A Parameterized Framework for Hardness of Approximation

Karthik C. S.
(Weizmann Institute of Science)

Joint work with

Bundit Laekhanukit
(Shanghai University of Finance and Economics)

Pasin Manurangsi
(UC Berkeley)
Dominating Set Problem

$G(V, E)$

$S \subseteq V$ is a Dominating Set of $G$ if $\forall u \in V$: $u \in S$, or $\exists v \in S$ such that $(u, v) \in E$.

Computational Problem: Given $G$ and $k \in \mathbb{N}$, determine if $\exists S \subseteq V$: $S$ is a Dominating Set of $G$ $\Rightarrow |S| \leq k$.

$\rightarrow$ NP-Complete [Karp'/seven.taboldstyle/two.taboldstyle]

$\rightarrow$ $\ln |V|$-approximation is in P [Slavík'/nine.taboldstyle/six.taboldstyle]

$\rightarrow$ $(1 - \varepsilon) \ln |V|$-approximation is NP-Complete [DS'/one.taboldstyle/four.taboldstyle]
Dominating Set Problem

$S \subseteq V$ is a Dominating Set of $G$ if \( \forall u \in V: \)
- \( u \in S \), or
- \( \exists v \in S \) such that \( (u, v) \in E \)

$G(V, E)$
Dominating Set Problem

\[ S \subseteq V \] is a Dominating Set of \( G \) if
\[ \forall u \in V: \]
\[ \begin{align*}
&\circ \ u \in S, \text{ or} \\
&\circ \ \exists v \in S \text{ such that } (u, v) \in E
\end{align*} \]
Dominating Set Problem

$S \subseteq V$ is a Dominating Set of $G$ if \( \forall u \in V: \)
- \( u \in S, \) or
- \( \exists v \in S \) such that \((u, v) \in E\)

**Computational Problem:** Given $G$ and $k \in \mathbb{N}$, determine if \( \exists S \subseteq V: \)
- \( S \) is a Dominating Set of $G$
- \( |S| \leq k \)
Dominating Set Problem

$S \subseteq V$ is a **Dominating Set** of $G$ if $\forall u \in V$:
- $u \in S$, or
- $\exists v \in S$ such that $(u, v) \in E$

**Computational Problem**: Given $G$ and $k \in \mathbb{N}$, determine if $\exists S \subseteq V$:
- $S$ is a Dominating Set of $G$
- $|S| \leq k$

$\longrightarrow$ **NP-Complete** [Karp’72]
Dominating Set Problem

$S \subseteq V$ is a **Dominating Set** of $G$ if

$\forall u \in V$:

- $u \in S$, or
- $\exists v \in S$ such that $(u, v) \in E$

**Computational Problem**: Given $G$ and $k \in \mathbb{N}$, determine if $\exists S \subseteq V$:

- $S$ is a Dominating Set of $G$
- $|S| \leq k$

$\ln |V|$ approximation is in $P$ [Slavík’96]

NP-Complete [Karp’72]
Dominating Set Problem

$S \subseteq V$ is a Dominating Set of $G$ if
\[ \forall u \in V:\]
\[ \begin{align*}
\circ & \quad u \in S, \text{ or } \\
\circ & \quad \exists v \in S \text{ such that } (u, v) \in E
\end{align*} \]

Computational Problem: Given $G$ and $k \in \mathbb{N}$, determine if $\exists S \subseteq V$:
\[ \begin{align*}
\circ & \quad S \text{ is a Dominating Set of } G \\
\circ & \quad |S| \leq k 
\end{align*} \]

\[ \rightarrow \text{ ln}|V| \text{ approximation is in P} \quad [\text{Slavík’96}] \]

\[ \rightarrow (1 - \varepsilon) \text{ ln}|V| \text{ approximation is NP-Complete} \quad [\text{DS’14}] \]

\[ \rightarrow \text{ NP-Complete} \quad [\text{Karp’72}] \]
Parameterized Dominating Set Problem

Computational Problem: Given \( G \) and parameter \( k \in \mathbb{N} \), determine if \( \exists S \subseteq V: \)

- \( S \) is a Dominating Set of \( G \)
- \( |S| \leq k \)
Parameterized Dominating Set Problem

Computational Problem: Given $G$ and parameter $k \in \mathbb{N}$, determine if $\exists S \subseteq V$:

- $S$ is a Dominating Set of $G$
- $|S| \leq k$

Fixed Parameter Tractability (FPT): The problem can be decided in $F(k) \cdot \text{poly}(|V|)$ time, for some computable function $F$. 

Parameterized Dominating Set Problem

Computational Problem: Given $G$ and parameter $k \in \mathbb{N}$, determine if $\exists S \subseteq V$:

- $S$ is a Dominating Set of $G$
- $|S| \leq k$

Fixed Parameter Tractability (FPT): The problem can be decided in $F(k) \cdot \text{poly}(|V|)$ time, for some computable function $F$. 

- $k$-Dominating Set
- $k$-Clique
- $k$-Vertex Cover
Parameterized Dominating Set Problem

Computational Problem: Given $G$ and parameter $k \in \mathbb{N}$, determine if $\exists S \subseteq V$:

- $S$ is a Dominating Set of $G$
- $|S| \leq k$

Fixed Parameter Tractability (FPT): The problem can be decided in $F(k) \cdot \text{poly}(|V|)$ time, for some computable function $F$.

\[ \begin{align*}
\text{W[2]} & \quad \text{W[1]} \quad \text{FPT} \\
\text{k-Dominating Set} & \quad \text{k-Clique} \quad \text{k-Vertex Cover}
\end{align*} \]
Parameterized Complexity of Dominating Set Problem

Given graph on $N$ vertices and parameter $k$:

○ $W[2]$ complete [DF’95]
Parameterized Complexity of Dominating Set Problem

Given graph on $N$ vertices and parameter $k$:

- $W[2]$ complete [DF'95]
- **Trivial Algorithm**: $O(N^{k+1})$ time
Parameterized Complexity of Dominating Set Problem

Given graph on $N$ vertices and parameter $k$:

- \( W[2] \) complete [DF’95]
- **Trivial Algorithm**: \( O(N^{k+1}) \) time
- **State of the Art**: \( N^{k+o(1)} \) time [EG’04, PW’10]
Given graph on \( N \) vertices and parameter \( k \):

- **\( W[2] \)** complete [DF’95]
- **Trivial Algorithm**: \( O(N^{k+1}) \) time
- **State of the Art**: \( N^{k+o(1)} \) time [EG’04, PW’10]
- No \( N^{o(k)} \) time algorithm assuming ETH [CHKX’06]
Given graph on $N$ vertices and parameter $k$:

- $W[2]$ complete [DF’95]
- **Trivial Algorithm**: $O(N^{k+1})$ time
- **State of the Art**: $N^{k+o(1)}$ time [EG’04, PW’10]
- No $N^{o(k)}$ time algorithm assuming ETH [CHKX’06]

There exists $\delta > 0$ such that no algorithm can solve $3$-CNF-SAT in $O(2^{\delta n})$ time where $n$ is the number of variables.
Given graph on $N$ vertices and parameter $k$:

- $W[2]$ complete [DF’95]
- **Trivial Algorithm**: $O(N^{k+1})$ time
- **State of the Art**: $N^{k+o(1)}$ time [EG’04, PW’10]
- No $N^{o(k)}$ time algorithm assuming ETH [CHKX’06]
- No $O(N^{k-\epsilon})$ algorithm assuming SETH [PW’10]
Parameterized Complexity of Dominating Set Problem

Given graph on $N$ vertices and parameter $k$:

- $W[2]$ complete [DF’95]
- **Trivial Algorithm:** $O(N^{k+1})$ time
- **State of the Art:** $N^{k+o(1)}$ time [EG’04, PW’10]
- No $N^{o(k)}$ time algorithm assuming ETH [CHKX’06]
- No $O(N^{k-\varepsilon})$ algorithm assuming SETH [PW’10]

For every $\varepsilon > 0$, there exists $\ell(\varepsilon) \in \mathbb{N}$ such that no algorithm can solve $\ell$-SAT in $O(2^{(1-\varepsilon)n})$ time where $n$ is the number of variables.
FPT Approximability: The problem has a $T(k)$ approximation algorithm running in time $F(k) \cdot \text{poly}(N)$ time.
FPT Approximability: The problem has a $T(k)$ approximation algorithm running in time $F(k)\cdot\text{poly}(N)$ time.

**Approximate** Parameterized Dominating Set Problem: Given a graph $G$ and parameter $k$ distinguish between:

- $\exists$ a dominating set of size at most $k$
- There is no dominating set of size $T(k)\cdot k$
FPT Approximability: The problem has a $T(k)$ approximation algorithm running in time $F(k) \cdot \text{poly}(N)$ time.

