Average-Case Fine-Grained Hardness

Marshall Ball  Alon Rosen  Manuel Sabin  Prashant Nalini Vasudevan
Average-Case Fine-Grained Hardness
Average-Case Fine-Grained Hardness
Average-Case Fine-Grained Hardness

- 3SUM
Average-Case Fine-Grained Hardness

- 3SUM
- APSP
Average-Case Fine-Grained Hardness

- 3SUM
- APSP
- Orthogonal Vectors
Average-Case Fine-Grained Hardness
Average-Case Fine-Grained Hardness
Average-Case Fine-Grained Hardness

- Natural object of study
Average-Case Fine-Grained Hardness

- Natural object of study
- Necessary for cryptography
Average-Case Fine-Grained Hardness

- Natural object of study
- Necessary for cryptography
- Potential use in algorithm design
Plan

- Introduce problems
- Present average-case reduction
- Summarise
- Present Proof of Work
- ???
- Profit.
Worst-Case: Orthogonal Vectors

\[ u \in U, \ v \in V: \ \text{disjoint?} \]

Best known worst-case algorithm \([AWY15]\):

\[ O(n^2 - 1/\Omega(\log(d/\log n))) \]

OV Conjecture (implied by SETH \([Wil05]\))

If \( d = \omega(\log n) \), OV takes \( n^2 - o(1) \) time.
Worst-Case: **Orthogonal Vectors**

∃ \( u \in U, \ v \in V \) disjoint?

Best known worst-case algorithm \( [AWY15] \):

\[
O\left( n^2 - \frac{1}{O(\log (d/\log n))} \right)
\]

OV Conjecture (implied by SETH \( [Wil05] \))

If \( d = \omega(\log n) \), OV takes \( n^2 - o(1) \) time.
Worst-Case: **Orthogonal Vectors**

∃ \( u \in U, v \in V \) such that disjoint?

Best known worst-case algorithm: \( O(n^2 - \frac{1}{O(\log(d/\log(n)))}) \)

**OV Conjecture (implied by SETH [Wil05])**

If \( d = \omega(\log(n)) \), OV takes \( n^2 - o(1) \) time.
Worst-Case: Orthogonal Vectors

\[ \exists u \in U, \ v \in V : \text{disjoint?} \]

Best known worst-case algorithm [AWY15]:

\[ O \left( n^2 - \frac{1}{O \left( \log \left( \frac{d}{\log n} \right) \right)} \right) \]

OV Conjecture (implied by SETH [Wil05]):

If \( d = \omega \left( \log n \right) \), OV takes \( n^2 - o \left( 1 \right) \) time.
Worst-Case: Orthogonal Vectors

Best known worst-case algorithm [AWY15]: $O(n^{2-1/O(\log(d/\log n))})$
Worst-Case: Orthogonal Vectors

Best known worst-case algorithm [AWY15]: $O(n^{2-1/O(\log(d/\log n))})$

OV Conjecture (implied by SETH [Wil05])

If $d = \omega(\log n)$, OV takes $n^{2-o(1)}$ time.
Worst-Case: Orthogonal Vectors

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

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

\[\exists u \in U, v \in V : \text{disjoint?}\]

Best known worst-case algorithm [AWY15]: \(O(n^{2-1/O(\log(d/\log n))})\)

OV Conjecture (implied by SETH [Wil05])

