Lecture 5: Hardness for Sequence Problems under SETH and OVC

Thanks to Piotr Indyk and Arturs Backurs for some slides
Plan

• Define sequence problems:
  – (Discrete) Frechet Distance
  – Edit Distance and LCS
  – Dynamic Time Warping (DTW)

• Birds eye view on the upper bounds
  – Dynamic programming, quadratic time

• Show conditional quadratic lower bounds
  – Assuming SETH / OVC, example: LCS
Walks on sequences

Given two sequences \( \{p_i\} \) and \( \{q_j\} \), a walk on them starts at \( p_1 \) and \( q_1 \). In each step it is in some position \((p_i, q_j)\) and can next:

- go right only on \( p \) to \((p_{i+1}, q_j)\)
- go right only on \( q \) to \((p_i, q_{j+1})\)
- go right on both to \((p_{i+1}, q_{j+1})\)

Sequence walk problems optimize, over all such walks, some measure depending on the distances between \( p_i \) and \( q_j \) over all steps \((p_i, q_j)\) of the walk.
(Discrete) Frechet Distance  [Alt-Godau’95]

• “Dog walking distance”
  – Smallest length leash that enables dog-walking along two routes

• Definition:
  – Let $F = \text{set of monotone functions } [0,1] \rightarrow [0,1]$
  – For two curves $P, Q: [0,1] \rightarrow \mathbb{R}^2$:
    \[
    D_{Fr}(P,Q) = \min_{f,g \in F} \max_{t \in [0,1]} ||P(f(t)) - Q(g(t))||
    \]

• Discrete version:
  – $F = \{ f: [0,1] \rightarrow \{1...n\} , \text{ nondecreasing}\}$,
  – $P,Q: \{1...n\} \rightarrow \mathbb{R}^2$ : Curves are sequences of points in the plane
Frechet Distance: Algorithm

• Discrete version:
  – Let $F = \{ f: [0,1] \rightarrow \{1...n\}, \text{nondecreasing} \}$, mapping time to position,
  – For two sequences of points, $P, Q: \{1...n\} \rightarrow \mathbb{R}^2$:
    \[
    D_{Fr}(P,Q) = \min_{f,g \in F} \max_{t \in [0,1]} ||P(f(t)) - Q(g(t))||
    \]

• Dynamic programming:
  – $A[i,j] = \text{distance between curves } P(1)...P(i) \text{ and } Q(1) ...Q(j)$
  – $A[i,j]=\max[||P(i)-Q(j)||, \min(A[i-1, j-1], A[i, j-1], A[i-1, j])]$

• Time: $O(n^2)$

• Can be improved to $O(n^2 \log \log n / \log n)$ [Agarwal-Avraham-Kaplan-Sharir’12]
  (also [Buchin-Buchin-Meulemans-Mulzer’14])

• Many algorithms for special cases and variants
Dynamic Time Warping

- Definition:
  - $x, y$: two sequences of points of length $n$
  - $A[i, j]=\text{dist}(x_i, y_j)+\min(A[i-1,j], A[i-1,j-1], A[i,j-1])$
  - $\text{DTW}(x,y)=A[n,n]$

  **Find a walk along $x$ and $y$ that minimizes the sum of distances at each step.**

- Speech processing and other applications

- A simple $O(n^2)$ time dynamic programming algorithm
Longest Common Subsequence (LCS)

• Definition:
  – two sequences \( s \) and \( t \) of letters, length \( n \)
  – find a subsequence of both \( s \) and \( t \) of max length

• Example: \( \text{LCS}(\text{meaning}, \text{matching}) = \text{maing} \)

• Simple \( O(n^2) \) time algorithm:

\[
A[i, j] = \begin{cases} 
\max \{A[i-1, j], A[i, j-1], 1+A[i-1, j-1]\} & \text{if } s[i] = t[i] \\
\max \{A[i-1, j], A[i, j-1]\} & \text{otherwise.}
\end{cases}
\]
Edit distance
(a.k.a. Levenshtein distance)

• Definition:
  – \(x, y\) – two sequences of symbols of length \(n\)
  – \(\text{edit}(x, y)\) = the minimum number of symbol insertions, deletions or substitutions needed to transform \(x\) into \(y\)

• Example: \(\text{edit}(\text{meaning}, \text{matching}) = 4\)

\[
\begin{align*}
\text{meaning} & \xrightarrow{\text{insert } a} \text{maeaning} & \text{e} \rightarrow \text{t} & \text{mataning} \\
& & a \rightarrow c & \text{matching}
\end{align*}
\]
Computing edit distance

• A simple $O(n^2)$ time dynamic programming algorithm [Wagner-Fischer’74]

• Can be improved to $O(n^2 / \log n)$ [Masek-Paterson’80]

• Better algorithms for special cases: [U83, LV85, M86, GG88, GP89, UW90, CL90, CH98, LMS98, U85, CL92, N99, CPSV00, MS00, CM02, BCF08, AK08, AKO10...]