Approximate Parameterized Dominating Set Problem: Given a graph $G$ and parameter $k$ distinguish between:

- $\exists$ a dominating set of size at most $k$
- $\exists$ there is no dominating set of size $T(k) \cdot k$

Major Open Problem: Is there some computable function $T$ for which the above problem is in FPT?
Previous Works

Two decades later:

- Any constant approximation is \(W[1]\)-hard.
- No \((\log k)^{1/4}\) approximation algorithm in \(N^{O(\sqrt{k})}\) time, assuming ETH.
- No \(T(k)\) approximation algorithm in \(N^{O(k)}\) time, assuming Gap-ETH.
- Can we show every \(T(k)\) approximation is \(W[1]\)-hard?
- Can we show no \(T(k)\) approximation algorithm exists running in time \(N^{O(k)}\), assuming ETH?
- Can we show no \(T(k)\) approximation algorithm exists running in time \(N^{k-\varepsilon}\), assuming SETH?

Yes!

There exists a constant \(\delta > 0\) such that any algorithm that, on input a 3-SAT formula \(\varphi\) on \(n\) variables and \(O(n)\) clauses, can distinguish between \(\text{SAT}(\varphi) = 1\) and \(\text{SAT}(\varphi) < 0.9\), must run in time at least \(2^\delta n\).
Two decades later:

○ Any **constant** approximation is $W[1]$-hard [CL’16]
Two decades later:

- Any constant approximation is $W[1]$-hard [CL’16]
- No $(\log k)^{1/4}$ approximation algorithm in $N^{o(\sqrt{k})}$ time, assuming ETH [CL’16]
Two decades later:

- Any constant approximation is \( W[1] \)-hard [CL’16]
- No \((\log k)^{1/4}\) approximation algorithm in \( N^{o(\sqrt{k})} \) time, assuming ETH [CL’16]
- No \( T(k) \) approximation algorithm in \( N^{o(k)} \) time, assuming Gap-ETH [CCKLMNT’17]
Two decades later:

- Any constant approximation is $W[1]$-hard [CL’16]
- No $(\log k)^{1/4}$ approximation algorithm in $N^{o(\sqrt{k})}$ time, assuming ETH [CL’16]
- No $T(k)$ approximation algorithm in $N^{o(k)}$ time, assuming Gap-ETH [CCKLMNT’17]

There exists a constant $\delta > 0$ such that any algorithm that, on input a 3-SAT formula $\varphi$ on $n$ variables and $O(n)$ clauses, can distinguish between $\text{SAT}(\varphi) = 1$ and $\text{SAT}(\varphi) < 0.9$, must run in time at least $2^{\delta n}$. 
Two decades later:

- Any constant approximation is $W[1]$-hard [CL’16]
- No $(\log k)^{1/4}$ approximation algorithm in $N^{o(\sqrt{k})}$ time, assuming ETH [CL’16]
- No $T(k)$ approximation algorithm in $N^{o(k)}$ time, assuming Gap-ETH [CCKLMNT’17]

★ Can we show every $T(k)$ approximation is $W[1]$-hard?
Two decades later:

- Any constant approximation is $W[1]$-hard [CL’16]
- No $(\log k)^{1/4}$ approximation algorithm in $N^{o(\sqrt{k})}$ time, assuming ETH [CL’16]
- No $T(k)$ approximation algorithm in $N^{o(k)}$ time, assuming Gap-ETH [CCKLMNT’17]

Can we show every $T(k)$ approximation is $W[1]$-hard?

Can we show no $T(k)$ approximation algorithm exists running in time $N^{o(k)}$, assuming ETH?
Two decades later:

- Any constant approximation is $W[1]$-hard [CL’16]
- No $(\log k)^{1/4}$ approximation algorithm in $N^{o(\sqrt{k})}$ time, assuming ETH [CL’16]
- No $T(k)$ approximation algorithm in $N^{o(k)}$ time, assuming Gap-ETH [CCKLMNT’17]

★ Can we show every $T(k)$ approximation is $W[1]$-hard?
★ Can we show no $T(k)$ approximation algorithm exists running in time $N^{o(k)}$, assuming ETH?
★ Can we show no $T(k)$ approximation algorithm exists running in time $N^{k-\epsilon}$, assuming SETH?
Previous Works

Two decades later:

- Any constant approximation is \text{W[1]}-hard \cite{CL16}
- No \((\log k)^{1/4}\) approximation can run in \(N^{o(\sqrt{k})}\) time, assuming ETH \cite{S07}
- No \(T(k)\) approximation algorithm in \(N^{o(k)}\) time, assuming Gap-ETH \cite{CLMNT17}

\[\star\] Can we show every \(T(k)\) approximation is \(\text{W[1]}\)-hard?
\[\star\] Can we show no \(T(k)\) approximation algorithm exists running in time \(N^{o(k)}\), assuming ETH?
\[\star\] Can we show no \(T(k)\) approximation algorithm exists running in time \(N^{k-\varepsilon}\), assuming SETH?
Previous Works

Two decades later:

- Any constant approximation is $W[1]$-hard [CL’16]
- No $(\log k)^{1/4}$ approximation algorithm in $N^{o(\sqrt{k})}$ time, assuming ETH [CL’16]
- No $T(k)$ approximation algorithm in $N^{o(k)}$ time, assuming Gap-ETH [CCKLMNT’17]

★ Can we show every $T(k)$ approximation is $W[1]$-hard?
★ Can we show no $T(k)$ approximation algorithm exists running in time $N^{o(k)}$, assuming ETH?
★ Can we show no $T(k)$ approximation algorithm exists running in time $N^{k-\varepsilon}$, assuming SETH?
Previous Works

Two decades later:

- Any **constant** approximation is \( W[1] \)-hard [CL’16]
- No \((\log k)^{1/4}\) approximation algorithm in \( N^{o(\sqrt{k})} \) time, assuming ETH [CL’16]
- No \( T(k) \) approximation algorithm in \( N^{o(k)} \) time, assuming Gap-ETH [CCKLMNT’17]

★ Can we show every \( T(k) \) approximation is \( W[1] \)-hard?
★ Can we show no \( T(k) \) approximation algorithm exists running in time \( N^{o(k)} \), assuming ETH?
★ Can we show no \( T(k) \) approximation algorithm exists running in time \( N^{k^{-\varepsilon}} \), assuming SETH?

Yes!
Our Results

- Any $T(k)$ approximation is $W[1]$-hard
- No $T(k)$ approximation algorithm in $N^{o(k)}$ time, assuming ETH
- No $T(k)$ approximation algorithm in $N^{k-\varepsilon}$ time, assuming SETH
- No $T(k)$ approximation algorithm in $N^{[k/2]-\varepsilon}$ time, assuming $k$-SUM Hypothesis

---

$k$-SUM Problem: Given $A_1, \ldots, A_k \subseteq [-N/2, N/2]$ where $N = \sum_{i \in [k]} |A_i|$, determine whether there exist $x_i \in A_i$, $\forall i \in [k]$ such that $\sum_{i \in [k]} x_i = 0$.

$k$-SUM Hypothesis: For every integer $k \geq 3$ and every $\varepsilon > 0$, no $O(N^{\lceil k/2 \rceil - \varepsilon})$ time algorithm can solve the $k$-SUM problem.
Our Results

- Any $T(k)$ approximation is $W[1]$-hard
- No $T(k)$ approximation algorithm in $N^{o(k)}$ time, assuming ETH
- No $T(k)$ approximation algorithm in $N^{k-\varepsilon}$ time, assuming SETH
- No $T(k)$ approximation algorithm in $N^{\lceil k/2 \rceil - \varepsilon}$ time, assuming $k$-SUM Hypothesis

$k$-SUM Problem: Given $A_1, \ldots, A_k \subseteq [-N^{2k}, N^{2k}]$ where $N = \sum_{i \in [k]} |A_i|$, determine whether there exist $x_i \in A_i, \forall i \in [k]$ such that $\sum_{i \in [k]} x_i = 0$. 
Our Results

- Any $T(k)$ approximation is $W[1]$-hard
- No $T(k)$ approximation algorithm in $N^{o(k)}$ time, assuming ETH
- No $T(k)$ approximation algorithm in $N^{k-\varepsilon}$ time, assuming SETH
- No $T(k)$ approximation algorithm in $N^{[k/2]-\varepsilon}$ time, assuming $k$-SUM Hypothesis

**$k$-SUM Problem**: Given $A_1, \ldots, A_k \subseteq [-N^{2k}, N^{2k}]$ where $N = \sum_{i \in [k]} |A_i|$, determine whether there exist $x_i \in A_i, \forall i \in [k]$ such that $\sum_{i \in [k]} x_i = 0$.