If \(d = \omega(\log n)\), OV takes \(n^{2-o(1)}\) time.
Average-Case: A Polynomial for OV (independently featured in [Wil16])

\[
\begin{pmatrix}
  \begin{array}{c}
  i \\
  \end{array}
  & \begin{array}{c}
  u_{i1} \ u_{i2} \ldots \ u_{id}
  \end{array}
  \\
  \hline
  U
  \\
  \end{pmatrix}

  \begin{pmatrix}
  \begin{array}{c}
  j \\
  \end{array}
  & \begin{array}{c}
  v_{j1} \ v_{j2} \ldots \ v_{jd}
  \end{array}
  \\
  \hline
  V
  \\
  \end{pmatrix}
\]

\[
\sum_{i \in [n]} \sum_{j \in [n]}\!
\left(1 - u_{i1} \ v_{j1}\right) \left(1 - u_{i2} \ v_{j2}\right) \ldots \left(1 - u_{id} \ v_{jd}\right) \equiv u_{id}, \ v_{jd} \text{ disjoint}
\]
Average-Case: **A Polynomial for OV** (independently featured in [Wil16])

\[
\begin{pmatrix}
  f \\
  \begin{array}{c}
    i \\
    j
  \end{array}
  \begin{array}{c}
    u_{i1} u_{i2} \ldots u_{id} \\
    v_{j1} v_{j2} \ldots v_{jd}
  \end{array}
\end{pmatrix}
\]

\[
U \\
V
\]

\[
(1 - u_{i1}v_{j1})(1 - u_{i2}v_{j2}) \cdots (1 - u_{id}v_{jd})
\]
Average-Case: A Polynomial for OV (independently featured in [Wil16])

\[
\begin{pmatrix}
  u_{i1} & u_{i2} & \cdots & u_{id} \\
  v_{j1} & v_{j2} & \cdots & v_{jd}
\end{pmatrix}
\]

\[f(U, V)\]

1 \iff u_i, v_j disjoint

\[(1 - u_{i1}v_{j1})(1 - u_{i2}v_{j2})\cdots(1 - u_{id}v_{jd})\]
Average-Case: A Polynomial for OV (independently featured in [Wil16])

\[
\begin{pmatrix}
  u_{i1} & u_{i2} & \ldots & u_{id} \\
  \quad & \quad & \quad & \quad \\
  \quad & \quad & \quad & \quad \\
  v_{j1} & v_{j2} & \ldots & v_{jd} \\
\end{pmatrix}
\]

\[
f(U \cup V) = \sum_{i \in [n]} \sum_{j \in [n]} (1 - u_{i1}v_{j1})(1 - u_{i2}v_{j2}) \cdots (1 - u_{id}v_{jd})
\]
Average-Case: A Polynomial for OV (independently featured in [Wil16])

\[
\begin{pmatrix}
  \begin{array}{c}
  i \\
  j
  \end{array}
  \begin{array}{c}
  u_{i1} u_{i2} \ldots u_{id} \\
  v_{j1} v_{j2} \ldots v_{jd}
  \end{array}
  \\
  U \\
  V
\end{pmatrix}
\]

\[
f(U, V) = \sum_{i \in [n]} \sum_{j \in [n]} (1 - u_{i1}v_{j1})(1 - u_{i2}v_{j2}) \cdots (1 - u_{id}v_{jd})
\]

\[
p > n^2
\]

\[
f : \mathbb{F}_p^{2nd} \to \mathbb{F}_p
\]
Average-Case: A Polynomial for OV (independently featured in [Wil16])

\[
\begin{pmatrix}
  i \\
  j
\end{pmatrix}
\begin{pmatrix}
  u_{i1} & u_{i2} & \ldots & u_{id} \\
  v_{j1} & v_{j2} & \ldots & v_{jd}
\end{pmatrix}
\]

\[
f(U, V) = \sum_{i \in [n]} \sum_{j \in [n]} (1 - u_{i1}v_{j1})(1 - u_{i2}v_{j2}) \cdots (1 - u_{id}v_{jd})
\]

\[
p > n^2
\]

\[
f : \mathbb{F}_p^{2nd} \rightarrow \mathbb{F}_p
\]

\[
deg(f) = 2d
\]

\[
d = \log^2 n
\]
Worst-Case to Average-Case

Theorem

\[ \exists A \text{ in time } n^{1+\alpha} : \Pr_{x \leftarrow \mathbb{F}_p^{2nd}} [A(x) = f(x)] \geq \frac{1}{n^{o(1)}} \]

\[ \Downarrow \]

\[ \exists B \text{ in time } n^{1+\alpha+o(1)} \text{ that decides OV} \]
Worst-Case to Average-Case