• Approximation algorithm: $(\log n)^{O(1/\varepsilon)}$ –approx in $O(n^{1+\varepsilon})$ time [Andoni-Krauthgamer-Onak’10]
What do these problems have in common?

- Widely used metrics
- Simple dynamic-programming algorithms with (essentially) quadratic running time
- We have no idea if/how we can do any better

Plausible explanation:
- 3SUM-hard? People tried for years...
- hard under OVC and SETH?
Plan

• Define sequence problems:
  – (Discrete) Frechet Distance
  – Edit Distance and LCS
  – Dynamic Time Warping (DTW)

• Birds eye view on the upper bounds
  – Dynamic programming, quadratic time

• Show conditional quadratic lower bounds
  – Assuming SETH / OVC
  – Basic approach
  – Hardness for LCS
Reminder: Orthogonal Vectors Conjecture

- **Orthogonal Vectors Problem (OV).** Given a set of vectors $S \subseteq \{0, 1\}^d$, $d = \omega(\log n)$, $|S| = n$, are there $a, b \in S$ s.t. $\sum_{i=1}^{d} a_i b_i = 0$?

  - Can be solved trivially in $O(n^2d)$ time
  - Best known algorithm runs in $n^{2-1/O(\log c(n))}$ time, where $d = c(n) \cdot \log n$ [Abboud-Williams-Yu’15]

- **OV Conjecture:**
  
  OV can’t be solved in $n^{2-\varepsilon} \cdot d^{O(1)}$ time for any $\varepsilon > 0$.

- Implied by SETH.
Quadratic hardness under OVC

Theorem*: No $n^{2-\Omega(1)}$ time algorithm for EDIT, DTW, Frechet distances or LCS unless OVC fails [Bringmann’14; Backurs-Indyk’15; Abboud-Backurs-VW’15; Bringmann-Kunnemann’15]

• Approach: reduce OV to distance computation:
  – $S \subseteq \{0,1\}^d \rightarrow$ sequence $x$, $|x| \leq n \cdot d^{O(1)}$
  – $S \subseteq \{0,1\}^d \rightarrow$ sequence $y$, $|y| \leq n \cdot d^{O(1)}$
  – distance$(x,y) =$ small if exists $a, b \in S$ with $\Sigma_i a_i b_i = 0$
  – distance$(x,y) =$ large, otherwise
  – The construction time is $n \cdot d^{O(1)}$
  – Gadgets for coordinates and vectors

*See also [Abboud-V. Williams-Weimann’14]

Next: hardness for LCS
Hardness for LCS

I will present the ideas behind the proof from [Abboud-Backurs-VW’15].

Full construction. NO full proof.

[Bringmann-Kunnemann’15] obtained an independent proof.
Idea for hardness for LCS

- Let $S = \{s_1, s_2, \ldots, s_n\}$ be the vectors of the OV instance
- Suppose we have $s_i \rightarrow$ gadget sequences $f(s_i)$ and $g(s_i)$
  \[
  \text{LCS}(f(s_i), g(s_j)) = \beta \text{ if } s_i \cdot s_j \neq 0, \geq \beta + 1 \text{ otherwise.}
  \]
- $s_0$ – vector of all 1s (no vector orthog. to $s_0$)

Attempt 1:

$x = f(s_1) f(s_2) \ldots f(s_i) \ldots f(s_n)$

$y = (g(s_0))^{n-1} g(s_1) g(s_2) \ldots g(s_j) \ldots g(s_n) (g(s_0))^{n-1}$

Idea: Imagine gadgets are letters.

If no OV pair, LCS length is $n \beta$;
if $s_i \cdot s_j = 0$ can align $f(s_i)$ and $g(s_j)$ and all other $f(s_k)$
with $g(s_0)$ to get LCS length $\geq (n-1) \beta + (\beta+1) > n \beta$.

Problem: Opt LCS might not align entire gadgets!

Force them to behave like letters!
Let $S = \{s_1, s_2, \ldots, s_n\}$ be the vectors
Each $s_i \rightarrow$ gadget sequences $f(s_i)$ and $g(s_i)$
LCS($f(s_i), g(s_j)$) = $\beta$ if $s_i \cdot s_j \neq 0$, $\geq \beta + 1$ otherwise.
$s_0$ – vector of all 1s (no vector orthog. to $s_0$)

**Idea for hardness for LCS**

**Attempt 2:** surround each gadget with $Q$ and $R$

$Q = 0^q$, $R = 1^q$, $q$ large

$x = Q f(s_1) R Q f(s_2) R \ldots Q f(s_n) R$

$y = (Q g(s_0) R)^{n-1} Q g(s_1) R Q g(s_2) R \ldots Q g(s_n) R (Q g(s_0) R)^{n-1}$

**Lemma:** If a 0 (or 1) is matched, its entire $0^q$ (or $1^q$) block is matched.
(proof: greedy argument)

**Idea:** Pick $q$ big so all $Q$s and $R$s of $x$ must be matched in an LCS.
Now no $g(s_k)$ is aligned with two different $f(s_i)$ and $f(s_j)$.