**$k$-SUM Hypothesis**: For every integer $k \geq 3$ and every $\varepsilon > 0$, no $O(N^{[k/2]-\varepsilon})$ time algorithm can solve the $k$-SUM problem.
Our Results

- Any $T(k)$ approximation is $W[1]$-hard
- No $T(k)$ approximation algorithm in $N^{o(k)}$ time, assuming ETH
- No $T(k)$ approximation algorithm in $N^{k-\varepsilon}$ time, assuming SETH
- No $T(k)$ approximation algorithm in $N^{[k/2]-\varepsilon}$ time, assuming $k$-SUM Hypothesis

All results obtained in an Unified Proof Framework!
The Framework

PSP

\[ \text{MaxCover} \]

- \( W[1] \neq \text{FPT} \)
- \( \text{ETH} \)
- \( \text{SETH} \)
- \( k\text{-Sum Hyp.} \)

Reduction from \([CCKLMNT17]\)

\( k\text{-DomSet} \)

\[ \text{PSP} \text{(MultiEQ)} \rightarrow \text{PSP} \text{(Disj)} \rightarrow \text{PSP} \text{(SUMZERO)} \]
The Framework

Gap-ETH \xrightarrow{\text{Reduction from [CCKLMNT17]}} \text{MaxCover} \xrightarrow{\text{Reduction from [CCKLMNT17]}} k\text{-DomSet}
The Framework

- **MaxCover**

**PSP**
- **PSP**($\text{MultEQ}$)
- **PSP**($\text{Disj}$)
- **PSP**($\text{SumZero}$)

**Preprocessing Step**

- **W[1] ≠ FPT**
- **ETH**
- **SETH**
- **k-Sum Hyp.**

**MaxCover**

Reduction from [CCKLMNT17]

**k-DomSet**

Gap Amplification

Gap Translation

Gap Problems in P

Gap-ETH
The Framework

MaxCover

\[ \text{Reduction from } \text{[CCKLMNT17]} \]

PSP

\[ k \text{-} \text{DomSet} \]

\( k \text{-} \text{Sum Hyp.} \)

\( \text{ETH} \)

\( \text{SETH} \)

PSP\( (\text{MultiEQ}) \)

PSP\( (\text{Disj}) \)

PSP\( (\text{SumZero}) \)

Gap Amplification
The Framework

MaxCover

Reduction from [CCKLMNT/one.taboldstyle/seven.taboldstyle]

PSP

W[1] ≠ FPT

ETH

SETH

k-Sum Hyp.

PSP(MultEQ)

PSP(Disj)

PSP(SumZero)

MaxCover

PSP

PSP([M/u.sc/l.sc/t.scE/q.sc])

PSP([S/u.sc/m.scZ/e.sc/r.sc/o.sc])

PSP([D/i.sc/s.sc/j.sc])

Gap Problems

Gap-ETH

Reduction from [CCKLMNT17]

k-DomSet

Gap Amplification

Gap Translation

The Framework
Generalization of Distributed PCP Framework [ARW’17]
Generalization of Distributed PCP Framework [ARW’17]
The Framework Revisited

\begin{itemize}
  \item \textit{W[1] \neq \text{FPT}}
  \item \textit{ETH}
  \item \textit{SETH}
  \item \textit{k-Sum Hyp.}
\end{itemize}

\begin{itemize}
  \item \textit{PSP}
  \item \textit{PSP(MultEQ)}
  \item \textit{PSP(Disj)}
  \item \textit{PSP(SumZero)}
\end{itemize}

\begin{itemize}
  \item \textit{MaxCover}
  \item \textit{$k$-DomSet}
\end{itemize}

\textit{Reduction from [CCKLMNT17]}
Simultaneous Message Passing (SMP) Model

Player /one.taboldstyle

Player /two.taboldstyle

Player $k$

Referee

$x_1$

$x_2$

$x_k$

$f: \{0, 1\}^{m \times k} \rightarrow \{0, 1\}$

Public Randomness

Randomized Protocols:

Completeness: If $f(x_1, \ldots, x_k)$ /equalx 1 then the referee always accepts

Soundness: If $f(x_1, \ldots, x_k)$ /equalx 0 then the referee accepts with probability $\leq s$
Simultaneous Message Passing (SMP) Model

Player 1  Player 2  Player $k$
Simultaneous Message Passing (SMP) Model

Referee

Player 1  Player 2  Player k
Simultaneous Message Passing (SMP) Model

Referee

Player 1 \( x_1 \) \quad Player 2 \( x_2 \) \quad \ldots \quad Player k \( x_k \)
Simultaneous Message Passing (SMP) Model

\[ f : \{0, 1\}^{m \times k} \rightarrow \{0, 1\} \]
Simultaneous Message Passing (SMP) Model

\[ f : \{0, 1\}^{m \times k} \rightarrow \{0, 1\} \]
Simultaneous Message Passing (SMP) Model

\[ f : \{0, 1\}^{m \times k} \rightarrow \{0, 1\} \]

Referee

Public Randomness

Player 1

Player 2

Player k

\[ x_1 \]

\[ x_2 \]

\[ x_k \]
Simultaneous Message Passing (SMP) Model

Randomized Protocols:

**Completeness:** If $f(x_1, \ldots, x_k) = 1$ then the referee always accepts

**Soundness:** If $f(x_1, \ldots, x_k) = 0$ then the referee accepts with probability $\leq s$
Simultaneous Message Passing (SMP) Model

$$f : \{0, 1\}^{m \times k} \rightarrow \{0, 1\}$$

Referee

Public Randomness

Player 1

Player 2

Player k
Simultaneous Message Passing (SMP) Model

\[ f : \{0, 1\}^{m \times k} \rightarrow \{0, 1\} \]

Referee

\[ \mu \in \{0, 1\}^{o(m)} \]

Public Randomness

\[ x_1, x_2, \ldots, x_k \]

Player 1  Player 2  Player k

Completeness: If \( f(x_1, \ldots, x_k) \neq 1 \) then there exists \( \mu \) for which referee always accepts.

Soundness: If \( f(x_1, \ldots, x_k) = 0 \) then for all \( \mu \), the referee accepts with probability \( \leq \frac{s}{8} \).
Simultaneous Message Passing (SMP) Model

\[ f : \{0, 1\}^{m \times k} \rightarrow \{0, 1\} \]

Referee

\[ \mu \in \{0, 1\}^{o(m)} \]

Public Randomness

MA Protocols:

Completeness: If \( f(x_1, \ldots, x_k) = 1 \) then there exists \( \mu \) for which referee always accepts

Soundness: If \( f(x_1, \ldots, x_k) = 0 \) then for all \( \mu \), the referee accepts with probability \( \leq s \)
The Framework Revisited

\[ W[1] \neq \text{FPT} \]

\[ \text{ETH} \]

\[ \text{SETH} \]

\[ k\text{-Sum Hyp.} \]

\[ \text{PSP}(\text{MultE}_Q) \]

\[ \text{PSP}(\text{Disj}) \]

\[ \text{PSP}(\text{SumZero}) \]

\[ \text{MaxCover} \]

Reduction from [CCKLMNT17]

\[ k\text{-DomSet} \]
$k$-SUM problem: Given $A_1, \ldots, A_k \subseteq [-N^{2k}, N^{2k}]$ where $N = \sum_{i\in[k]} |A_i|$, determine whether there exist $x_i \in A_i$, $\forall i \in [k]$ such that $\sum_{i\in[k]} x_i = 0$. 
**k-sum to Maxcover: Proof Sketch**

**k-SUM problem:** Given $A_1, \ldots, A_k \subseteq [-N^{2k}, N^{2k}]$ where $N = \sum_{i\in[k]} |A_i|$, determine whether there exist $x_i \in A_i, \forall i \in [k]$ such that $\sum_{i\in[k]} x_i = 0$.

**SumZero problem:** Player $i$ is given $x_i \in [-N^{2k}, N^{2k}]$ as input. Referee wants to determine whether $\sum_{i\in[k]} x_i = 0$. 

Consider the following randomized protocol for SumZero:

1. The players and referee jointly draw a prime $p^\ast$ in $\{p_1, \ldots, p_r\}$ ($\log r$ random bits).
2. Player $i$ sends $x_i \mod p^\ast$ to the referee ($\log p^\ast$ bits).
3. The referee accepts if the sum of all the numbers he receives is zero.