**Theorem**

\[ \exists A \text{ in time } n^{1+\alpha} : \Pr_{x \leftarrow \mathbb{P}^{2nd}} [A(x) = f(x)] \geq \frac{1}{n^{o(1)}} \]

\[ \Downarrow \]

\[ \exists B \text{ in time } n^{1+\alpha+o(1)} \text{ that decides } OV \]

**Corollary**

OV takes \( n^{2-o(1)} \) \( \Rightarrow \) \( f \) takes \( n^{2-o(1)} \) on average
Worst-Case to Average-Case (using ideas from [Lip91, GS92, CPS99])

\[ f : \mathbb{F}_p^{2nd} \rightarrow \mathbb{F}_p, \ deg(f) = 2d \]
Worst-Case to Average-Case (using ideas from [Lip91, GS92, CPS99])

\[ f : \mathbb{F}_p^{2nd} \rightarrow \mathbb{F}_p, \ deg(f) = 2d \]

\[ \Pr_{x \leftarrow \mathbb{F}_p^{2nd}} [A(x) = f(x)] \geq 0.9 \]

Time: \( t = n^{1+\alpha} \)
Worst-Case to Average-Case (using ideas from [Lip91, GS92, CPS99])

\[ f : \mathbb{F}_p^{2nd} \rightarrow \mathbb{F}_p, \quad \text{deg}(f) = 2d \]

\[ \Pr_{x \sim \mathbb{F}_p^{2nd}} [A(x) = f(x)] \geq 0.9 \quad \forall x : \Pr_B [B(x) = f(x)] \geq \frac{2}{3} \]

Time: \( t = n^{1+\alpha} \)
Worst-Case to Average-Case (using ideas from [Lip91, GS92, CPS99])

\[ f : \mathbb{F}_p^{2nd} \rightarrow \mathbb{F}_p, \ deg(f) = 2d \]

\[ \Pr_{x \leftarrow \mathbb{F}_p^{2nd}} [A(x) = f(x)] \geq 0.9 \]

Time: \( t = n^{1+\alpha} \)

\[ \forall x : \Pr_{B} [B(x) = f(x)] \geq \frac{2}{3} \]

Time: \( \tilde{O}(d \cdot nd + d \cdot t + d^3) \)
Worst-Case to Average-Case (using ideas from [Lip91, GS92, CPS99])

\[ f : \mathbb{F}_{p}^{2nd} \rightarrow \mathbb{F}_{p}, \ deg(f) = 2d \]

\[ \Pr_{x \leftarrow \mathbb{F}_{p}^{2nd}} [A(x) = f(x)] \geq 0.9 \]

\[ \forall x : \Pr_{B} [B(x) = f(x)] \geq \frac{2}{3} \]

\[ \text{Time: } t = n^{1+\alpha} \]

\[ g(t) = f(x + yt) \]

\[ g(0) = f(x), \ deg(g) \leq 2d \]
Worst-Case to Average-Case (using ideas from [Lip91, GS92, CPS99])

\[ f : \mathbb{F}_p^{2nd} \rightarrow \mathbb{F}_p, \deg(f) = 2d \]

\[ \Pr_{x \leftarrow \mathbb{F}_p^{2nd}}[A(x) = f(x)] \geq 0.9 \quad \forall x : \Pr_B[B(x) = f(x)] \geq \frac{2}{3} \]

