Fine-Grained Complexity - Hardness in P

Lecture 1: SETH and OV

Karl Bringmann
Machine Model

Random Access Machine (RAM):

- memory of $O(1)$ cells
- can perform all reasonable operations on two cells in $O(1)$ time (arithmetic + logical operations…)
- can read/write any cell in $O(1)$ time

storage:

```
101 011 001 111  ...  100 101 111
```

cell / word consisting of $\Theta(\log n)$ bits

can recognize palindromes in time $O(n)$

the details do not matter!
Reminder: SETH and OV

Problem \textit{k-SAT}:
Given formula in \textit{k}-CNF with \( n \) variables, is it satisfiable?
\[ (x_1 \lor \neg x_2 \lor x_3) \land (x_2 \lor \neg x_3 \lor x_5) \land (\neg x_1 \lor x_4 \lor x_5) \]

\textbf{Strong Exponential Time Hypothesis:} \cite{ImpagliazzoPaturi01}
\[ \forall \varepsilon > 0 \exists k: \textit{k-SAT} \text{ has no } O(2^{(1-\varepsilon)n}) \text{-time algorithm} \]

Problem \textbf{Orthogonal Vectors}:  
Given sets \( A, B \subseteq \{0,1\}^d \) of size \( n \), 
are any \( a \in A, b \in B \) orthogonal? \( \sum_i a_i \cdot b_i = 0 \)

\textbf{OV-Hypothesis}: (moderate dimension)
\[ \forall \varepsilon > 0: \text{OV has no } O(n^{2-\varepsilon} \cdot \text{poly}(d)) \text{-time algorithm} \]

\textbf{LD-OV-Hypothesis}: (low dimension)
\[ \forall \varepsilon > 0 \exists c > 0: \text{OV in } d = c \log n \text{ has no } O(n^{2-\varepsilon}) \text{-time algorithm} \]
Reminder: Fine-Grained Reductions

transfer hardness of one problem to another one by reductions

problem $P$
instance $I$
size $n$

problem $Q$
instance $J$
size $s(n)$

$I$ is a ‘yes’-instance $\iff$ $J$ is a ‘yes’-instance

$t(n)$ algorithm for $Q$ implies a $r(n) + t(s(n))$ algorithm for $P$

if $P$ has no $r(n) + t(s(n))$ algorithm then $Q$ has no $t(n)$ algorithm
Reminder: Fine-Grained Reductions

A **fine-grained reduction** from \((P, T)\) to \((Q, T')\) is an algorithm \(A\) for \(P\) with **oracle** access to \(Q\) s.t.:

- For any instance \(I\), algorithm \(A(I)\) correctly solves problem \(P\) on \(I\)
- \(A\) runs in time \(r(n) = O(T(n)^{1-\gamma})\) for some \(\gamma > 0\)
- For any \(\varepsilon > 0\) there is a \(\delta > 0\) s.t. \(\sum_{i=1}^{k} T'(n_i)^{1-\varepsilon} \leq T(n)^{1-\delta}\)
Survey: On some fine-grained questions in algorithms and complexity, V. Vassilevska Williams
I. Easy Example

II. Longest Common Subsequence

III. Hardness of Approximation

IV. Further Topics

V. Summary
Landscape of Polytime Problems

SETH-hard

- k-DomSet $n^k$
- Frechet $n^2$
- Diameter $n^2$
- LCS $n^2$
- Longest Palindromic Subsequence $n^2$

SETH

- SAT $2^n$
- SubsetSum $n + t$
- Dynamic Time Warping $n^2$
- NFA-Acceptance $n^2$
- RegExp Matching $n^2$

OV $n^2$

- SubsetSum $n + t$
- Dynamic Time Warping $n^2$
- NFA-Acceptance $n^2$
- RegExp Matching $n^2$

3SUM-hard

- X+Y $n^2$
- Colinear $n^2$
- GeomBase $n^2$
- Separator $n^2$
- PlanarMotion Planning $n^2$

3SUM $n^2$

- GeomBase $n^2$
- Colinear $n^2$
- Separator $n^2$
- PlanarMotion Planning $n^2$

APSP-hard

- APSP $n^3$
- Radius $n^3$
- Metricity $n^3$
- Betweenness Centrality $n^3$
- Tree Edit Distance $n^3$
- NegTriangle $n^3$

APSP $n^3$

- APSP $n^3$
- Radius $n^3$
- Metricity $n^3$
- Betweenness Centrality $n^3$
- Tree Edit Distance $n^3$
- NegTriangle $n^3$

[Patrascu, Williams’10]

[Williams’05]

[B’14]

[V-Williams,Roditty’13]

[B,Künkemann’15, Abboud, Backurs, V-Williams’15]

