Quantum Fine-Grained Complexity

Conditional Lower Bounds Beyond P vs. NP

Jinge Bao

School of Informatics
University of Edinburgh

CDT QI Spring School

May 29, 2026

Plan for the lecture

1. Classical Fine-Grained Complexity
2. Quantum Algorithmic Tools
3. QSETH and Conditional Quantum Lower Bounds
4. Frontiers and Open Problems
5. Wrap-up

Goal. Understand why certain polynomial-time problems resist faster algorithms.

- **Classical FGC** — SETH, OV, and the reduction web.
- **Quantum tools** — Grover, amplitude amplification, quantum minimum finding.
- **QSETH** — the right quantum hypothesis and conditional lower bounds.
- **Frontiers** — open problems and recent directions.

Why fine-grained complexity? And what is P like?

P vs. NP separates polynomial from super-polynomial. (coarse-grained complexity)

Why fine-grained complexity? And what is P like?

P vs. NP separates polynomial from super-polynomial. (coarse-grained complexity)

But *within* P:

- Edit distance: textbook $O(n^2)$ since Wagner–Fischer (1974). Best: $n^2/2^{\Theta(\sqrt{\log n})}$ (Masek–Paterson).

Why fine-grained complexity? And what is P like?

P vs. NP separates polynomial from super-polynomial. (coarse-grained complexity)

But *within* P:

- Edit distance: textbook $O(n^2)$ since Wagner–Fischer (1974). Best: $n^2/2^{\Theta(\sqrt{\log n})}$ (Masek–Paterson).
- All-Pairs Shortest Paths: $O(n^3)$ since Floyd–Warshall (1962). Best: $n^3/2^{\Theta(\sqrt{\log n})}$ (Williams '14).

Why fine-grained complexity? And what is P like?

P vs. NP separates polynomial from super-polynomial. (coarse-grained complexity)

But *within* P:

- Edit distance: textbook $O(n^2)$ since Wagner–Fischer (1974). Best: $n^2/2^{\Theta(\sqrt{\log n})}$ (Masek–Paterson).
- All-Pairs Shortest Paths: $O(n^3)$ since Floyd–Warshall (1962). Best: $n^3/2^{\Theta(\sqrt{\log n})}$ (Williams '14).
- 3SUM: $O(n^2)$. Best: $n^2/\operatorname{polylog} n$ (Baran–Demaine–Pătraşcu '08).

Why fine-grained complexity? And what is P like?

P vs. NP separates polynomial from super-polynomial. (coarse-grained complexity)

But *within* P:

- Edit distance: textbook $O(n^2)$ since Wagner–Fischer (1974). Best: $n^2/2^{\Theta(\sqrt{\log n})}$ (Masek–Paterson).
- All-Pairs Shortest Paths: $O(n^3)$ since Floyd–Warshall (1962). Best: $n^3/2^{\Theta(\sqrt{\log n})}$ (Williams '14).
- 3SUM: $O(n^2)$. Best: $n^2/\operatorname{polylog} n$ (Baran–Demaine–Pătraşcu '08).

After $\sim$50 years, no truly polynomial speedup for any of these. *Why?*

Why fine-grained complexity? And what is P like?

P vs. NP separates polynomial from super-polynomial. (coarse-grained complexity)

But *within* P:

- Edit distance: textbook $O(n^2)$ since Wagner–Fischer (1974). Best: $n^2/2^{\Theta(\sqrt{\log n})}$ (Masek–Paterson).
- All-Pairs Shortest Paths: $O(n^3)$ since Floyd–Warshall (1962). Best: $n^3/2^{\Theta(\sqrt{\log n})}$ (Williams '14).
- 3SUM: $O(n^2)$. Best: $n^2/\operatorname{polylog} n$ (Baran–Demaine–Pătraşcu '08).

After $\sim$50 years, no truly polynomial speedup for any of these. *Why?*

Fine-grained complexity.

1. Pick a small list of well-believed "hard" problems and conjecture their best-known exponents are tight.

Why fine-grained complexity? And what is P like?

P vs. NP separates polynomial from super-polynomial. (coarse-grained complexity)

But *within* P:

- Edit distance: textbook $O(n^2)$ since Wagner–Fischer (1974). Best: $n^2/2^{\Theta(\sqrt{\log n})}$ (Masek–Paterson).
- All-Pairs Shortest Paths: $O(n^3)$ since Floyd–Warshall (1962). Best: $n^3/2^{\Theta(\sqrt{\log n})}$ (Williams '14).
- 3SUM: $O(n^2)$. Best: $n^2/\operatorname{polylog} n$ (Baran–Demaine–Pătraşcu '08).