Time: \( t = n^{1+\alpha} \)

\[ g(t) = f(x + yt) \]

\[ g(0) = f(x), \deg(g) \leq 2d \]
Worst-Case to Average-Case (using ideas from [Lip91, GS92, CPS99])

\[ f : \mathbb{F}_p^{2nd} \to \mathbb{F}_p, \text{deg}(f) = 2d \]

\[ \Pr_{x \leftarrow \mathbb{F}_p^{2nd}} [A(x) = f(x)] \geq 0.9 \quad \forall x : \Pr_B [B(x) = f(x)] \geq \frac{2}{3} \]

Time: \( t = n^{1+\alpha} \)

Error-correct from (noisy) \( g(1) \), \( g(2) \), \ldots, \( g(cd) \)
Worst-Case to Average-Case (using ideas from [Lip91, GS92, CPS99])

\[ f : \mathbb{F}_p^{2nd} \rightarrow \mathbb{F}_p, \ deg(f) = 2d \]

\[ \Pr_{x \leftarrow \mathbb{F}_p^{2nd}} [A(x) = f(x)] \geq 0.9 \quad \forall x : \Pr_B [B(x) = f(x)] \geq \frac{2}{3} \]

Time: \( t = n^{1+\alpha} \)

\[ g(t) = f(x + yt) \]

\[ g(0) = f(x), \ deg(g) \leq 2d \]

Error-correct from (noisy) \( g(1), g(2), \ldots, g(cd) \)

\[ \Pr_y [\text{too many } t's : A(x + yt) \neq g(t)] < \frac{1}{3} \]

(Markov Bound)
Worst-Case to Average-Case (using ideas from [Lip91, GS92, CPS99])

\[ f : \mathbb{F}_p^{2nd} \rightarrow \mathbb{F}_p, \ deg(f) = 2d \]

\[ \Pr_{\mathbf{x} \leftarrow \mathbb{F}_p^{2nd}} [A(\mathbf{x}) = f(\mathbf{x})] \geq 0.9 \]

Time: \( t = n^{1+\alpha} \)

\[ \forall \mathbf{x} : \Pr_{\mathbf{B}} [B(\mathbf{x}) = f(\mathbf{x})] \geq \frac{2}{3} \]

Time: \( \tilde{O}(d \cdot nd + d \cdot t + d^3) \)

\[ g(t) = f(\mathbf{x} + \mathbf{y}t) \]

\[ g(0) = f(\mathbf{x}), \ deg(g) \leq 2d \]

Error-correct from (noisy) \( g(1), g(2), \ldots, g(cd) \)

\[ \Pr_{\mathbf{y}} [\text{too many } t's : A(\mathbf{x} + \mathbf{y}t) \neq g(t)] < \frac{1}{3} \]

(Markov Bound)
Worst-Case to Average-Case (using ideas from [Lip91, GS92, CPS99])

\[ f : \mathbb{F}_p^{2nd} \to \mathbb{F}_p, \ deg(f) = 2d \]

\[
\Pr_{x \leftarrow \mathbb{F}_p^{2nd}} [A(x) = f(x)] \geq 0.9
\]

\[
\forall x : \Pr_B [B(x) = f(x)] \geq \frac{2}{3}
\]

Time: \( t = n^{1+\alpha} \)

Time: \( \tilde{O}(d \cdot nd + d \cdot t + d^3) \)

\[
f(U, V) = \sum_{i \in [n]} \sum_{j \in [n]} \prod_{\ell \in [d]} (1 - u_{i\ell} v_{j\ell})
\]
Worst-Case to Average-Case (using ideas from [Lip91, GS92, CPS99])

\[ f : \mathbb{F}_p^{2nd} \to \mathbb{F}_p, \text{deg}(f) = 2d \]

\[ \Pr_{\mathbf{x} \leftarrow \mathbb{F}_p^{2nd}} [A(\mathbf{x}) = f(\mathbf{x})] \geq 0.9 \]

\[ \forall \mathbf{x} : \Pr_B [B(\mathbf{x}) = f(\mathbf{x})] \geq \frac{2}{3} \]