[B,Künkemann’15]
NFA Acceptance Problem

nondeterministic finite automaton $G$ accepts input string $s$ if there is a walk in $G$ from starting state to some accepting state, labelled with $s$

dynamic programming algorithm in time $O(|s||G|)$:

$$T[i] := \text{set of states reachable via walks labelled with } s[1..i]$$

$$T[0] := \{\text{starting state}\}$$

$$T[i] := \{v \mid \exists u \in T[i - 1] \text{ and } \exists \text{ transition } u \rightarrow v \text{ labelled } s[i]\}$$

string: 01011010

![NFA Diagram]
OV-Hardness Result

OV sets $A, B \subseteq \{0,1\}^d$ of size $n$

reduction

time $O(dn)$

NFA acceptance grammar $G$, string $s$

$|G| = O(dn)$

$|s| = O(dn)$

$O(n^{2-\varepsilon}\text{poly}(d))$ algorithm $\iff$ $O((|s| |G|)^{1-\varepsilon})$ algorithm

Thm: NFA acceptance has no $O((|s| |G|)^{1-\varepsilon})$ algorithm unless OVH fails. [Impagliazzo]
Proof:

for any $a \in A$:
construct string:
\[
0011
\]
\[
= a_1 a_2 \ldots a_d
\]

for any $b \in B$:
construct part of NFA:
\[
\text{if } b_i = 1
\]
\[
\text{if } b_i = 0
\]
Proof:

OV

sets \( A, B \subseteq \{0,1\}^d \)

of size \( n \)

Proof:

for any \( a \in A \):

construct string:

\[
0011
\]

\[
= a_1a_2 \ldots a_d
\]

for any \( b \in B \):

construct part of NFA:

if \( b_i = 1 \)

if \( b_i = 0 \)

NFA acceptance

grammar \( G \), string \( s \)

\[
|G| = O(dn)
\]

\[
|s| = O(dn)
\]
Proof

for any \( a \in A \):

construct string:

\[ 0011 \]

for any \( b \in B \):

construct part of NFA:

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 \\
\end{array}
\]

string \( s = 11000110 \ldots 00110 \) (for all \( a \in A \))

- equivalent to OV instance
- size \(|s| = |G| = O(dn)|

NFA \( G \):

\[
\begin{array}{cccc}
0,1,\& & 0,1,\& \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
0,1,\& & 0,1,\& \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
0,1,\& & 0,1,\& \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
0,1,\& & 0,1,\& \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]
OV-Hardness Result

OV
sets $A, B \subseteq \{0,1\}^d$
of size $n$

NFA acceptance
grammar $G$, string $s$
$|G| = O(dn)$
$|s| = O(dn)$

time $O(dn)$

$O(n^{2-\varepsilon}\text{poly}(d))$ algorithm \iff $O((|s| |G|)^{1-\varepsilon})$ algorithm

Thm: NFA acceptance has no $O((|s| |G|)^{1-\varepsilon})$
algorithm unless OVH fails.

[Impagliazzo]
I. Easy Example

II. Longest Common Subsequence

III. Hardness of Approximation

IV. Further Topics

V. Summary
Landscape of Polytime Problems

SETH-hard

SAT $2^n$

- $k$-DomSet $n^k$
  - [Patrascu, Williams’10]
  - [B’14]
- Frechet $n^2$
  - [V-Williams, Roditty’13]
- Diameter $n^2$
  - [B, Künne mann’15, Abboud, Backurs, V-Williams’15]
- LCS $n^2$
  - [B, Künne mann’15]  
  - [V-Williams, Roditty’13]
- Longest Palindromic Subsequence $n^2$
  - [B, Künne mann’15]
- OV $n^2$
  - [Williams’05]
- SubsetSum $n + t$
  - [Abboud, B, Hermelin, Shabtay’17+]
  - [Backurs, Indyk’15]
- Dynamic Time Warping $n^2$
  - [Impagliazzo]
  - [Backurs, Indyk’16]
- Edit Distance $n^2$
  - [Backurs, Indyk’15]
- RegExp Matching $n^2$
  - [Backurs, Indyk’16]

3SUM-hard

3SUM $n^2$

- X+Y $n^2$
  - [Gajentaan, Overmars’98]
- GeomBase $n^2$
- Colinear $n^2$
- Separator $n^2$
- Planar Motion Planning $n^2$

APSP-hard

APSP $n^3$

- Radius $n^3$
  - [V-Williams, Williams’10]
- Metricity $n^3$
- Betweenness Centrality $n^3$
- NegTriangle $n^3$
- Maximum Submatrix $n^3$
  - [B, Gawrychowski, Mozes, Weimann’18]
- Tree Edit Distance $n^3$
  - [Backurs, Dikkala, Tzamos’16]

SAT $2^n$

- SubsetSum $n + t$
- Dynamic Time Warping $n^2$
- Edit Distance $n^2$
- RegExp Matching $n^2$

Patrascu, Williams’10
Williams’05
B, Künnemann’15
Abboud, Backurs, V-Williams’15
Longest Common Subsequence (LCS)

given strings $x, y$ of length $n \geq m$, compute longest string $z$ that is a subsequence of both $x$ and $y$

natural dynamic program $O(n^2)$

```
\begin{array}{c|cccc}
|   & x[1] & \ldots & x[n] \\
\hline
y[1] & & & & \\
\vdots & & & & \\
y[m] & & & & \\
\end{array}
```

$T[i, j] = \max\{T[i - 1, j], T[i, j - 1]\}$

if $x[i] = y[j]$:

$T[i, j] = \max\{T[i, j], T[i - 1, j - 1] + 1\}$

write $LCS(x, y) = |z|$

logfactor improvement:

$O(n^2 / \log^2 n)$

[Masek, Paterson'80]

→ edit distance
OV-Hardness of LCS

**OV**
- sets $A, B \subseteq \{0,1\}^d$ of size $n$
- $\exists a \in A, b \in B: \forall i: a_i \cdot b_i = 0$?

**reduction**
- time $O(d^2 n)$

**LCS**
- strings $x, y$
- of length $O(d^2 n)$

$O(n^{2-\varepsilon}\text{poly}(d))$ algorithm \iff $O(n^{2-\varepsilon})$ algorithm

**Thm:** Longest Common Subsequence has no $O(n^{2-\varepsilon})$ algorithm unless the OV-Hypothesis fails.

[B., Künemann'15] Abboud, Backurs, V-Williams'15
Proof: Coordinate Gadgets

OV: Given $A, B \subseteq \{0,1\}^d$ of size $n$ each
Are there $a \in A, b \in B$ such that $\forall i: a_i \cdot b_i = 0$

we want to simulate the coordinates $\{0,1\}$ and the behavior of $a_i \cdot b_i$

replace $a_i$ by $a_i^A$ and $b_i$ by $b_i^B$

$LCS(a_i^A, b_i^B)$ can be written as $f(a_i \cdot b_i)$, with $f(0) > f(1)$
Proof: Vector Gadgets

**OV:** Given $A, B \subseteq \{0,1\}^d$ of size $n$ each
Are there $a \in A, b \in B$ such that $\forall i: a_i \cdot b_i = 0$

we want to simulate **orthogonality** of $a \in A, b \in B$
concatenate $a_1^A, \ldots, a_d^A$, padded with a new symbol 2

\[
VG(a) := a_1^A 2 \ldots 2 a_2^A 2 \ldots 2 a_3^A 2 \ldots 2 a_4^A
\]
\[
VG(b) := b_1^B 2 \ldots 2 b_2^B 2 \ldots 2 b_3^B 2 \ldots 2 b_4^B
\]

- no LCS matches symbols in $a_i^A$ with symbols in $b_j^B$ where $i \neq j$

in the picture: $d = 4$

length $4d$
Proof: Vector Gadgets

OV: Given \( A, B \subseteq \{0,1\}^d \) of size \( n \) each. Are there \( a \in A, b \in B \) such that \( \forall i: a_i \cdot b_i = 0 \)

we want to simulate orthogonality of \( a \in A, b \in B \)

concatenate \( a_1^A, \ldots, a_d^A \), padded with a new symbol 2

\[
VG(a) := a_1^A 2 \ldots 2 a_2^A 2 \ldots 2 a_3^A 2 \ldots 2 a_4^A
\]

\[
VG(b) := b_1^B 2 \ldots 2 b_2^B 2 \ldots 2 b_3^B 2 \ldots 2 b_4^B
\]

- no LCS matches symbols in \( a_i^A \) with symbols in \( b_j^B \) where \( i \neq j \)

lose one block of 2's

cannot make up for it with symbols 0/1 since there are to few of them
Proof: Vector Gadgets

Given \( A, B \subseteq \{0,1\}^d \) of size \( n \) each.

Are there \( a \in A, b \in B \) such that \( \forall i: a_i \cdot b_i = 0 \)

we want to simulate **orthogonality** of \( a \in A, b \in B \)

concatenate \( a_1^A, \ldots, a_d^A \), padded with a new symbol 2

\[
VG(a) := a_1^A 2 \ldots 2 a_2^A 2 \ldots 2 a_3^A 2 \ldots 2 a_4^A
\]

\[
VG(b) := b_1^B 2 \ldots 2 b_2^B 2 \ldots 2 b_3^B 2 \ldots 2 b_4^B
\]

- no LCS matches symbols in \( a_i^A \) with symbols in \( b_j^B \) where \( i \neq j \)
- some LCS matches all 2’s
Proof: Vector Gadgets

**OV:** Given $A, B \subseteq \{0,1\}^d$ of size $n$ each
Are there $a \in A, b \in B$ such that $\forall i: a_i \cdot b_i = 0$

we want to simulate **orthogonality** of $a \in A, b \in B$
concatenate $a_1^A,\ldots,a_d^A$, padded with a new symbol 2

$$VG(a) := a_1^A 2 \ldots 2 a_2^A 2 \ldots 2 a_3^A 2 \ldots 2 a_d^A$$

$$VG(b) := b_1^B 2 \ldots 2 b_2^B 2 \ldots 2 b_3^B 2 \ldots 2 b_d^B$$

$$-LCS(VG(a),VG(b)) = (d-1)4d + \sum_{i=1}^{d} LCS(a_i^A,b_i^B)$$

= $f(a_i \cdot b_i)$

#2's

$LCS(VG(a),VG(b)) = C + 2$ if $a \perp b$

$LCS(VG(a),VG(b)) \le C$ otherwise

for some constant $C$
Proof: Normalized Vectors Gadgets

**OV:** Given $A, B \subseteq \{0,1\}^d$ of size $n$ each. Are there $a \in A, b \in B$ such that $\forall i: a_i \cdot b_i = 0$