**Completeness:** If $\sum_{i\in[k]} x_i = 0$ then $\sum_{i\in[k]} x_i \mod p^\ast = 0$.

**Soundness:** If $\sum_{i\in[k]} x_i \neq 0$ then the number of prime factors of $\sum_{i\in[k]} x_i$ is at most $r^\ast \leq 2k \log N + \log k$.

Therefore if $r \geq 2r^\ast$ then the referee rejects with probability $\geq 1/2$.

**Input:** $m$ bits

**Randomness:** $O(\log m)$ bits

**Message Length:** $O(\log m)$ bits

**Soundness:** $1/2$
**k-sum to Maxcover: Proof Sketch**

**k-SUM problem:** Given $A_1, \ldots, A_k \subseteq [-N^{2k}, N^{2k}]$ where $N = \sum_{i \in [k]} |A_i|$, determine whether there exist $x_i \in A_i, \forall i \in [k]$ such that $\sum_{i \in [k]} x_i = 0$.

**SumZero problem:** Player $i$ is given $x_i \in [-N^{2k}, N^{2k}]$ as input. Referee wants to determine whether $\sum_{i \in [k]} x_i = 0$.

Consider the following randomized protocol for SumZero [Nisan’94]:

1. The players and referee jointly draw a prime $p^*$ in $\{p_1, \ldots, p_r\}$ (log $r$ random bits)
2. Player $i$ sends $x_i \mod p^*$ to the referee (log $p^*$ bits)
3. The referee accepts if the sum of all the numbers he receives is zero
**k-sum to Maxcover: Proof Sketch**

**k-SUM problem:** Given \( A_1, \ldots, A_k \subseteq \{-N^{2k}, N^{2k}\} \) where \( N = \sum_{i \in [k]} |A_i| \), determine whether there exist \( x_i \in A_i, \forall i \in [k] \) such that \( \sum_{i \in [k]} x_i = 0 \).

**SumZero problem:** Player \( i \) is given \( x_i \in \{-N^{2k}, N^{2k}\} \) as input. Referee wants to determine whether \( \sum_{i \in [k]} x_i = 0 \).

Consider the following randomized protocol for SumZero [Nisan’94]:

1. The players and referee jointly draw a prime \( p^* \) in \( \{p_1, \ldots, p_r\} \) (log \( r \) random bits)
2. Player \( i \) sends \( x_i \mod p^* \) to the referee (log \( p^* \) bits)
3. The referee accepts if the sum of all the numbers he receives is zero

**Completeness:** If \( \sum_{i \in [k]} x_i = 0 \) then \( \sum_{i \in [k]} x_i \mod p^* = 0 \)
\( k \)-SUM problem: Given \( A_1, \ldots, A_k \subseteq [-N^{2k}, N^{2k}] \) where \( N = \sum_{i \in [k]} |A_i| \), determine whether there exist \( x_i \in A_i, \forall i \in [k] \) such that \( \sum_{i \in [k]} x_i = 0 \).

\( \text{SumZero} \) problem: Player \( i \) is given \( x_i \in [-N^{2k}, N^{2k}] \) as input. Referee wants to determine whether \( \sum_{i \in [k]} x_i = 0 \).

Consider the following randomized protocol for \( \text{SumZero} \) [Nisan’94]:

1. The players and referee jointly draw a prime \( p^* \) in \( \{p_1, \ldots, p_r\} \) (\( \log r \) random bits)
2. Player \( i \) sends \( x_i \mod p^* \) to the referee (\( \log p^* \) bits)
3. The referee accepts if the sum of all the numbers he receives is zero

Completeness: If \( \sum_{i \in [k]} x_i = 0 \) then \( \sum_{i \in [k]} x_i \mod p^* = 0 \)

Soundness: If \( \sum_{i \in [k]} x_i \neq 0 \) then the number of prime factors of \( \sum_{i \in [k]} x_i \) is at most \( r^* = 2k \log N + \log k \).
**k-sum to Maxcover: Proof Sketch**

**k-SUM problem:** Given \( A_1, \ldots, A_k \subseteq [-N^{2k}, N^{2k}] \) where \( N = \sum_{i \in [k]} |A_i| \), determine whether there exist \( x_i \in A_i, \forall i \in [k] \) such that \( \sum_{i \in [k]} x_i = 0 \).

**SumZero problem:** Player \( i \) is given \( x_i \in [-N^{2k}, N^{2k}] \) as input. Referee wants to determine whether \( \sum_{i \in [k]} x_i = 0 \).

Consider the following **randomized** protocol for SumZero [Nisan’94]:

1. The players and referee jointly draw a prime \( p^* \) in \( \{p_1, \ldots, p_r\} \) (log \( r \) random bits)
2. Player \( i \) sends \( x_i \mod p^* \) to the referee (log \( p^* \) bits)
3. The referee accepts if the sum of all the numbers he receives is zero

**Completeness:** If \( \sum_{i \in [k]} x_i = 0 \) then \( \sum_{i \in [k]} x_i \mod p^* = 0 \)

**Soundness:** If \( \sum_{i \in [k]} x_i \neq 0 \) then the number of prime factors of \( \sum_{i \in [k]} x_i \) is at most \( r^* = 2k \log N + \log k \). Therefore if \( r \geq 2r^* \) then the referee rejects with probability \( \geq 1/2 \).
\textbf{\textit{k-sum to Maxcover: Proof Sketch}}

\textbf{\textit{k-SUM problem}}: Given $A_1, \ldots, A_k \subseteq [-N^{2k}, N^{2k}]$ where $N = \sum_{i \in [k]} |A_i|$, determine whether there exist $x_i \in A_i, \forall i \in [k]$ such that $\sum_{i \in [k]} x_i = 0$.

\textbf{\textit{SumZero problem}}: Player $i$ is given $x_i \in [-N^{2k}, N^{2k}]$ as input. Referee wants to determine whether $\sum_{i \in [k]} x_i = 0$.

Consider the following \textit{randomized} protocol for \textit{SumZero} [Nisan’94]:

1. The players and referee jointly draw a prime $p^*$ in \{\(p_1, \ldots, p_r\}\) (\(\log r\) random bits)
2. Player $i$ sends $x_i \mod p^*$ to the referee (\(\log p^*\) bits)
3. The referee accepts if the sum of all the numbers he receives is zero

\textbf{Completeness}: If $\sum_{i \in [k]} x_i = 0$ then $\sum_{i \in [k]} x_i \mod p^* = 0$

\textbf{Soundness}: If $\sum_{i \in [k]} x_i \neq 0$ then the number of prime factors of $\sum_{i \in [k]} x_i$ is at most $r^* = 2k \log N + \log k$. Therefore if $r \geq 2r^*$ then the referee rejects with probability $\geq 1/2$

\textbf{Input}: \(m\) bits \hspace{1cm} \textbf{Randomness}: \(O(\log m)\) bits

\textbf{Message Length}: \(O(\log m)\) bits \hspace{1cm} \textbf{Soundness}: \(1/2\)
Parameters of the SumZero protocol [Nisan’94]:

- **Input:** $m$ bits
- **Message Length:** $O(\log m)$ bits
- **Randomness:** $O(\log m)$ bits
- **Soundness:** $\frac{1}{2}$
Parameters of the SumZero protocol [Nisan’94]:

**Input:** $m$ bits

**Randomness:** $O(\log m)$ bits

**Message Length:** $O(\log m)$ bits

**Soundness:** $\frac{1}{2}$
Parameters of the SumZero protocol [Nisan’94]:

**Input:** $m$ bits

**Message Length:** $O(\log m)$ bits

**Randomness:** $O(\log m)$ bits

**Soundness:** $1/2$

Nodes in $p_i$ are all $(z_1, \ldots, z_k) \in \mathbb{Z}_p^k$ such that $\sum_{j \in [k]} z_j = 0 \mod p_i$
Parameters of the SumZero protocol [Nisan’94]:

**Input**: \( m \) bits

**Randomness**: \( O(\log m) \) bits

**Message Length**: \( O(\log m) \) bits

**Soundness**: \( \frac{1}{2} \)

Nodes in \( p_i \) are all \( (z_1, \ldots, z_k) \in \mathbb{Z}_p^k \) such that \( \sum_{j \in [k]} z_j = 0 \mod p_i \)

For every \( x \in A_j \) and \( z = (z_1, \ldots, z_k) \in p_i \), \( (x, z) \in E \iff z_j = x \mod p_i \)
Parameters of the SumZero protocol [Nisan'94]:

**Input:** $m$ bits  
**Randomness:** $O(\log m)$ bits  
**Message Length:** $O(\log m)$ bits  
**Soundness:** $1/2$

Nodes in $p_i$ are all $(z_1, \ldots, z_k) \in \mathbb{Z}_p^k$ such that $\sum_{j \in [k]} z_j = 0 \mod p_i$

For every $x \in A_j$ and $z = (z_1, \ldots, z_k) \in p_i$, 
$(x, z) \in E \iff z_j = x \mod p_i$

A labeling $(x_1, \ldots, x_k)$ covers $p_i$

The referee accepts on random prime $p_i$
Parameters of the SumZero protocol [Nisan’94]:

**Input:** \( m \) bits

**Randomness:** \( O(\log m) \) bits

**Message Length:** \( O(\log m) \) bits

**Soundness:** \( 1/2 \)

Nodes in \( p_i \) are all \( (z_1, \ldots, z_k) \in \mathbb{Z}_p^k \) such that \( \sum_{j \in [k]} z_j = 0 \mod p_i \)

For every \( x \in A_j \) and \( z = (z_1, \ldots, z_k) \in p_i \), \( (x, z) \in E \iff z_j = x \mod p_i \)

A labeling \( (x_1, \ldots, x_k) \) covers \( p_i \)

The referee accepts on random prime \( p_i \)

Soundness of SumZero protocol

Soundness of MaxCover
The Framework Revisited

W[1] ≠ FPT

ETH

SETH

k-Sum Hyp.