Time: \( t = n^{1+\alpha} \)

\[ \text{Time: } \tilde{O}(d \cdot nd + d \cdot t + d^3) \]

\[ f(U, V) = \sum_{i \in [n]} \sum_{j \in [n]} \prod_{\ell \in [d]} (1 - u_{i\ell} v_{j\ell}) \]

\[ = \left( \sum_{i \in [n/2]} \sum_{j \in [n/2]} + \sum_{i \in [n/2]} \sum_{j \in (n/2, n]} + \sum_{i \in (n/2, n]} \sum_{j \in [n/2]} + \sum_{i \in (n/2, n]} \sum_{j \in (n/2, n]} \right) \prod_{\ell \in [d]} (1 - u_{i\ell} v_{j\ell}) \]
Worst-Case to Average-Case (using ideas from [Lip91, GS92, CPS99])

\[ f : \mathbb{F}_p^{2nd} \rightarrow \mathbb{F}_p, \text{deg}(f) = 2d \]

\[
\Pr_{x \leftarrow \mathbb{F}_p^{2nd}} [A(x) = f(x)] \geq \frac{1}{n^{p(1)}}
\]

\[ \forall x : \Pr_B [B(x) = f(x)] \geq \frac{2}{3} \]

Time: \( t = n^{1+\alpha} \)

\[ \text{Time: } \tilde{O}(d \cdot nd + d \cdot t + d^3) \]

\[
f(U, V) = \sum_{i \in [n]} \sum_{j \in [n]} \prod_{\ell \in [d]} (1 - u_{i\ell} v_{j\ell})
\]

\[
= \left( \sum_{i \in [n/2]} + \sum_{i \in [n/2]} + \sum_{i \in (n/2, n]} + \sum_{i \in (n/2, n]} \right) \prod_{\ell \in [d]} (1 - u_{i\ell} v_{j\ell})
\]
Worst-Case to Average-Case (using ideas from [Lip91, GS92, CPS99])

\[ f : \mathbb{F}_p^{2nd} \to \mathbb{F}_p, \ deg(f) = 2d \]

\[ \Pr_{x \leftarrow \mathbb{F}_p^{2nd}} [A(x) = f(x)] \geq \frac{1}{p^{\Theta(1)}} \]

\[ \forall x : \Pr_B [B(x) = f(x)] \geq \frac{2}{3} \]

Time: \( t = n^{1+\alpha} \)

Time: \( t^{1+o(1)} \)

\[ f(U, V) = \sum_{i \in [n]} \sum_{j \in [n]} \prod_{\ell \in [d]} (1 - u_i v_{j\ell}) \]

\[ = \left( \sum_{i \in [n/2]} \sum_{j \in [n/2]} + \sum_{i \in [n/2]} \sum_{j \in (n/2, n]} + \sum_{i \in (n/2, n]} \sum_{j \in [n/2]} + \sum_{i \in (n/2, n]} \sum_{j \in (n/2, n]} \right) \prod_{\ell \in [d]} (1 - u_i v_{j\ell}) \]
Intermediate Summary

We have a worst-to-average case reduction from OV (resp. 3SUM, APSP) to evaluating a polynomial $f$ (other respective polynomials).
Intermediate Summary

We have a worst-to-average case reduction from OV (resp. 3SUM, APSP) to evaluating a polynomial $f$ (other respective polynomials). In addition,

- $f$ has low degree – $\text{polylog}(n)$.
- $f$ is somewhat efficiently computable – $\tilde{O}(n^2)$.
- $f$ is downward self-reducible.

Theorem \cite{Wil16} 
There is an MA proof system for proving $(f(x) = y)$ that has:

- perfect completeness and negligible soundness.
- prover complexity $\tilde{O}(n^2)$.
- verifier complexity $\tilde{O}(n)$.
Intermediate Summary

We have a worst-to-average case reduction from OV (resp. 3SUM, APSP) to evaluating a polynomial $f$ (other respective polynomials). In addition,

- $f$ has low degree – polylog($n$).
- $f$ is somewhat efficiently computable – $\tilde{O}(n^2)$.
- $f$ is downward self-reducible.

