** Nothing better than the naive algorithm.

- The three together give tight lower bounds for $\approx 20$ problems, including st-reach.
Part 3

Unconditional Lower Bounds?
Conjectures are sometimes attempted in the cell-probe model.

Examples:

- [Cl-Gr-L’15]: Cell probe lower bounds for $OM_v$ problem over very large finite fields $F$, space usage $S=\min(n \log |F|, n^2)$ when $|F|=n^{\Omega(1)}$, $S=O(n)$.
  - This does not imply the OMv Conjecture (need the Boolean case).
- [Larsen-Williams’17]: The OMv conjecture cannot be true on the cell-probe model.
Patrascu’s multiphase problem and communication model

**Multiphase Problem:** Three phases of inputs
- **Phase 1:** $n \times n$ Boolean matrix $M$
- **Phase 2:** Vector $v$
- **Phase 3:** Integer $i$

**Output:** $(Mv)_i$

**Naïve** algorithm: Compute $Mv$ in phase 2 ($O(n^2)$ time) or $M_ivi$ in Phase 3 ($O(n)$ time)

**Observe:** $OMv$ implies that the native algorithm is best.

**Weaker** lower bounds can be derived from, e.g., 3SUM
Patrascu’s multiphase problem and communication model

Claim:
• If exists algorithm A with $O(n^{1.9})$ & $O(n^{0.9})$ time in Phases 2 & 3,
• Then exists protocol where teacher sends $O(n^{1.9})$ bits and Alice and Bob exchanges $O(n^{0.9})$ bits.

Proof:
• Teacher sends what CPU wrote on memory in Phase 2 to Alice. $[O(n^{1.9})$ bits$]$
• Alice simulate Phase 3, and ask Bob for some missing bits (written in Phase 1). $[O(n^{0.9})$ bits$]$

Enough to show lower bounds for communication with Advice

Naïve: Teacher sends $M$ ($O(n^2)$) or Alice sends $v$ ($O(n)$)
Want: $(Mv)_i$
Part 4

Partially-Dynamic Algorithms
Notes

• Partially dynamic means insertions-only or deletions-only

• Instead of amortized update time, we can analyze total update time instead.

• We have see:
  – Incremental connectivity with $O(\log n)$ worst-case update time.
  – Incremental single-source reachability with $O(m)$ total update time ($O(1)$ amortized).
Motivation

• **Enough for some data**: social networks rarely have deletions (“unfriend”)

• Sometimes **equivalent** to fully-dynamic case
  – E.g. fully-dynamic connectivity is equivalent to the deletion-only one

• **Enough as a subroutine for some problems**

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Decremental SSSP [HKN FOCS’14, ?]</td>
<td>Approx. s-t flow</td>
</tr>
<tr>
<td>Decremental min-cut (restricted)</td>
<td>Interval Packing, Traveling salesperson [Chekuri-Quanrud SODA’17, FOCS’17]</td>
</tr>
</tbody>
</table>
Example: Dyn. Shortest Paths → Max Flow

Garg-Konemann [FOCS’98], Madry [STOC’10]:
Deterministic $m^{1+o(1)}$ total update time for weighted $(1+\varepsilon)$-approx decremental st-shortest path → $m^{1+o(1)}$-time $(1+\varepsilon)$-approx max flow

Randomized algorithm against adaptive adversary is also enough.

Known: Randomized $m^{1+o(1)}$ total update time
[HenzingerKN. FOCS’14]
Example 1:

**st-Distance under insertions**

(It is possible to prove tight total update time!)

Thanks Thatchaphol Saranurak for slides
Theorem [Even-Shiloach JACM’81, Dinitz’71]

A BFS tree can be maintained with $O(mn)$ total time for $m$ edge insertions.

(Thus $O(n)$ amortized over $m$ insertions)
Even-Shiloach [JACM’81]

Well-known as Even-Shiloach Tree (ES-tree)
Dinitz [Voprosy Kibernetiki’71]

Original version of Dinitz’s maxflow algorithm

For detail, see “Dinitz' Algorithm: The Original Version and Even's Version”
Description of Even-Shiloach tree as nodes talking to each other
\begin{technical}
Compute BFS tree from $s$. Every node maintains its *level*.
Add edge \((s, b) \rightarrow s\) and \(b\) check if their levels should change
b changes its level.
It informs this to all neighbors.
Neighbors check if they should change levels. Node e should in this case. Again e informs neighbors.

This is what we obtain after adding (s,b)
Even-Shiloach tree can be implemented in such a way that total update time = number of messages.

Takes 2 time steps.
Exercise

Number of messages (thus time complexity) after $m$ insertions is $O(mn)$

Hint

Node $v$ sends $\text{degree}(v)$ messages every time $\text{level}(v)$ decreases.
\end{technical}
Lemma: st-distance cannot have total update time $O((mn)^{1-\varepsilon})$, assuming the OMv conjecture.

Proof sketch:

\[
\text{dist}(s,t) = 2n + 1 \text{ iff “yes”}
\]

\[
\text{dist}(s,t) = 2(n-1) + 1 \text{ iff “yes”}
\]
Example 2

st-Reach under insertions
This example shows ...

- Converting *amortized* fully-dynamic lower to *worst-case (only!)* for partially-dynamic lower bounds.
- It works for most problems.
Claim: Incremental st-Reach has $\Omega(n)$ worst-case lower bound

- Trick: *Undo (roll-back)* insertions before new insertions
- Worst-case update time $O(n^{0.9}) \rightarrow O(n^{1.9})$ time per $(L_i, R_i)$. Contradicting OuMv conj.

Doesn’t work for total update time: If assume, say, $O(n^2)$ total update time, we may spend $O(mn^{1-\epsilon})$ time per $(L_i, R_i)$. Nothing to contradict.
Questions?

Acknowledgements:
Sayan Bhattacharya, Jan van den Brand, Deeparnab Chakraborty, Sebastian Forster, Monika Henzinger, Christian Wulff-Nilsen, Thatchaphol Saranurak

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 715672