**new vector gadgets:**

$VG'(a):$  
$VG'(b):$  

$LCS(VG'(a), VG'(b)) = \max\{LCS(VG(a), VG(b)), C\}$

$LCS(VG'(a), VG'(b)) = \begin{cases}  C + 2 & \text{if } a \perp b \\  C & \text{otherwise} \end{cases}$

write $VG$ for $VG'$
Proof: OR-Gadget

OV: Given $A, B \subseteq \{0,1\}^d$ of size $n$ each

Are there $a \in A, b \in B$ such that $\forall i: a_i \cdot b_i = 0$.

fresh symbol 4, want to construct:

in the picture: $n = 3$

$VG(A[1]) 4 \ldots 4 VG(A[2]) 4 \ldots 4 VG(A[3]) 4 \ldots 4 VG(A[1]) 4 \ldots 4 VG(A[2]) 4 \ldots 4 VG(A[3])$

$4 \ldots \ldots \ldots \ldots 4 VG(B[1]) 4 \ldots 4 VG(B[2]) 4 \ldots 4 VG(B[3]) 4 \ldots \ldots \ldots \ldots 4$

length $100d^2$

length $100d^2 \cdot 2n$
Proof: OR-Gadget

OV: Given $A, B \subseteq \{0,1\}^d$ of size $n$ each.
Are there $a \in A, b \in B$ such that $\forall i: a_i \cdot b_i = 0$

fresh symbol 4, want to construct:

in the picture: $n = 3$

$V(G(A[1])) 4 \ldots 4 V(G(A[2])) 4 \ldots 4 V(G(A[3])) 4 \ldots 4 V(G(A[1])) 4 \ldots 4 V(G(A[2])) 4 \ldots 4 V(G(A[3]))$

$4 \ldots \ldots \ldots 4 V(G(B[1])) 4 \ldots 4 V(G(B[2])) 4 \ldots 4 V(G(B[3])) 4 \ldots \ldots \ldots \ldots 4$

can align $V(G(B[j]))$ with $V(G(A[\Delta + j \mod n]))$ for any offset $\Delta$

$LCS \geq (2n - 1)100d^2 + \max_{\Delta} \sum_{j=1}^{n} LCS(V(G(A[\Delta + j \mod n])), V(G(B[j])))$

#4's in upper string
maximize over offset
need normalization!

If there is an orthogonal pair, some offset $\Delta$ aligns this pair, and we get

$LCS \geq (2n - 1)100d^2 + nC + 2$
Proof: OR-Gadget

**OV:** Given $A, B \subseteq \{0,1\}^d$ of size $n$ each, are there $a \in A, b \in B$ such that $\forall i: a_i \cdot b_i = 0$

fresh symbol 4, want to construct: in the picture: $n = 3$

If an orthogonal pair exists then $LCS \geq (2n - 1)100d^2 + nC + 2$

**Claim:** otherwise: $LCS \leq (2n - 1)100d^2 + nC$

this finishes the proof: - equivalent to OV instance
- length $O(d^2n)$
Proof of Claim

**OV:** Given \( A, B \subseteq \{0,1\}^d \) of size \( n \) each

Are there \( a \in A, b \in B \) such that \( \forall i: a_i \cdot b_i = 0 \)

**Claim:** if no orthogonal pair exists:  \( LCS \leq (2n - 1)100d^2 + nC \)

Consider how an LCS matches the \( VG(B[j]) \)

- no crossings
Proof of Claim

Given $A, B \subseteq \{0,1\}^d$ of size $n$ each
Are there $a \in A, b \in B$ such that $\forall i: a_i \cdot b_i = 0$

Claim: if no orthogonal pair exists: $LCS \leq (2n - 1)100d^2 + nC$

$VG(A[1]) 4 \ldots 4 VG(A[2]) 4 \ldots 4 VG(A[3]) 4 \ldots 4 VG(A[1]) 4 \ldots 4 VG(A[2]) 4 \ldots 4 VG(A[3])$

$4 \ldots \ldots \ldots \ldots 4 VG(B[1]) 4 \ldots 4 VG(B[2]) 4 \ldots 4 VG(B[3]) 4 \ldots \ldots \ldots \ldots 4$

$LCS \leq (2n - 1)100d^2 + \sum_{j=1}^{n} C$

$\sum_{j=1}^{n} (0$ if $VG(B[j])$ is not matched $\leq 0$

if $VG(B[j])$ is matched to one

$|VG(B[j])| - |4 \ldots 4|$ if $VG(B[j])$ is matched to $> 1$

#4's in upper string

could match VG completely, but lose many 4's
OV-Hardness of LCS

OV
sets $A, B \subseteq \{0,1\}^d$
of size $n$

LCS
strings $x, y$
of length $O(d^2n)$

reduction
time $O(d^2n)$

$O(n^{2-\varepsilon}\text{poly}(d))$ algorithm $\iff$ $O(n^{2-\varepsilon})$ algorithm

Thm: Longest Common Subsequence has no $O(n^{2-\varepsilon})$ algorithm unless the OV-Hypothesis fails.