**Theorem [Wil16]**

There is an MA proof system for proving $(f(x) = y)$ that has:

- perfect completeness and negligible soundness.
- prover complexity $\tilde{O}(n^2)$.
- verifier complexity $\tilde{O}(n)$. 
Proof of Work

\[ \text{Prover} \quad \vdash \quad \text{Verifier} \]

- \( x \leftarrow F \)
- Find \( p \)
- Compute \( f(x) = z \) and MA proof \( \pi \)
- Verify using \( \pi \) that \( f(x) = z \)

\[ \tilde{O}(n) \quad \tilde{O}(n^2) \]

\[ \Pr \left[ \text{Prover can run in } n^2 - \epsilon \text{ and convince Verifier} \right] \leq \frac{1}{n^{\epsilon/2}} \]

(See \[DN92\] for generic constructions and applications.)
Proof of Work

$\mathbf{x} \leftarrow \mathbb{F}_p^{2nd}$

Compute $f(x) = z$ and MA proof $\pi(z, \pi)$

Verify using $\pi$ that $f(x) = \tilde{O}(n)$ and $\tilde{O}(n^2)$

$\Pr[\text{Prover can run in } n^2 - \epsilon \text{ and convince Verifier}] \leq 1/n^{\epsilon/2}$

(See [DN92] for generic constructions and applications.)
Proof of Work

Prover

Verifier

\[ x \leftarrow \mathbb{F}_{p}^{2nd} \]

Compute \( f(x) = z \)
and MA proof \( \pi \)

\[ z, \pi \]
Proof of Work

Compute $f(x) = z$ and MA proof $\pi$

Verify using $\pi$ that $f(x) = z$
Proof of Work

<table>
<thead>
<tr>
<th>Prover</th>
<th>Verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>x</strong> ← ( \mathbb{F}_{p}^{2nd} )</td>
<td></td>
</tr>
<tr>
<td>Compute ( f(x) = z ) and MA proof ( \pi )</td>
<td></td>
</tr>
<tr>
<td>( z, \pi )</td>
<td>Verify using ( \pi ) that ( f(x) = z )</td>
</tr>
<tr>
<td>( \tilde{O}(n^2) )</td>
<td>( \tilde{O}(n) )</td>
</tr>
</tbody>
</table>

Pr \( [\text{Prover can run in } n^2 - \epsilon \text{ and convince Verifier} ] \leq \frac{1}{n \epsilon / 2} \) (See [DN92] for generic constructions and applications.)
Proof of Work

\[
\begin{align*}
\text{Prover} & \quad \text{Verifier} \\
\text{\texttt{x} \leftarrow \mathbb{F}_p^{2nd}} & \\
\text{Compute } f(\texttt{x}) = z \quad \text{Verify using } \pi \text{ that } f(\texttt{x}) = z \\
\text{and MA proof } \pi & \\
\tilde{O}(n^2) & \quad \tilde{O}(n) \\
\Pr [\text{Prover can run in } n^{2-\epsilon} \text{ and convince Verifier}] & \leq \frac{1}{n^{\epsilon/2}}
\end{align*}
\]
Proof of Work

\begin{align*}
\text{Prover} & \quad \text{Verifier} \\
\text{Compute } f(x) = z & \quad \text{Verify using } \pi \text{ that } f(x) = z \\
\tilde{O}(n^2) & \quad \tilde{O}(n) \\
\Pr[\text{Prover can run in } n^{2-\epsilon} \text{ and convince Verifier}] & \leq \frac{1}{n^{\epsilon/2}}
\end{align*}

(See [DN92] for generic constructions and applications.)
What Next?