PSP

PSP(MultEQ)

PSP(Disj)

PSP(SumZero)

MaxCover

Reduction from [CCKLMNT17]

k-DomSet

MaxCover → k-DomSet

Reduction from [CCKLMNT17]

MaxCover

PSP

PSP(MultEQ)

PSP(Disj)

PSP(SumZero)
Let $f : \{0, 1\}^{m \times k} \rightarrow \{0, 1\}$

**Problem:** PSP($f$)

**Input:** $A_1, \ldots, A_k \subseteq \{0, 1\}^m$ where $|A_i| \leq N$

**Output:** Determine if $\exists a_i \in A_i, \forall i \in [k]$, such that $f(a_1, \ldots, a_k) = 1$
Product Space Problems

Let $f : \{0, 1\}^{m \times k} \to \{0, 1\}$

**Problem:** PSP($f$)

**Input:** $A_1, \ldots A_k \subseteq \{0, 1\}^m$ where $|A_i| \leq N$

**Output:** Determine if $\exists a_i \in A_i, \forall i \in [k]$, such that $f(a_1, \ldots, a_k) = 1$

---

**Product Space Problem (PSP)**

Let $m : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be any function and $\mathcal{F}$ be a family of Boolean functions indexed by $N$ and $k$ as follows: $\mathcal{F} := \{ f_{N,k} : \{0, 1\}^{m(N,k) \times k} \to \{0, 1\} \}_{N,k \in \mathbb{N}}$. 
Let $f : \{0,1\}^{m \times k} \to \{0,1\}$

**Problem:** PSP$(f)$

**Input:** $A_1, \ldots, A_k \subseteq \{0,1\}^m$ where $|A_i| \leq N$

**Output:** Determine if $\exists a_i \in A_i, \forall i \in [k]$, such that $f(a_1,\ldots,a_k) = 1$

---

**Product Space Problem (PSP)**

Let $m : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be any function and $\mathcal{F}$ be a family of Boolean functions indexed by $N$ and $k$ as follows: $\mathcal{F} := \{f_{N,k} : \{0,1\}^{m(N,k) \times k} \to \{0,1\}\}_{N,k \in \mathbb{N}}$.

For each $k \in \mathbb{N}$, the *product space problem* PSP$(k, \mathcal{F})$ of order $N$ is defined as follows: given $k$ subsets $A_1, \ldots, A_k$ of $\{0,1\}^{m(N,k)}$ each of cardinality at most $N$ as input, determine if there exists $(a_1, \ldots, a_k) \in A_1 \times \cdots \times A_k$ such that $f_{N,k}(a_1, \ldots, a_k) = 1$. 
Let $f : \{0, 1\}^{m \times k} \to \{0, 1\}$

**Problem:** PSP($f$)

**Input:** $A_1, \ldots, A_k \subseteq \{0, 1\}^m$ where $|A_i| \leq N$

**Output:** Determine if $\exists a_i \in A_i, \forall i \in [k]$, such that $f(a_1, \ldots, a_k) = 1$

**Product Space Problem (PSP)**

Let $m : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be any function and $\mathcal{F}$ be a family of Boolean functions indexed by $N$ and $k$ as follows: $\mathcal{F} := \{ f_{N,k} : \{0, 1\}^{m(N,k) \times k} \to \{0, 1\} \}_{N,k \in \mathbb{N}}$.

For each $k \in \mathbb{N}$, the *product space problem* PSP($k, \mathcal{F}$) of order $N$ is defined as follows: given $k$ subsets $A_1, \ldots, A_k$ of $\{0, 1\}^{m(N,k)}$ each of cardinality at most $N$ as input, determine if there exists $(a_1, \ldots, a_k) \in A_1 \times \cdots \times A_k$ such that $f_{N,k}(a_1, \ldots, a_k) = 1$.

For the rest of the talk, $m(N, k) = \text{poly}(k) \cdot \log N$. 
The Framework Revisited

- **W[1] ≠ FPT**
- **ETH**
- **SETH**
- **k-Sum Hyp.**

- **PSP**
  - **PSP(MultEQ)**
  - **PSP(Disj)**
  - **PSP(SumZero)**

- **MaxCover**

Reduction from [CCKLMNT17] to **k-DomSet**
The Framework Revisited

**SumZero**: \( \{0, 1\}^{m \times k} \rightarrow \{0, 1\} \),

\[
\text{SumZero}(x_1, \ldots, x_k) = \begin{cases} 
1 & \text{if } \sum_{i \in [k]} x_i = 0, \\
0 & \text{otherwise}.
\end{cases}
\]
The Framework Revisited

W[1] \neq FPT

ETH

SETH

k-Sum Hyp.

PSP

PSP(MultEQ)

PSP(Disj)

PSP(SumZero)

MaxCover

Reduction from \[\text{CCKLMNT}17\]

\(k\)-DomSet
The Framework Revisited

W[1] ≠ FPT

ETH

SETH

k-Sum

PSP

PSP(MultEQ)

PSP(Disj)

MaxCover

Reduction from [CCKLMNT17]

k-DomSet

\[ \text{Disj} : \{0, 1\}^{m \times k} \to \{0, 1\}, \]

\[ \text{Disj}(x_1, \ldots, x_k) = \neg \left( \bigvee_{i \in [m]} \left( \bigwedge_{j \in [k]} (x_j)_i \right) \right). \]
The Framework Revisited

PSP

- \(W[1] \neq \text{FPT}\)
- \(\text{ETH}\)
- \(\text{SETH}\)
- \(k\)-Sum Hyp.

PSP\(\text{MULTEQ}\)

PSP\(\text{DISJ}\)

PSP\(\text{SUMZERO}\)

MaxCover

Reduction from \([\text{CCKLMN17}]\)

\(k\)-DomSet
Popular Hypotheses to PSP

$\text{SETH} \implies \text{PSP}(\text{Disj})$

Let $X = X_1 \cup \cdots \cup X_k$. For every partial assignment $\sigma$ to $X_i$, we build $\sigma \in A_i \subseteq \{0, 1\}^m$ as follows:

$a_\sigma(j) = \begin{cases} 0 & \text{if } \sigma \text{ satisfies the } j\text{th clause} \\ 1 & \text{otherwise} \end{cases}$

Note from above that $\text{ETH} \implies \text{PSP}(\text{Disj})$. We will skip $\text{ETH} \implies \text{PSP}(\text{Majority})$.

$\text{W/G, FPT} \implies \text{PSP}(\text{Majority})$

Starting point: $\ell$-clique problem on graph $G(V, E)$. Let $k = \ell^2$ and set $A_i = E$, i.e., each edge $\in \{0, 1\}^{\log |V| \times \{\bot, \top\}}$.