[B., Künemann'15 Abboud, Backurs, V-Williams'15]
Extensions

**similar problems:** edit distance, dynamic time warping, …  
[Backurs, Indyk’15]  [B., Künemann’15, Abboud, Backurs, V-Williams’15]

**harder hardness:** reductions from Formula-SAT / branching programs  
[Abboud, Hansen, V-Williams, Williams’16]  [Abboud, B.’18]

**LCS-hard problems:** longest palindromic subseq., longest tandem subseq.  
[B., Künemann’15]

**alphabet size:**  
[B., Künemann’15]

  LCS and edit distance are hard on *binary* strings, i.e., alphabet \{0,1\}

**k-LCS:** LCS of \(k\) strings takes time \(\Omega(n^{k-\varepsilon})\)  
[Abboud, Backurs, V-Williams’15]

Based on \(k\)-OV
Landscape of Polytime Problems

- SAT $2^n$
  - k-DomSet $n^k$
    - [Patrascu, Williams’10]
    - [Williams’05]
  - Frechet $n^2$
    - [B’14]
    - [V-Williams, Roditty’13]
  - Diameter $n^2$
    - [B, Künne, Abboud, Backurs, V-Williams’15]
  - LCS $n^2$
    - [B, Künne, Abboud, Backurs, V-Williams’15]
  - Longest Palindromic Subsequence $n^2$
    - [B, Künne, Abboud, Backurs, V-Williams’15]
    - [B, Künne, Abboud, Backurs, V-Williams’15]

- SubsetSum $n + t$
  - [Abboud, B, Hermelin, Shabtay’17+]
  - [Backurs, Indyk’15]
  - [Impagliazzo]

- OV $n^2$
  - Dynamic Time Warping $n^2$
    - [Impagliazzo]
  - Edit Distance $n^2$
    - [Backurs, Indyk’15]

- SETH-hard
  - SAT $2^n$
  - 3SUM-hard
    - 3SUM $n^2$
      - X+Y $n^2$
      - Colinear $n^2$
      - GeomBase $n^2$
      - Separator $n^2$
      - PlanarMotion Planning $n^2$

- APSP-hard
  - APSP $n^3$
    - Radius $n^3$
    - Metricity $n^3$
    - Betweenness Centrality $n^3$
    - NegTriangle $n^3$
  - Maximum Submatrix $n^3$
    - [B, Gawrychowski, Mozes, Weimann’18]
  - Tree Edit Distance $n^3$
    - [Backurs, Dikkala, Tzamos’16]

- [V-Williams, Williams’10]

- [B, Gajentaan, Overmars’98]
I. Easy Example

II. Longest Common Subsequence

III. Hardness of Approximation

IV. Further Topics

V. Summary
Hardness of Approximation in P

„deterministic subquadratic-time approximation algorithms imply circuit lower bounds“

e.g. Longest Common Subsequence

„subquadratic-time approximation algorithms violate SETH“

e.g. Max-Inner-Product

„FPT-approximation algorithms violate SETH“

Dominating Set
Problem Max-Inner-Product:
Given sets $A, B \subseteq \{0,1\}^d$ of size $n$, compute $\max_{a \in A, b \in B} \langle a, b \rangle$.

"optimization version of OV"

Is in time $O(n^2 d)$, but not in time $O(n^{2-\varepsilon} \cdot \text{poly}(d))$ assuming OV-H.

**Thm:** Let $\log n \ll d \leq n^{o(1)}$. [Chen CCC’18]

1) $\forall \delta > 0 \exists \varepsilon > 0$: A $(d/\log n)^\delta$-multiplicative approximation to Max-IP can be computed in time $O(n^{2-\varepsilon} \cdot \text{poly}(d))$.

2) If $\exists \varepsilon > 0 \forall \delta > 0$: a $(d/\log n)^\delta$-multiplicative approximation to Max-IP can be computed in time $O(n^{2-\varepsilon} \cdot \text{poly}(d))$, then LD-OVH fails.

**tight hardness of approximation!**
Tool: Communication Complexity

Input: $x \in \{0,1\}^d$

Input: $y \in \{0,1\}^d$

want to compute $f(x, y)$

with as few communication as possible

e.g. $f = \text{Disj}$:

$$\text{Disj}(x, y) = \begin{cases} 
1 & \text{if } \langle x, y \rangle = 0 \\
0 & \text{otherwise}
\end{cases}$$

deterministic communication complexity of $\text{Disj}$ is $\Theta(d)$

randomized communication complexity of $\text{Disj}$ is $\Theta(d)$
Communication Complexity Setup

non-deterministic randomized one-way communication:

In: $x \in \{0,1\}^d$
accept/reject

In: $y \in \{0,1\}^d$

Merlin: powerful, untrusted, sends advice, knows $x, y$

Messages $M, R, B$ of bitlength $m, r, b$

We require:

Efficiency: Alice and Bob are efficiently computable

Completeness: If $f(x, y) = 1$:
  $\exists M$: $\Pr_R[\text{Alice accepts}] = 1$

Soundness: If $f(x, y) = 0$:
  $\forall M$: $\Pr_R[\text{Alice accepts}] \leq p$

Then we say that $f$ has an $(m, r, b, p)$-efficient protocol
From CC to Reductions

Any \((m, r, b, p)\)-efficient protocol for \(\text{Disj}\) yields a reduction:

- **OV** 
  - \(n\) 
  - \(d\)

Constructs \(2^m\) instances of Max-IP with gap \(p\) in time \(O(2^m \cdot n \cdot \text{poly}(d))\)

- **Max-IP**
  - \(n' = n\)
  - \(d' = 2^{r+b}\)

**YES-instance** \(\Rightarrow\) Max-IP \(\geq 2^r\) for some instance

**NO-instance** \(\Rightarrow\) Max-IP \(\leq p \cdot 2^r\) for all instances

**Complete:** If \(f(x, y) = 1\):
- \(\exists M: \Pr_R[\text{Alice accepts}] = 1\)

**Sound:** If \(f(x, y) = 0\):
- \(\forall M: \Pr_R[\text{Alice accepts}] \leq p\)
From CC to Reductions

Any \((m, r, b, p)\)-efficient protocol for \textit{Disj} yields a reduction:

\begin{align*}
\text{OV} \quad \text{n} \quad \text{d} \\
\text{constructs } 2^m \text{ instances of Max-IP with gap } p \\
\text{time } O(2^m \cdot n \cdot \text{poly}(d))
\end{align*}

\textbf{Max-IP}

\begin{align*}
\text{Max-IP} \\
\text{n}' = n \\
\text{d}' = 2^{r+b}
\end{align*}

\textbf{Proof:} Given OV instance \(X, Y \subseteq \{0, 1\}^d\)

Iterate over all advice strings \(M \in [2^m]:\)

Construct Max-IP instance \(X^M, Y^M:\)

\begin{align*}
\forall x \in X: \quad \forall R \in [2^r]: \text{create } x^{M,R} \in \{0, 1\}^{2^b} \text{ with } \\
\quad x_{B}^{M,R} = \text{whether Alice accepts on input } x, \text{ advice } M, \\
\quad \text{randomness } R \text{ and Bob's message } B
\end{align*}

\begin{align*}
\forall y \in Y: \quad \forall R \in [2^r]: \text{create } y^{M,R} \in \{0, 1\}^{2^b} \text{ with } \\
\quad y_{B}^{M,R} = \text{whether Bob sends message } B \text{ on input } y, \\
\quad \text{advice } M \text{ and randomness } R
\end{align*}

Then \(\langle x^{M,R}, y^{M,R} \rangle = \text{whether Alice accepts on inputs } x, y, \text{ advice } M, \text{ randomness } R\)
From CC to Reductions

Any \((m, r, b, p)\)-efficient protocol for \(Disj\) yields a reduction:

\[
\begin{align*}
\text{OV} & \quad \text{constructs } 2^m\text{ instances of Max-IP with gap } p \\
\text{OV} & \quad \text{time } O(2^m \cdot n \cdot \text{poly}(d))
\end{align*}
\]

**Proof:** Given OV instance \(X, Y \subseteq \{0,1\}^d\)

Iterate over all advice strings \(M \in [2^m]\):

Construct Max-IP instance \(X^M, Y^M\):

\[
\begin{align*}
\forall x \in X: & \quad \text{create } x^M \text{ by concatenating all } x^{M,R} \\
\forall y \in Y: & \quad \text{create } y^M \text{ by concatenating all } y^{M,R}
\end{align*}
\]

Then \(\langle x^M, y^M \rangle = 2^r \cdot \Pr_R [\text{Alice accepts on inputs } x, y \text{ and advice } M]\)

\[
\begin{align*}
\text{Max-IP} & \quad n' = n \\
d' = 2^{r+b} & \quad \text{time } O(2^m \cdot n \cdot \text{poly}(d))
\end{align*}
\]

**Complete:** If \(f(x, y) = 1\):

\(\exists M: \Pr_R [\text{Alice accepts}] = 1\)

**Sound:** If \(f(x, y) = 0\):

\(\forall M: \Pr_R [\text{Alice accepts}] \leq p\)

Alice and Bob efficient

Then \(\langle x^{M,R}, y^{M,R} \rangle = \text{whether Alice accepts on inputs } x, y, \text{ advice } M, \text{ randomness } R\)
From CC to Reductions

Any \((m, r, b, p)\)-efficient protocol for \(Disj\) yields a reduction:

- **OV** constructs \(2^m\) instances of Max-IP with gap \(p\)
- time \(O(2^m \cdot n \cdot \text{poly}(d))\)

**Max-IP**

\[
\begin{align*}
n' &= n \\
d' &= 2^{r+b}
\end{align*}
\]

**Proof:** Given OV instance \(X, Y \subseteq \{0,1\}^d\)

Iterate over all advice strings \(M \in [2^m]\):