▶ Average-case complexity of OV, 3SUM, etc.
▶ Fine-grained cryptography
▶ Some prior work under other assumptions [Mer78, Hås87, BGI08, DVV16, . . . ]
▶ Fine-grained OWFs from SETH?
▶ Beat Merkle’s key agreement under these assumptions?
▶ Average-case algorithms
▶ Design algorithms to evaluate polynomials that work on average.
▶ Better reductions
▶ Is it actually possible to do better than guessing at random?
What Next?

- Average-case complexity of OV, 3SUM, etc.
What Next?

- Average-case complexity of OV, 3SUM, etc.
- Fine-grained cryptography
  - Some prior work under other assumptions [Mer78, Hås87, BGI08, DVV16, …].
  - Fine-grained OWFs from SETH?
  - Beat Merkle’s key agreement under these assumptions?
What Next?

- Average-case complexity of OV, 3SUM, etc.
- Fine-grained cryptography
  - Some prior work under other assumptions [Mer78, Hås87, BGI08, DVV16, ...].
  - Fine-grained OWFs from SETH?
  - Beat Merkle's key agreement under these assumptions?
- Average-case algorithms
  - Design algorithms to evaluate polynomials that work on average.
What Next?

- Average-case complexity of OV, 3SUM, etc.
- Fine-grained cryptography
  - Some prior work under other assumptions [Mer78, Hås87, BGI08, DVV16, …].
  - Fine-grained OWFs from SETH?
  - Beat Merkle’s key agreement under these assumptions?
- Average-case algorithms
  - Design algorithms to evaluate polynomials that work on average.
- Better reductions
  - Is it actually possible to do better than guessing at random?
To be passed in case of an abundance of time.
$k$-SAT and SETH

\[
\begin{aligned}
(k, 
(x_1 \lor \overline{x_2} \lor \ldots) \land (\ldots \lor x_n \lor \ldots) \land \ldots \land (\ldots \lor \ldots \lor \ldots)
\end{aligned}
\]

Best known worst-case algorithm [PPSZ05]:

\[
\tilde{O}(2^{(1 - c/k)n})
\]

Strong Exponential Time Hypothesis (SETH) [IPZ98]:

\[
\forall \epsilon \exists k: k\text{-SAT takes } \tilde{\Omega}(2^{(1 - \epsilon)n}) \text{ time.}
\]
$k$-SAT and SETH

\[
(k \text{ SAT}) = (x_1 \lor \overline{x_2} \lor \ldots) \land (\ldots \lor x_n \lor \ldots) \land \ldots \land (\ldots \lor \ldots \lor \ldots)
\]

Best known worst-case algorithm [PPSZ05]: $\tilde{O}(2^{(1-c/k)n})$

Strong Exponential Time Hypothesis (SETH) [IPZ98]

$\forall \epsilon \exists k: k$-SAT takes $\tilde{O}(2^{(1-\epsilon/n)}n)$ time.
**$k$-SAT and SETH**

\[
\begin{align*}
  &k \\
\left( x_1 \lor \overline{x}_2 \lor \ldots \right) \land \left( \ldots \lor x_n \lor \ldots \right) \land \ldots \land \left( \ldots \lor \ldots \lor \ldots \right)
\end{align*}
\]

Best known worst-case algorithm [PPSZ05]: \( \tilde{O}(2^{(1-c/k)n}) \)

**Strong Exponential Time Hypothesis (SETH) [IPZ98]**

\[\forall \epsilon \exists k: k\text{-SAT takes } \tilde{\Omega}(2^{(1-\epsilon)n}) \text{ time.}\]
An Efficient MA Protocol for $f$ [Wil16]

$(U, V) \in \mathbb{F}_p^{2nd}, z \in \mathbb{F}_p$
An Efficient MA Protocol for $f$ [Wil16]

