Chapter 3

Lower Bounds from SETH

Danupon Nanongkai

KTH, Sweden
Summary

**OMv conjecture**

- Usually refutes $n^{1-\epsilon}$ update time on dense graph, or $m^{1-\epsilon}$ in general.
- This helps refute polylogarithmic time, but might not be tight.

**SETH**

- SETH refutes $n^{1-\epsilon}$ time for sparse graph, refuting $m^{1-\epsilon}$ in general.
- To start from SETH, reduce from dynamic OV, a.k.a. dynamic client-server.
- For some problem (e.g. diameter), SETH even refutes $n^{2-\epsilon}$ bound (but need 3OV)!

$n = \#$ of nodes, $m=\#$ of edges
* There are some exceptions.
Part 1

SETH and Dynamic OV
(a.k.a. Dynamic Client-Server Problem)
Starting point: Client-Server Problem

Preprocess: \( \text{poly}(N) \) time

Updates: A server becomes active/inactive

Output: All clients are connected to active servers?

Naïve algorithm takes \( O(N) \) update time

\[ \text{SETH} \rightarrow \text{No } N^{1-\epsilon} \text{ amortized per server update} \]

Sparsity: 
\[ \#\text{edges} = O(N \log N) \]
Starting point: Client-Server Problem

Preprocess: \( \text{poly}(N) \) time

Updates: A server becomes active/inactive

Output: All clients are connected to active servers?

Naïve algorithm takes \( O(N) \) update time

\[ \text{SETH} \rightarrow \quad \text{No } N^{1-\epsilon} \text{ amortized per server update} \]

Sparsity:
\[ \#\text{edges} = O(N \log N) \]
Details: \#servers depends on preprocessing time

- Otherwise, you can prepare for all $2^{\#servers} = n^{O(1)}$ possible sets of active servers during the preprocessing time.

- The claim should be interpreted as: If someone claims to have an algorithm with $N^c$ preprocessing time, then we can pick the number of servers to be $f(c) \log N$. Then, SETH implies that there is no algorithm with $N^c$ preprocessing time and $N^{1-\epsilon}$ update time.
Motivation: OuMv can be viewed as Client-Server with \( \#\text{clients}=\#\text{server}^2 \)

OMv \(\rightarrow\) No \( N^{1/2-\epsilon} \) amortized time per node recolor with polynomial preprocess
**Claim:** SETH implies no $N^{1-\varepsilon}$ amortized per server update

**Proof** (sketched):

**OV:** Given sets $A$ and $B$ of vectors, exists $u \in A, v \in B$ s.t. $\langle u, v \rangle = 0$?

- SETH implies no $(|A||B|)^{1-\varepsilon}$ time.
- Hold for: $|A| = N, |B| = poly(N)$ and dimension $n = O(\log N)$

**Reduction:**

1. Vectors in $A$ $\rightarrow$ **Clients**.
2. Coordinates $\rightarrow$ **Servers**.
3. Each vector in $B$ $\rightarrow$ Each set of active servers

**Example:** $A = \{10, 11, 01\}, B = \{11, 01\}$
Claim: SETH implies no $N^{1-\epsilon}$ amortized per server update

Proof (sketched):

OV: Given sets $A$ and $B$ of vectors, exists $u \in A$, $v \in B$ s.t. $\langle u, v \rangle = 0$?

- SETH implies no $(|A||B|)^{1-\epsilon}$ time.
- Hold for: $|A| = N$, $|B| = poly(N)$ and dimension $n = O(\log N)$

Reduction:
1. Vectors in $A$ $\rightarrow$ Clients.
2. Coordinates $\rightarrow$ Servers.
3. Each vector in $B$ $\rightarrow$ Each set of active servers

Example: $A = \{10, 11, 01\}$, $B = \{11, 01\}$

Analysis:
- Preprocessing time = $poly(|A|) < (|A||B|)^{1-\epsilon}$ if $|B|$ is big enough compared to $|A|$.
- Assume time per server update is $N^{1-\epsilon}$. Then, time per vector $v \in B$ is $(N)^{1-\epsilon}n$.
- So total time is $|B|(Nn)^{1-\epsilon} = |A|^{1-\epsilon}|B| = (|A||B|)^{1-\epsilon}$
Another form: Dynamic OV

• **Preprocess**: Set $A$ of Boolean vectors
  – Let $N = |A|$. Vectors have dimension $O(\log N)$.
• **Update**: A Boolean vector $\nu$
• **Output**: Exists $u \in A$ s.t. $\langle u, \nu \rangle = 0$?

Naïve algorithm takes $O(N \log N)$ time per $\nu$.

**Claim**: $\text{SETH} \implies$ No $N^{1-\epsilon}$-time algorithm with polynomial preprocessing time.
Part 2

Some Reductions from Dynamic OV (Client-Server)
Plan

• **Single-Source Reachability Count (#SSR):** Counting number of nodes reachable from \( s \)

• **Strongly-Connected Component Count (#SCC):** Counting number of strongly connected components

**Lesson:** SETH may give higher lower bounds (than OMv) in \( m \) for **counting** versions.

**Intuition:** The client-server problem is about the number of connected clients.
Example 1

Reachability - #SSR
st-Reachability (recall)

• Exists directed path from s to t?
• No \( n^{1-\epsilon} \) update time (on dense graph) assuming OMv
  – Hold against randomized and amortized algorithms
  – Implies \( m^{1/2-\epsilon} \) lower bound
• Open: Higher lower bound*?