Check for each vertex that the $\ell$ incident edges have assigned the same vertex (equality checking).
SETH $\implies$ PSP(D\text{Disj})

Let $X = X_1 \cup \cdots X_k$

For every partial assignment $\sigma$ to $X_i$, we build $a_\sigma \in A_i \subseteq \{0, 1\}^m$ as follows:

$$a_\sigma(j) = \begin{cases} 
0 & \text{if } \sigma \text{ satisfies } j^{th} \text{ clause} \\
1 & \text{otherwise}
\end{cases}$$
**SETH \(\implies\) PSP(D_{\text{Disj}})**

Let \(X = X_1 \cup \cdots X_k\)

For every partial assignment \(\sigma\) to \(X_i\), we build \(a_\sigma \in A_i \subseteq \{0, 1\}^m\) as follows:

\[
a_\sigma(j) = \begin{cases} 
0 & \text{if } \sigma \text{ satisfies } j^{\text{th}} \text{ clause} \\
1 & \text{otherwise}
\end{cases}
\]

Note from above that ETH \(\implies\) PSP(D_{\text{Disj}}). We will skip ETH \(\implies\) PSP(MultEQ)
SETH $\implies$ PSP($\text{Disj}$)

Let $X = X_1 \cup \cdots X_k$

For every partial assignment $\sigma$ to $X_i$, we build $a_\sigma \in A_i \subseteq \{0, 1\}^m$ as follows:

$$a_\sigma(j) = \begin{cases} 
0 & \text{if } \sigma \text{ satisfies } j^{\text{th}} \text{ clause} \\
1 & \text{otherwise}
\end{cases}$$

Note from above that ETH $\implies$ PSP($\text{Disj}$). We will skip ETH $\implies$ PSP($\text{MultEQ}$)

$W[1] \neq \text{FPT} \implies$ PSP($\text{MultEQ}$)
Popular Hypotheses to PSP

\[
\text{SETH} \implies \text{PSP}(\text{Disj})
\]

Let \( X = X_1 \cup \cdots X_k \)

For every partial assignment \( \sigma \) to \( X_i \), we build \( a_{\sigma} \in A_i \subseteq \{0, 1\}^m \) as follows:

\[
a_{\sigma}(j) = \begin{cases} 
0 & \text{if } \sigma \text{ satisfies } j^{\text{th}} \text{ clause} \\
1 & \text{otherwise}
\end{cases}
\]

Note from above that \( \text{ETH} \implies \text{PSP}(\text{Disj}) \). We will skip \( \text{ETH} \implies \text{PSP}(\text{MultEQ}) \)

\[
\text{W[1] \#FPT} \implies \text{PSP}(\text{MultEQ})
\]

Starting point: \( \ell \)-clique problem on graph \( G(V, E) \)
Popular Hypotheses to PSP

\[ \text{SETH} \implies \text{PSP(Disj)} \]

Let \( X = X_1 \cup \cdots \cup X_k \)

For every partial assignment \( \sigma \) to \( X_i \), we build \( a_\sigma \in A_i \subseteq \{0, 1\}^m \) as follows:

\[
a_\sigma(j) = \begin{cases} 
0 & \text{if } \sigma \text{ satisfies } j^{\text{th}} \text{ clause} \\
1 & \text{otherwise}
\end{cases}
\]

Note from above that \( \text{ETH} \implies \text{PSP(Disj)} \). We will skip \( \text{ETH} \implies \text{PSP(MultEq)} \)

\[ W[1] \neq \text{FPT} \implies \text{PSP(MultEq)} \]

Starting point: \( \ell \)-clique problem on graph \( G(V, E) \)

Let \( k = \binom{\ell}{2} \) and set \( A_i = E \), i.e., each edge \( \in \left( \{0, 1\}^{\log |V|} \times \{\bot, \top\} \right)^\ell \)
**Popular Hypotheses to PSP**

**SETH \(\implies\) PSP(Disj)**

Let \(X = X_1 \cup \cdots \cup X_k\)

For every **partial assignment** \(\sigma\) to \(X_i\), we build \(a_\sigma \in A_i \subseteq \{0, 1\}^m\) as follows:

\[
a_\sigma(j) = \begin{cases} 
0 & \text{if } \sigma \text{ satisfies } j^{\text{th}} \text{ clause} \\
1 & \text{otherwise}
\end{cases}
\]

Note from above that **ETH \(\implies\) PSP(Disj)**. We will skip **ETH \(\implies\) PSP(MultEQ)**

**W[1] \(\not=\) FPT \(\implies\) PSP(MultEQ)**

**Starting point:** \(\ell\)-clique problem on graph \(G(V, E)\)

Let \(k = \binom{\ell}{2}\) and set \(A_i = E\), i.e., each edge \(\in \left(\{0, 1\}^{\log|V|} \times \{\bot, \top\}\right)^{\ell}\)

Check for each vertex that the \(\ell\) incident edges have assigned the same vertex (equality checking)
The Framework Revisited

- **W[1] ≠ FPT**
- **ETH**
- **SETH**
- **k-Sum Hyp.**

The diagram shows the relationships between various complexity classes and problems:

- **PSP**
  - **PSP(MultEQ)**
  - **PSP(Disj)**
  - **PSP(SumZero)**

- **MaxCover**

Reduction from [CCKLMNT17]

**k-DomSet**
Determine if $\text{MaxCover}(\Gamma) = 1$ or $\text{MaxCover}(\Gamma) \leq s$.

Each $W_i$ is a Right Super Node.

Each $U_i$ is a Left Super Node.

$S \subseteq W$ is a labeling of $W$ if $\forall i \in [k]$, $|S \cap W_i| = 1$.

$S$ covers $U_i$ if $\exists u \in U_i$, $\forall v \in S$, $(u, v) \in E$.

$\text{MaxCover}(\Gamma, S)$ is the fraction of $U_i$'s covered by $S$.

$\text{MaxCover}(\Gamma)$ is the maximum $\text{MaxCover}(\Gamma, S)$.
Determine if \( \text{MaxCover}(\Gamma, S) = 1 \) or \( \text{MaxCover}(\Gamma, S) \leq s \).

Each \( W_i \) is a **Right Super Node**
Each \( U_i \) is a **Left Super Node**
Determine if $\text{MaxCover}(\Gamma)$ is equal to 1 or $\text{MaxCover}(\Gamma) \leq s$.

Each $W_i$ is a **Right Super Node**.
Each $U_i$ is a **Left Super Node**.

$S \subseteq W$ is a **labeling** of $W$ if

$$\forall i \in [k], |S \cap W_i| = 1$$
Maxcover [CCKLMNT’17]

Each $W_i$ is a **Right Super Node**
Each $U_i$ is a **Left Super Node**

$S \subseteq W$ is a **labeling** of $W$ if
\[ \forall i \in [k], |S \cap W_i| = 1 \]

$S$ **covers** $U_i$ if
\[ \exists u \in U_i, \forall v \in S, (u, v) \in E \]
Each $W_i$ is a **Right Super Node**
Each $U_i$ is a **Left Super Node**

$S \subseteq W$ is a **labeling** of $W$ if

$\forall i \in [k], |S \cap W_i| = 1$

$S$ **covers** $U_i$ if

$\exists u \in U_i, \forall v \in S, (u, v) \in E$

$\text{MaxCover}(\Gamma, S) =$ Fraction of $U_i$’s covered by $S$
Each $W_i$ is a **Right Super Node**
Each $U_i$ is a **Left Super Node**

$S \subseteq W$ is a **labeling** of $W$ if

\[ \forall i \in [k], |S \cap W_i| = 1 \]

$S$ **covers** $U_i$ if

\[ \exists u \in U_i, \forall v \in S, (u, v) \in E \]

$\text{MaxCover}(\Gamma, S) = \text{Fraction of } U_i \text{'s covered by } S$

$\text{MaxCover}(\Gamma) = \max_S \text{MaxCover}(\Gamma, S)$
Maxcover [CCKLMNT’17]

Each \( W_i \) is a Right Super Node
Each \( U_i \) is a Left Super Node

\( S \subseteq W \) is a labeling of \( W \) if
\[
\forall i \in [k], |S \cap W_i| = 1
\]

\( S \) covers \( U_i \) if
\[
\exists u \in U_i, \forall v \in S, (u, v) \in E
\]

\( \Gamma(U, W, E) \)

\[ \text{MaxCover}(\Gamma, S) = \frac{\text{Fraction of } U_i \text{'s covered by } S}{\text{MaxCover}(\Gamma)} = \max_S \text{MaxCover}(\Gamma, S) \]

Determine if \( \text{MaxCover}(\Gamma) = 1 \)
or \( \text{MaxCover}(\Gamma) \leq s \)
The Framework Revisited

W[1] ≠ FPT

ETH

SETH

k-Sum Hyp.