$(U, V) \in \mathbb{F}_p^{2nd}, z \in \mathbb{F}_p$

$\phi_1, \ldots, \phi_d : \mathbb{F}_p \to \mathbb{F}_p$

$\forall i \in [n] : \phi_\ell(i) = u_{i\ell}$

$\deg(\phi_\ell) \leq n - 1$
An Efficient MA Protocol for $f$ [Wil16]

$$(U, V) \in \mathbb{F}_p^{2nd}, z \in \mathbb{F}_p$$

$\phi_1, \ldots, \phi_d : \mathbb{F}_p \to \mathbb{F}_p$

$\forall i \in [n] : \phi_\ell(i) = u_{i\ell}$

$deg(\phi_\ell) \leq n - 1$

$$f(U, V) = \sum_{i \in [n]} \sum_{j \in [n]} \prod_{\ell \in [d]} (1 - u_{i\ell}v_{j\ell}) = \sum_{i \in [n]} \left[ \sum_{j \in [n]} \prod_{\ell \in [d]} (1 - \phi_\ell(i)v_{j\ell}) \right] = \sum_{i \in [n]} r(i)$$
An Efficient MA Protocol for $f$ [Wil16]

$(U, V) \in \mathbb{F}^{2nd}_p$, $z \in \mathbb{F}_p$

$\phi_1, \ldots, \phi_d : \mathbb{F}_p \rightarrow \mathbb{F}_p$

$\forall i \in [n] : \phi_\ell(i) = u_{i\ell}$

$\deg(\phi_\ell) \leq n - 1$

$f(U, V) = \sum \sum \prod (1 - u_{i\ell}v_{j\ell}) = \sum \left[ \sum \prod (1 - \phi_\ell(i)v_{j\ell}) \right] = \sum r(i)$

- Proof: Coefficients of $r$. (Interpolation – $\tilde{O}(n^2)$)
An Efficient MA Protocol for $f$ \cite{Wil16}

$$(U, V) \in \mathbb{F}_p^{2nd}, z \in \mathbb{F}_p$$

$\phi_1, \ldots, \phi_d : \mathbb{F}_p \rightarrow \mathbb{F}_p$

$\forall i \in [n] : \phi_\ell(i) = u_{i,\ell}$

$\deg(\phi_\ell) \leq n - 1$

$$f(U, V) = \sum_{i \in [n]} \sum_{j \in [n]} \prod_{\ell \in [d]} (1 - u_{i,\ell}v_{j,\ell}) = \sum_{i \in [n]} \left[ \sum_{j \in [n]} \prod_{\ell \in [d]} (1 - \phi_\ell(i)v_{j,\ell}) \right] = \sum_{i \in [n]} r(i)$$

- Proof: Coefficients of $r$. (Interpolation – $\tilde{O}(n^2)$)
- Verification:
  - Check $r$ at random point. (Computation of $\phi$ and correct value – $\tilde{O}(n)$)
  - Compute $r(i)$ for $i \in [n]$ and sum to get $f(U, V)$. (Batch evaluation – $\tilde{O}(n)$)
Amir Abboud, Richard Ryan Williams, and Huacheng Yu.
More applications of the polynomial method to algorithm design.

Eli Biham, Yaron J. Goren, and Yuval Ishai.
Basing weak public-key cryptography on strong one-way functions.

Jin-yi Cai, Aduri Pavan, and D. Sivakumar.
On the hardness of permanent.

Cynthia Dwork and Moni Naor.
Pricing via processing or combatting junk mail.


Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials.

Johan Håstad.
One-way permutations in $NC^0$.

Russell Impagliazzo, Ramamohan Paturi, and Francis Zane.
Which problems have strongly exponential complexity?

Richard Lipton.
New directions in testing.

Ralph C. Merkle.
Secure communications over insecure channels.

Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane.
An improved exponential-time algorithm for k-sat.

Ryan Williams.
A new algorithm for optimal 2-constraint satisfaction and its implications.
Strong ETH breaks with merlin and arthur: Short non-interactive proofs of batch evaluation.

In 31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan, pages 2:1–2:17, 2016.