**Problem:** LCS might align $f(s_i)$ with several $g(s_k)$.
The $g(s_k)$ are partitioned into blocks aligned with at most a single $f(s_i)$. 

Proof:
Let $S = \{s_1, s_2, \ldots, s_n\}$ be the vectors. Each $s_i \rightarrow$ sequences $f(s_i)$ and $g(s_i)$.

$LCS(f(s_i), g(s_j)) = \beta$ if $s_i \cdot s_j \neq 0$, $\geq \beta + 1$ otherwise. $s_0$ – vector of all 1s (no vector orthog. to $s_0$)

### Attempt 3:

$x = P|y|Q f(s_1)R Q f(s_2)R Q \ldots RQ f(s_n) R P|y|$

$y = P (Q g(s_0) R P)^{n-1} Q g(s_1) R P \ Q g(s_2) R P \ldots \ Q g(s_n) R P (Q g(s_0) R P)^{n-1}$

**Idea:**

$P = 2^r$, $r$ big but $r<<q$, so that in an LCS all Qs and Rs of $x$ are still aligned, and also as many Ps as possible from $y$ are aligned.

$\geq n-1$ Ps of $y$ not matched in an LCS due to the matched Qs and Rs of $x$.

Thus, **exactly** $n-1$ Ps will be unmatched, and every $f(s_i)$ will be fully aligned with some $g(s_j)$ (possibly $j=0$).

**The gadgets** $f(s_i)$ and $g(s_j)$ act as letters!
Let $S = \{s_1, s_2, \ldots, s_n\}$ be the vectors
Each $s_i \rightarrow$ sequences $f(s_i)$ and $g(s_i)$
$LCS(f(s_i), g(s_j)) = \beta$ if $s_i \cdot s_j \neq 0$, $\geq \beta + 1$ otherwise.
$s_0$ – vector of all 1s (no vector orthog. to $s_0$)

**Attempt 3:**

$x = P|y|Q f(s_1) R Q f(s_2) R Q \ldots R Q f(s_n) R P|y|

$y = P (Q g(s_0) R P)^{n-1} Q g(s_1) R P \ Q g(s_2) R P \ldots \ Q g(s_n) R P (Q g(s_0) R P)^{n-1}$

**LCS length:**  
#Ps in $y$ is $3n-1$, and $n-1$ are not matched, so $2n$ aligned.

$2n |P| + n(|Q| + |R|) + \sum_{i=1}^{n} LCS(f(s_i), g(s_j))$, $g(s_j)$ aligned with $f(s_i)$

$= 2nr + 2qn + n \beta$  if no orthog. pair

$\geq [2nr + 2qn + n \beta] + 1$ if $\exists$ an orthog. pair.
Let $S = \{s_1, s_2, \ldots, s_n\}$ be the vectors
Each $s_i \rightarrow$ sequences $f(s_i)$ and $g(s_i)$
$LCS(f(s_i), g(s_j)) = \beta$ if $s_i \cdot s_j \neq 0$, $\geq \beta + 1$ otherwise.
$s_0$ – vector of all 1s (no vector orthog. to $s_0$)

**Reduction:**

\[
x = P^{|y|} Q f(s_1) R Q f(s_2) R Q \ldots R Q f(s_n) R P^{|y|}
\]
\[
y = P (Q g(s_0) R P)^{n-1} Q g(s_1) R P Q g(s_2) R P \ldots Q g(s_n) R P (Q g(s_0) R P)^{n-1}
\]

Tricky proof in paper shows the following suffice:

$|Q|, |R|, |P|, |f(s_i)|, |g(s_i)| \leq poly(d)$, so that

$|x|, |y| \leq n \text{ poly}(d)$.

Remains to construct the **vector gadgets** $f(s_i), g(s_i)$. 
Vector gadgets

Suppose that we have coordinate gadgets $x \in \{0, 1\} \rightarrow c(x)$ and $e(x)$, s.t. $\text{LCS}(c(x), e(y)) = 0$ if $x = y = 1$ and 1 otherwise; also, $|c(x)|, |e(x)| \leq 2$.