Single-Source Reachability Count (#SSR)

• How many nodes are reachable from s?
• No \( m^{1-\epsilon} \) update time assuming SETH
  – Hold against randomized and amortized algorithms

Later: SSR with \( n^{o(1)} \) query time also has no \( m^{1-\epsilon} \) update time assuming OMv.

---

\( n = \# \) of nodes, \( m = \# \) of edges
* We do have higher bound for worst-case update time
Claim: No $m^{1-\epsilon}$ update time assuming SETH

Reduction: Add edges from $s$ to all active servers

$\text{Seth} \rightarrow \text{No } n^{1-\epsilon} \text{ time per server update}$

$\rightarrow \text{No } n^{1-\epsilon} \text{ time per edge update for } \#\text{SSR}$

$\rightarrow \text{No } m^{1-\epsilon} \text{ time since graph is sparse}$!

$n = \# \text{ of nodes, } m = \# \text{ of edges}$
Example 2

**Strong Connectivity - #SCC**
**Strong Connectivity (recall)**
- Exists directed path from *every* s to *every* t?
- **No** $n^{1-\epsilon}$ update time assuming OMv
  - Hold against randomized and amortized algorithms
  - Implies $m^{1/2-\epsilon}$ lower bound
- **Open**: Higher lower bound?

**Strongly Connected Components Count (#SCC)**
- How many strongly connected components are there?
- **No** $m^{1-\epsilon}$ update time assuming SETH

$n = \# \text{ of nodes}, m = \# \text{ of edges}$
Claim: No $m^{1-\epsilon}$ update time assuming SETH

Reduction:
- Add edges from all clients to $s$ to all active servers
- Add edges between $t$ and all inactive servers

Observe: Yes for clients-servers $\leftrightarrow$ #SCC$\leq 2$.
- All active servers and adjacent clients form one component with $s$.
- Other clients are not in any connected components.
- Inactive servers form another component with $y$. 

Claim: No $m^{1-\epsilon}$ update time assuming SETH
Part 3

Diameter from dynamic 30V
**Dynamic Diameter:** Output the diameter of an **undirected** graph

**Algorithms**
- **Naïve algorithm:** $O(mn)$ per update.
- **Best (via APSP):** $O(n^2)$ amortized update time and $O(n^{2\frac{2}{3}})$ worst-case.

**Lower Bounds**
- **No** $n^{1-\epsilon}$ update time assuming OMv [Thanks to a participant!]
  - Implies $m^{1/2-\epsilon}$ lower bound
- **No** $n^{2-\epsilon}$ update time assuming SETH
  - **Not** known how to prove this from dynamic OV (client-server)
  - Instead, reduce from **dynamic 3-OV**

*Both hold against randomized and amortized algorithms*

**Lesson:** Dynamic 3-OV might be useful for problems that involve **many** pairs of nodes.

\[ n = \# \text{ of nodes}, \quad m = \# \text{ of edges} \]
Dynamic 3-OV

- **Preprocess**: Set $A, B$ of Boolean vectors
  - Let $N = |A| = |B|$. Vectors have dimension $O(\log N)$.
- **Update**: A Boolean vector $w$
- **Output**:Exists $u \in A, v \in B$ s.t. entry-wise multiplication of $u \circ v \circ w = 0$?

Naïve algorithm takes $O(N^2 \log N)$ time per $v$
(Keep track of all pairs)

Claim: SETH $\Rightarrow$ No $N^{2-\epsilon}$-time algorithm with polynomial preprocessing time
Proof: Omitted.
Client-Server Form of dynamic 3OV

Exists pair of red-black client that doesn’t share active server?

No: a, b, c, d

Yes: a, c

SETH $\Rightarrow$
No $N^{2-\epsilon}$ amortized per server update
Example of how it’s related to 3OV

\[
SETH \rightarrow \\
\text{No } N^{2-\epsilon} \text{ amortized per server update}
\]
Reduction to Diameter (partial)

1. Create copies of servers.
2. Connect black and red clients to different copies.
3. If a server is active, connect its two copies.

**Intuition:** Red-Black clients that share active servers has distance 3. (Otherwise distance will be more.)

**Problem:** How about black-black clients, etc?

**3OV:** Exists pair without shared server?

**Diameter**

\[
\text{dist}(a, c) > 3
\]

\[
\text{dist}(a, d) = 3
\]
Reduction to Diameter (full)

1. Create copies of servers.
2. Connect black and red clients to different copies.
3. If a server is active, connect its two copies.
4. Add edges between black clients and x, servers and y, red clients and z.

**3OV: Exists pair without shared server?**

**Diameter**

Claim (Tedious to check): Diameter > 3 iff exists pair without shared server
Questions?

Thanks to co-authors:
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