- Construct Max-IP instance \(X^M, Y^M\):
  - \(\forall x \in X:\) create \(x^M\) by concatenating all \(x^{M,R}\)
  - \(\forall y \in Y:\) create \(y^M\) by concatenating all \(y^{M,R}\)

Then \(\langle x^M, y^M \rangle = 2^r \cdot \Pr_R[\text{Alice accepts on inputs } x, y \text{ and advice } M]\)

If \(\exists x \in X, y \in Y:\) \(\langle x, y \rangle = 0\) then:

\(\exists M: \langle x^M, y^M \rangle = 2^r\)

Otherwise:

\(\forall x \in X, y \in Y:\) \(\forall M: \langle x^M, y^M \rangle \leq p \cdot 2^r\)

**Complete:** If \(f(x, y) = 1:\)

\(\exists M: \Pr_R[\text{Alice accepts}] = 1\)

**Sound:** If \(f(x, y) = 0:\)

\(\forall M: \Pr_R[\text{Alice accepts}] \leq p\)
From CC to Reductions

Best known protocol for Disj:

For any $d, \alpha$ and $p < 1/2$:

$$\left(\frac{d}{\alpha}, \log_2 d + O\left(\log \frac{1}{p}\right), \text{poly}(\alpha) \cdot \log \frac{1}{p}, p\right)$$-efficient protocol

[Chen CCC’18]

Complicated: algebraic codes, expander mixing lemma, ...

Easier:

$$(m, r, b, p) = \left(o(d), o(d), o(d) + \frac{1}{2}\right)$$-efficient protocol

[Aaronson, Wigderson’09]

Implies reduction:

OV
$n$
$d = c \log n$

constructs $n^{o(1)}$ instances of Max-IP with gap $1/2$
time $O(n^{1+o(1)} \cdot \text{poly}(d))$

Max-IP

$n' = n$
$d' = n^{o(1)}$

$2^m, 2^{r+b} = n^{o(1)}$

[Abboud, Rubinstein, Williams’17]

So a $1.99$-approximation of Max-IP in dimension $d = n^{o(1)}$ in time $O(n^{2-\varepsilon})$ violates LD-OVH
The Disj-Protocol [Aaronson, Wigderson'09]

\[(m, r, b, p) = \left(\tilde{O}(\sqrt{d}), O(\log d), \tilde{O}(\sqrt{d}), \frac{1}{2}\right)\text{-efficient protocol}\]

**Proof:** Fix prime \( p \in (4d, 8d] \), \( T := \sqrt{d} \), assume that \( T \in \mathbb{N} \)

Given \( x, y \) compute polynomials of degree < \( d/T \) over \( \mathbb{Z}_p \):

\[\psi_{x,t}(Z) \text{ s.t. } \psi_{x,t}(i) = x_{i \cdot T + t} \text{ for } 0 \leq i < d/T\]

\[\psi_{y,t}(Z) \text{ s.t. } \psi_{y,t}(i) = y_{i \cdot T + t} \text{ for } 0 \leq i < d/T\]

Define \( \Phi(Z) := \sum_{t=1}^{T} \psi_{x,t}(Z) \cdot \psi_{y,t}(Z) \) of degree < \( 2d/T \)

**Protocol:**

**randomness:** \( R \in \mathbb{Z}_p \) uniformly at random

**Bob:** sends \( \psi_{y,1}(R), \ldots, \psi_{y,T}(R) \)

**Merlin:** sends polynomial \( P(Z) \) of degree < \( 2d/T \) over \( \mathbb{Z}_p \)

„good proof“: \( P(Z) = \Phi(Z) \)

**Alice:** accepts iff \( P(i) = 0 \) for all \( 0 \leq i < d/T \) and

\[P(R) = \sum_{t=1}^{T} \psi_{x,t}(R) \cdot \psi_{y,t}(R)\]

Alice and Bob efficient

**Complete:** If \( f(x, y) = 1 \):

\( \exists M: \Pr_R[\text{Alice accepts}] = 1 \)

**Sound:** If \( f(x, y) = 0 \):

\( \forall M: \Pr_R[\text{Alice accepts}] \leq p \)

\( x, y \) disjoint iff

\( \Phi(i) := 0 \) for all \( 0 \leq i < d/T \)
Problem Max-Inner-Product:
Given sets $A, B \subseteq \{0,1\}^d$ of size $n$,
compute $\max_{a \in A, b \in B} \langle a, b \rangle$

"optimization version of OV"

Is in time $O(n^2d)$, but not in time $O(n^{2-\varepsilon} \cdot \text{poly}(d))$ assuming OV-H

Thm: Let $\log n \ll d \leq n^{o(1)}$.