Then:

$$f(s_i) = 3^r \, 5^u \, c(s_i[1]) \, 5^u \ldots \, 5^u \, c(s_i[d]) \, 5^u$$

$$g(s_i) = 5^u \, e(s_i[1]) \, 5^u \ldots \, 5^u \, e(s_i[d]) \, 5^u$$

where $r = u(d+1)+d-1$, $u > d+1$.

Want: Each $s_i \rightarrow$ sequences $f(s_i)$ and $g(s_i)$ $\text{LCS}(f(s_i), g(s_j)) = \beta$ if $s_i \cdot s_j \neq 0$, $\geq \beta + 1$ ow.

If two $5$s are matched together, their entire $5^u$ blocks are matched.

If any $3$ is matched, no non-$3$ symbols are, so the LCS length is $r$.

If no $3$ is matched in an LCS, then all $5$s must be: if a $5^u$ block is not matched, then the subsequence length would be $\leq du + 2d < r$. 

$u$ is large, $r$ even larger
Vector gadgets

\[ f(s_i) = 3^r \Delta^u c(s_i[1]) \Delta^u \ldots \Delta^u c(s_i[d]) \Delta^u \]
\[ g(s_j) = 5^u \Delta^u e(s_j[1]) \Delta^u \ldots \Delta^u e(s_j[d]) \Delta^u 3^r \]

where \( r = u(d+1)+d-1, \ u > d. \)

Assume no 3 is matched. All 5s are matched. Thus, for all \( t, \) \( c(s_i[t]) \) and \( e(s_j[t]) \) are matched.

If \( s_i \cdot s_j \neq 0, \) the alignment of \( c(s_i[t]) \) with \( e(s_j[t]) \) for all \( t \) gives \( < d, \) so we get \( \leq (d+1)u+d-1 = r. \)

If \( s_i \cdot s_j = 0, \) we get \( (d+1)u+d > r. \)

Suppose that we have coordinate gadgets \( x \in \{0, 1\} \rightarrow c(x) \) and \( e(x), \) s.t.
\( \text{LCS}(c(x), e(y)) = 0 \) if \( x = y = 1 \) and 1 otherwise; also, \( |c(x)|, |e(x)| \leq 2. \)
Want: \( x \in \{0, 1\} \rightarrow c(x) \) and \( e(x) \) such that \( \text{LCS}(c(x), e(y)) \) is 0 if \( x = y =1 \) and 1 otherwise; also, \( |c(x)|, |e(x)| \leq 2 \).

\[
\begin{align*}
c(0) &= 46 & e(0) &= 64 \\
c(1) &= 4 & e(1) &= 6
\end{align*}
\]

\( \text{LCS}(c(1), e(1)) = 0 \), and
\( \text{LCS}(c(x), e(y)) = 1 \) for \( (x,y) \neq (1,1) \).
Recap of reduction

1. Use coordinate gadgets to make vector gadgets using extra symbols between them to make coordinate gadgets line up

2. Use vector gadgets to make sequences with some extra symbols between them to choose a vector pair and enforce alignment of vectors
Extensions

• **Thm**: For any integer $k \geq 2$, $k$-LCS cannot be solved in $O(n^{k-\varepsilon})$ time under SETH.

• [BK’15]: LCS hard even for binary alphabet

• Hardness based on even more believable assumptions:
  – Reduction works from Max-$k$-SAT, so base on: MAX-$k$-SAT cannot be solved in $2^{n(1-\varepsilon)}\text{poly}(n)$ time for all $k$.
    (although – maybe this is equivalent to SETH...)
  – On much more believable assumptions!
Circuit-Strong-ETH

- SETH is ultimately about SAT of linear size CNF-formulas
- There are more difficult satisfiability problems:
  - CIRCUIT-SAT
  - NC-SAT
  - NC1-SAT ...

**C-SETH**: satisfiability of circuits from circuit class C on n variables and size s requires $2^{n-o(n)} \text{ poly}(s)$ time.

E.g. NC-SETH should be much more believable!
LCS, Edit Distance and Friends are very hard

1. Edit Distance / LCS / ... require $n^{2-o(1)}$ time under NC-SETH.

2. Shaving logarithms from $n^2$ implies novel circuit lower bounds!

OV and APSP have such algs. W’14,AWY’15

An $\frac{n^2}{\log^{\omega(1)} n}$ alg. $\rightarrow$ $E^{NP}$ is not in NC1.

An $\frac{n^2}{\log^{1000} n}$ time alg. $\rightarrow$ $E^{NP}$ has no non-uniform Boolean formulas of size $n^5$.

Best alg: $\frac{n^2}{\log^2 n}$
Conclusion

The reductions to Frechet, Edit Distance and DTW are similar.

Some open problems:

- Hardness for approximating LCS, Edit Distance?
- Hardness of more than $\log^2 n$ speed-ups over $n^2$?
- What about walking $k$ dogs...?