PSP

PSP(MultEQ)

PSP(Disj)

PSP(SumZero)

MaxCover

Reduction from [CCKLMNT17]

k-DomSet
Parameters of SMP protocol $\Pi$ for $f : \{0, 1\}^{m \times k} \rightarrow \{0, 1\}$:

Advice: $\gamma$ bits

Randomness: $R$ bits

Message Length: $L$ bits

Soundness: $s$
PSP to Maxcover

Parameters of SMP protocol $\Pi$ for $f : \{0, 1\}^{m \times k} \rightarrow \{0, 1\}$:

- **Advice:** $\gamma$ bits
- **Randomness:** $R$ bits
- **Message Length:** $L$ bits
- **Soundness:** $s$

![Diagram of $\Gamma(U, W, E)$]
Parameters of SMP protocol $\Pi$ for $f : \{0, 1\}^{m\times k} \rightarrow \{0, 1\}$:

**Advice:** $\gamma$ bits

**Randomness:** $R$ bits

**Message Length:** $L$ bits

**Soundness:** $s$

![Diagram](attachment:diagram.png)
PSP to Maxcover

Parameters of SMP protocol $\Pi$ for $f : \{0, 1\}^{m \times k} \rightarrow \{0, 1\}$:

- **Advice**: $\gamma$ bits
- **Randomness**: $R$ bits
- **Message Length**: $L$ bits
- **Soundness**: $s$

- $2^\gamma$ instances of MaxCover

$\Gamma(U, W, E)$
PSP to Maxcover

Parameters of SMP protocol $\Pi$ for $f : \{0, 1\}^{m \times k} \rightarrow \{0, 1\}$:

- **Advice:** $\gamma$ bits
- **Randomness:** $R$ bits
- **Message Length:** $L$ bits
- **Soundness:** $s$

$2^\gamma$ instances of MaxCover

Nodes in $U_i$ are all $k$-tuples of messages that referee accepts on randomness $i$ and advice $\mu \in \{0, 1\}^\gamma$.
PSP to Maxcover

Parameters of SMP protocol $\Pi$ for $f : \{0, 1\}^{m \times k} \rightarrow \{0, 1\}$:

- **Advice**: $\gamma$ bits
- **Randomness**: $R$ bits
- **Message Length**: $L$ bits
- **Soundness**: $s$

Nodes in $U_i$ are all $k$-tuples of messages that referee **accepts** on randomness $i$ and advice $\mu \in \{0, 1\}^\gamma$

For every $x \in A_j$ and $z = (z_1, \ldots, z_k) \in U_i$, $(x, z) \in E \iff z_j$ is message of player $j$ on input $x$ and randomness $i$

$2^\gamma$ instances of MaxCover
Parameters of SMP protocol $\Pi$ for $f : \{0,1\}^{m \times k} \rightarrow \{0,1\}$:

- **Advice:** $\gamma$ bits
- **Randomness:** $R$ bits
- **Message Length:** $L$ bits
- **Soundness:** $s$

$2^\gamma$ instances of MaxCover

Nodes in $U_i$ are all $k$-tuples of messages that referee accepts on randomness $i$ and advice $\mu \in \{0,1\}^\gamma$

For every $x \in A_j$ and $z = (z_1, \ldots, z_k) \in U_i$, $(x, z) \in E \iff z_j$ is message of player $j$ on input $x$ and randomness $i$

Soundness of $\Pi$

Soundness of MaxCover
The Framework Revisited

\begin{itemize}
  \item \textbf{W[1] \neq FPT}
  \item \textbf{ETH}
  \item \textbf{SETH}
  \item \textbf{$k$-Sum Hyp.}
\end{itemize}

\begin{itemize}
  \item \textbf{PSP}
  \item \textbf{PSP(MultEq)}
  \item \textbf{PSP(Disj)}
  \item \textbf{PSP(SumZero)}
  \item \textbf{MaxCover}
  \item \textbf{$k$-DomSet}
\end{itemize}

Reduction from [CCKLMNT17]
Maxcover to Parameterized Dominating Set

Reduction from MaxCover to $k$-DomSet [CCKLMNT17]

There is a reduction from MaxCover instance $\Gamma = \left( U = \bigcup_{j=1}^{r} U_j, W = \bigcup_{j=1}^{k} W_i, E \right)$ to a $k$-DomSet instance $G$ such that
Reduction from MaxCover to $k$-DomSet [CCKLMNT17]

There is a reduction from MaxCover instance $\Gamma = \left( U = \bigcup_{j=1}^{r} U_j, W = \bigcup_{j=1}^{k} W_i, E \right)$ to a $k$-DomSet instance $G$ such that

- If $\text{MaxCover}(\Gamma) = 1$, then $\text{DomSet}(G) = k$
- If $\text{MaxCover}(\Gamma) \leq \epsilon$, then $\text{DomSet}(G) \geq \left(\frac{1}{\epsilon}\right)^{1/k} \cdot k$
Reduction from MaxCover to $k$-DomSet [CCKLMNT17]

There is a reduction from MaxCover instance $\Gamma = \left( U = \bigcup_{j=1}^{r} U_j, W = \bigcup_{j=1}^{k} W_i, E \right)$ to a $k$-DomSet instance $G$ such that

- If $\text{MaxCover}(\Gamma) = 1$, then $\text{DomSet}(G) = k$
- If $\text{MaxCover}(\Gamma) \leq \varepsilon$, then $\text{DomSet}(G) \geq (1/\varepsilon)^{1/k} \cdot k$
- $|V(G)| = |W| + \sum_{j \in [r]} k|U_j|$
Maxcover to Parameterized Dominating Set

Reduction from MaxCover to $k$-DomSet [CCKLMNT17]

There is a reduction from MaxCover instance $\Gamma = \left( U = \bigcup_{j=1}^{r} U_j, W = \bigcup_{j=1}^{k} W_i, E \right)$ to a $k$-DomSet instance $G$ such that

- If $\text{MaxCover}(\Gamma) = 1$, then $\text{DomSet}(G) = k$
- If $\text{MaxCover}(\Gamma) \leq \epsilon$, then $\text{DomSet}(G) \geq \left( \frac{1}{\epsilon} \right)^{1/k} \cdot k$
- $|V(G)| = |W| + \sum_{j \in [r]} k|U_j|$
- The reduction runs in time $O \left( |W| \left( \sum_{j \in [r]} k|U_j| \right) \right)$. 
Maxcover to Parameterized Dominating Set

Reduction from MaxCover to $k$-DomSet [CCKLMNT17]

There is a reduction from MaxCover instance $\Gamma = \left( U = \bigcup_{j=1}^{r} U_j, W = \bigcup_{j=1}^{k} W_i, E \right)$ to a $k$-DomSet instance $G$ such that

- If $\text{MaxCover}(\Gamma) = 1$, then $\text{DomSet}(G) = k$
- If $\text{MaxCover}(\Gamma) \leq \varepsilon$, then $\text{DomSet}(G) \geq \left( \frac{1}{\varepsilon} \right)^{1/k} \cdot k$
- $|V(G)| = |W| + \sum_{j \in [r]} k|U_j|$
- The reduction runs in time $O\left(|W| \left( \sum_{j \in [r]} k|U_j| \right)\right)$.

We want $1/\varepsilon = \omega(1)$ and $|U_j| = o(m)$
Greedyly we want SMP protocols:

**Input**: $m$ bits  
**Randomness**: $\text{polylog}(m)$ bits

**Message Length**: $O_k(1)$ bits  
**Soundness**: $1/2$
SMP Protocol for \( k \)-sumZero

SMP Protocol of Nisan [Nisan’94]:

**Input:** \( m \) bits

**Randomness:** \( O(\log m) \) bits

**Message Length:** \( O(\log m) \) bits

**Soundness:** \( 1/2 \)
SMP Protocol for $k$-sumZero

SMP Protocol of Nisan [Nisan’94]:

**Input**: $m$ bits

**Randomness**: $O(\log m)$ bits

**Message Length**: $O(\log m)$ bits

**Soundness**: $\frac{1}{2}$

SMP Protocol of Viola [Viola’15]:

**Input**: $m$ bits

**Randomness**: $O(m)$ bits

**Message Length**: $O_k(1)$ bits

**Soundness**: $\frac{1}{2}$
SMP Protocol for $k$-sumZero

SMP Protocol of Nisan [Nisan’94]:

Input: $m$ bits
Randomness: $O(\log m)$ bits
Message Length: $O(\log m)$ bits
Soundness: $1/2$

SMP Protocol of Viola [Viola’15]:

Input: $m$ bits
Randomness: $O(m)$ bits
Message Length: $O_k(1)$ bits
Soundness: $1/2$

New SMP Protocol:

Input: $m$ bits
Randomness: $O_k(\log m)$ bits
Message Length: $O_k(1)$ bits
Soundness: $1/2$
Idea: Use any binary code of constant rate and distance $\delta$
Idea: Use any binary code of constant rate and distance $\delta$

SMP Protocol Parameters:

- **Input**: $m$ bits
- **Randomness**: $O(\log m)$ bits
- **Message Length**: $O(1)$ bits
- **Soundness**: $1 - \delta$
SMP Protocol for $k$-disjointness

A straightforward extension of Rubinstein’s two-party protocol [R’18,ARW’17,AW’09]
A straightforward extension of Rubinstein’s two-party protocol [R’18,ARW’17,AW’09]

**Good Pointwise Product (GPP) Codes**

Let $q$ be a prime power and $k \in \mathbb{N}$. A code $C$ over $\mathbb{F}_q$ is said to be a $q$-GHP code if there exists a constant $\delta(k) > 0$ such that the following holds.