1) $\forall \delta > 0 \exists \varepsilon > 0$: A $(d/\log n)^{\delta}$-multiplicative approximation
to Max-IP can be computed in time $O(n^{2-\varepsilon} \cdot \text{poly}(d))$

2) If $\exists \varepsilon > 0 \forall \delta > 0$: a $(d/\log n)^{\delta}$-multiplicative approximation
to Max-IP can be computed in time $O(n^{2-\varepsilon} \cdot \text{poly}(d))$,
then LD-OVH fails
I. Easy Example

II. Longest Common Subsequence

III. Hardness of Approximation

IV. Further Topics

V. Summary
Mutivariate Algorithms for LCS

parameters for LCS: [B.,Künnemann’18]

\[ n = |x| \quad \text{.. length of longer string} \]
\[ m = |y| \quad \text{.. length of shorter string} \]
\[ L = LCS(x, y) \quad \text{.. length of LCS} \]
\[ |\Sigma| \quad \text{.. size of alphabet } \Sigma \]
\[ \Delta = n - L \quad \text{.. number of deletions in } x \]
\[ \delta = m - L \quad \text{.. number of deletions in } y \]
\[ M \quad \text{.. number of matching pairs} \]
\[ d \quad \text{.. number of dominant pairs} \]

multivariate algorithms: \( \tilde{O}(n + \min\{d, \delta m, \delta \Delta\}) \)

under SETH, this is optimal for any relations \( m = \Theta(n^{\alpha_m}), L = \Theta(n^{\alpha_L}), \ldots \)
Grammar-Compressed Strings

String $T$ given by **Straight-Line Program**: [Abboud, Backurs, B., Künnemann’17]

Nonterminals $S_1, \ldots, S_n$

Each $S_i$ is associated with a rule:

$S_i \rightarrow c$ \quad for some alphabet symbol/terminal $c$

or

$S_i \rightarrow S_\ell S_r$ \quad for some $\ell, r < i$

$T$ is the string generated by $S_n$

**compressed size** $n = \text{number of nonterminals}$

**uncompressed size** $N = \text{length of } T$

$N \geq n$, and $N$ can be as large as $2^{\Omega(n)}$

Computing the LCS-length of strings $x, y$ of length $N$ compressed to size $n$ is in time $\tilde{O}(Nn)$... [Tiskin’08] [Gawrychowski’12]

... and not in $O((Nn)^{1-\varepsilon})$ time assuming SETH [Abboud, Backurs, B., Künnemann’17]
Regular Expression Matching

describe patterns using the operations:

- concatenation $\circ$
- alternation $|$ (OR)
- Kleene star $\ast$ (repetition with empty string)
- Kleene plus $+$ (repetition without empty string)

starting from alphabet symbols in $\Sigma$

great expressiveness: describe regular languages

standard tool to describe pattern matching problems on strings
Regular Expression Matching

Reg-Exp Pattern Matching:
given reg-exp \( R \) and string \( S \), does some substring of \( S \) match \( R \)?

Reg-Exp Membership Testing:
given reg-exp \( R \) and string \( S \), does \( S \) match \( R \)?

\( O(nm) \)  
[Thompson’68]

\((nm)^{1-o(1)}\) under SETH  
[Backurs, Indyk’16]

\( O(nm) \)

\((nm)^{1-o(1)}\) under SETH

\( n = |S|, m = |R| \)
Homogeneous Regular Expressions

reg-exp is **homogeneous of type** \( t \) if:

- any internal node on level \( i \) has operation \( t_i \)
- the depth is at most \( |t| \)
- leaves can be at any level

restrict to homogeneous reg-exps of type \( t \):

- \( t \)-Pattern Matching, \( t \)-Membership Testing
**t-Pattern Matching and t-Membership Testing**

complete characterization!

for any type: **near-linear time** or **SETH-hard**

only exception: **Word Break** problem

\[
\text{string matching } O(n + m) \quad \text{whole subtree } O(n + m) \\
\text{simplifies } O(n + m) \\
\]

\[
\text{superset matching } O(n \log^2 m + m) \\
\text{dictionary matching } O(n + m) \\
\text{simplifies } O(n + m) \\
\]

\[
\text{SETH-hard} \quad \text{SETH-hard} \\
\text{SETH-hard} \quad \text{SETH-hard} \\
\text{SETH-hard} \quad \text{SETH-hard} \\
\]

[Backurs,Indyk’16] [B.,Larsen,Grønlund’17]
Word Break Problem

**Word Break** = +| ∘-Membership Testing

given string $S$ and dictionary $D$
can $S$ be split into words in $D$?

$n = |S|$

$m = \sum_{d \in D} |d|$

$O(nm)$ trivial algorithm

$\tilde{O}(n\sqrt{m} + m)$ improved algorithm

$\tilde{O}(nm^{1/2-1/18} + m)$ [Backurs, Indyk ’16]

$\tilde{O}(nm^{1/3} + m)$ [B., Larsen, Grønlund ’17]

matching conditional lower bound from $k$-Clique for combinatorial algorithms

Word Break is the only **intermediate** $t$-Membership problem!
I. Easy Example

II. Longest Common Subsequence

III. Hardness of Approximation

IV. Further Topics

V. Summary
Conclusion

**we have seen:**

- an involved lower bound
- coordinate gadgets, vector gadgets, normalized vector gadgets
- communication complexity (non-deterministic, randomized, one-sided)
- hardness of approximation in $\mathbb{P}$

**Open:** $(2 - \varepsilon)$-approximation of $\text{LCS}$ on binary strings in time $O(n^{2-\varepsilon})$?

Hardness of approximation for non-product-like problems.