- $C$ is systematic and can be encoded efficiently.
- Let $C^k$ be the set of all $k$-pointwise product of codewords of $C$. Then, there exists a linear good code $\tilde{C}$ such that $C^k \subseteq \tilde{C}$, i.e., $\tilde{C}$ has relative distance and rate greater than $\delta$. 
SMP Protocol for $k$-disjointness

A straightforward extension of Rubinstein’s two-party protocol [R’18,ARW’17,AW’09]

**Good Pointwise Product (GPP) Codes**

Let $q$ be a prime power and $k \in \mathbb{N}$. A code $C$ over $\mathbb{F}_q$ is said to be a $q$-GHP code if there exists a constant $\delta(k) > 0$ such that the following holds.

- $C$ is systematic and can be encoded efficiently.
- Let $C^k$ be the set of all $k$-pointwise product of codewords of $C$. Then, there exists a linear good code $\tilde{C}$ such that $C^k \subseteq \tilde{C}$, i.e., $\tilde{C}$ has relative distance and rate greater than $\delta$.

- Player $i$ divides his input $x_i$ into $T$ parts $x_{i1}^1, \ldots x_{iT}^T$
SMP Protocol for $k$-disjointness

A straightforward extension of Rubinstein’s two-party protocol [R’18,ARW’17,AW’09]

Good Pointwise Product (GPP) Codes

Let $q$ be a prime power and $k \in \mathbb{N}$. A code $C$ over $\mathbb{F}_q$ is said to be a $q$-GHP code if there exists a constant $\delta(k) > 0$ such that the following holds.

- $C$ is systematic and can be encoded efficiently.
- Let $C^k$ be the set of all $k$-pointwise product of codewords of $C$. Then, there exists a linear good code $\tilde{C}$ such that $C^k \subseteq \tilde{C}$, i.e., $\tilde{C}$ has relative distance and rate greater than $\delta$.

- Player $i$ divides his input $x_i$ into $T$ parts $x_i^1, \ldots, x_i^T$.
- The advice $\mu$ of the referee is $\sum_{j \in [T]} \prod_{\ell \in [k]} C(x_i^j) \text{ – a codeword of } \tilde{C}$!
SMP Protocol for $k$-disjointness

A straightforward extension of Rubinstein’s two-party protocol [R’18,ARW’17,AW’09]

Good Pointwise Product (GPP) Codes

Let $q$ be a prime power and $k \in \mathbb{N}$. A code $C$ over $\mathbb{F}_q$ is said to be a $q$-GHP code if there exists a constant $\delta(k) > 0$ such that the following holds.

- $C$ is systematic and can be encoded efficiently.
- Let $C^k$ be the set of all $k$-pointwise product of codewords of $C$. Then, there exists a linear good code $\widetilde{C}$ such that $C^k \subseteq \widetilde{C}$, i.e., $\widetilde{C}$ has relative distance and rate greater than $\delta$.

- Player $i$ divides his input $x_i$ into $T$ parts $x_{i}^1, \ldots, x_{i}^T$
- The advice $\mu$ of the referee is $\sum_{j \in [T]} \prod_{\ell \in [k]} C(x_{i}^j) - a codeword of \widetilde{C}$!
- Referee checks that $\mu$ is zero in the systematic part and on a random coordinate
SMP Protocol for $k$-disjointness

A straightforward extension of Rubinstein’s two-party protocol [R’18, ARW’17, AW’09]

Good Pointwise Product (GPP) Codes

Let $q$ be a prime power and $k \in \mathbb{N}$. A code $C$ over $\mathbb{F}_q$ is said to be a $q$-GHP code if there exists a constant $\delta(k) > 0$ such that the following holds.

- $C$ is systematic and can be encoded efficiently.
- Let $C^k$ be the set of all $k$-pointwise product of codewords of $C$. Then, there exists a linear good code $\tilde{C}$ such that $C^k \subseteq \tilde{C}$, i.e., $\tilde{C}$ has relative distance and rate greater than $\delta$.

- Player $i$ divides his input $x_i$ into $T$ parts $x_i^1, \ldots, x_i^T$.
- The advice $\mu$ of the referee is $\sum_{j \in [T]} \prod_{\ell \in [k]} C(x_{j \ell}^j)$ – a codeword of $\tilde{C}$!
- Referee checks that $\mu$ is zero in the systematic part and on a random coordinate.

Advice: $O_k(m/T \log q)$ bits
Randomness: $O_k(\log m)$ bits
Message Length: $T \log q$ bits
Soundness: $1 - \delta$
SMP Protocol Parameters:

**Advice:** $O_k(m/T \log q)$ bits

**Randomness:** $O_k(\log m)$ bits

**Message Length:** $T \log q$ bits

**Soundness:** $1 - \delta$
### SMP Protocol Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advice</strong></td>
<td>$O_k\left(\frac{m}{T \log q}\right)$ bits</td>
</tr>
<tr>
<td><strong>Randomness</strong></td>
<td>$O_k(\log m)$ bits</td>
</tr>
<tr>
<td><strong>Message Length</strong></td>
<td>$T \log q$ bits</td>
</tr>
<tr>
<td><strong>Soundness</strong></td>
<td>$1 - \delta$</td>
</tr>
</tbody>
</table>

### Reed Solomon Codes

Let $\ell \in \mathbb{N}$ and $q$ be a prime number in $[4\ell, 8\ell)$. Then, there exists a $q$-GPP code of message length $\ell$. 

---

Algebraic Geometric Codes [GS’96, SAKSD’01]

There exists a constant $c \in \mathbb{N}$ such that for any prime number $q$ greater than $c$ there is a $q^2$-GPP code for every message length $\ell \in \mathbb{N}$. 

"/two.taboldstyle/eight.taboldstyle"
SMP Protocol Parameters:

**Advice:** $O_k(m/T \log q)$ bits

**Message Length:** $T \log q$ bits

**Randomness:** $O_k(\log m)$ bits

**Soundness:** $1 - \delta$

---

**Reed Solomon Codes**

Let $\ell \in \mathbb{N}$ and $q$ be a prime number in $[4\ell, 8\ell)$. Then, there exists a $q$-GPP code of message length $\ell$.

---

**Algebraic Geometric Codes [GS’96, SAKSD’01]**

There exists a constant $c \in \mathbb{N}$ such that for any prime number $q$ greater than $c$ there is a $q^2$-GPP code for every message length $\ell \in \mathbb{N}$. 
Recap of the Results

- Any $T(k)$ approximation is $W[1]$-hard
- No $T(k)$ approximation algorithm in $N^{o(k)}$ time, assuming ETH
- No $T(k)$ approximation algorithm in $N^{k-\varepsilon}$ time, assuming SETH
- No $T(k)$ approximation algorithm in $N^{\lceil k/2 \rceil - \varepsilon}$ time, assuming $k$-SUM Hypothesis
Summary of the Framework

- **W[1] ≠ FPT**
- **ETH**
- **SETH**
- **k-Sum Hyp.**

**PSP**
- **PSP(MultEQ)**
- **PSP(Disj)**
- **PSP(SumZero)**

**MaxCover**

Reduction from [CCKLMNT17]

**k-DomSet**
Important Open Questions

- Parameterized Dominating Set is $W[2]$-complete. Can we show every $T(k)$ approximation is also $W[2]$-hard?
Important Open Questions

- Parameterized Dominating Set is $W[2]$-complete. Can we show every $T(k)$ approximation is also $W[2]$-hard?

- Parameterized Clique is $W[1]$-complete. Can we show every $T(k)$ approximation is also $W[1]$-hard?
Important Open Questions

- Parameterized Dominating Set is $W[2]$-complete. Can we show every $T(k)$ approximation is also $W[2]$-hard?

- Parameterized Clique is $W[1]$-complete. Can we show every $T(k)$ approximation is also $W[1]$-hard? Can we show 1.01 approximation is $W[1]$-hard?
Are there natural problems in PSP which do not have efficient MA protocols?
Are there natural problems in PSP which do not have efficient MA protocols?

Conceptually/Philosophically can we say something about the various time hypotheses?
THANK YOU!
The Framework

W[1] ≠ FPT

ETH

SETH

k-Sum Hyp.

PSP

PSP(MULTEQ)

PSP(DISJ)

PSP(SUMZERO)

MaxCover

Reduction from [CCKLMNT17]

k-DomSet