After $\sim$50 years, no truly polynomial speedup for any of these. *Why?*

Fine-grained complexity.

1. Pick a small list of well-believed "hard" problems and conjecture their best-known exponents are tight.
2. *Reduce* between them, with tight quantitative bookkeeping, to explain everything else.

Why fine-grained complexity? And what is P like?

P vs. NP separates polynomial from super-polynomial. (coarse-grained complexity)

But *within* P:

- Edit distance: textbook $O(n^2)$ since Wagner–Fischer (1974). Best: $n^2/2^{\Theta(\sqrt{\log n})}$ (Masek–Paterson).
- All-Pairs Shortest Paths: $O(n^3)$ since Floyd–Warshall (1962). Best: $n^3/2^{\Theta(\sqrt{\log n})}$ (Williams '14).
- 3SUM: $O(n^2)$. Best: $n^2/\operatorname{polylog} n$ (Baran–Demaine–Pătraşcu '08).

After $\sim$50 years, no truly polynomial speedup for any of these. *Why?*

Fine-grained complexity.

1. Pick a small list of well-believed "hard" problems and conjecture their best-known exponents are tight.
2. *Reduce* between them, with tight quantitative bookkeeping, to explain everything else.
3. Get *conditional* lower bounds — the only realistic option, since unconditional ones imply $\mathrm{P} \neq \mathrm{NP}$ or worse.

The three pillars

Problem	Best known	Conjecture	Implies hardness of ...
k-SAT	$2^{n-n/O(k)}$	SETH	OV, edit dist., LCS, Fréchet
3SUM	$n^2/\operatorname{polylog} n$	3SUM-Hyp	geometric / incidence problems
APSP	$n^3/2^{\sqrt{\log n}}$	APSP-Hyp	radius, betweenness, neg. triangle

The three pillars

Problem	Best known	Conjecture	Implies hardness of ...
k-SAT	$2^{n-n/O(k)}$	SETH	OV, edit dist., LCS, Fréchet
3SUM	$n^2/\operatorname{polylog} n$	3SUM-Hyp	geometric / incidence problems
APSP	$n^3/2^{\sqrt{\log n}}$	APSP-Hyp	radius, betweenness, neg. triangle

Why these three?

- Each is the "natural" bottleneck for an entire family of algorithms.

The three pillars

Problem	Best known	Conjecture	Implies hardness of ...
k-SAT	$2^{n-n/O(k)}$	SETH	OV, edit dist., LCS, Fréchet
3SUM	$n^2/\operatorname{polylog} n$	3SUM-Hyp	geometric / incidence problems
APSP	$n^3/2^{\sqrt{\log n}}$	APSP-Hyp	radius, betweenness, neg. triangle

Why these three?

- Each is the "natural" bottleneck for an entire family of algorithms.
- Those conjectures can not imply each other.

The three pillars

Problem	Best known	Conjecture	Implies hardness of ...
k-SAT	$2^{n-n/O(k)}$	SETH	OV, edit dist., LCS, Fréchet
3SUM	$n^2/\operatorname{polylog} n$	3SUM-Hyp	geometric / incidence problems
APSP	$n^3/2^{\sqrt{\log n}}$	APSP-Hyp	radius, betweenness, neg. triangle

Why these three?

- Each is the "natural" bottleneck for an entire family of algorithms.
- Those conjectures can not imply each other.
- Together they explain the hardness of > 100 problems studied in the FGC literature.

The k-SAT problem

Input. A Boolean formula in *conjunctive normal form* (CNF) with n variables $x_1, \ldots, x_n$ and m clauses, each clause containing at most k literals.

The k-SAT problem

Input. A Boolean formula in *conjunctive normal form* (CNF) with n variables $x_1, \ldots, x_n$ and m clauses, each clause containing at most k literals.

Example (3-SAT).

$$\varphi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4).$$

The k-SAT problem

Input. A Boolean formula in *conjunctive normal form* (CNF) with n variables $x_1, \ldots, x_n$ and m clauses, each clause containing at most k literals.

Example (3-SAT).

$$\varphi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4).$$

Question. Is there an assignment $x \in \{0,1\}^n$ that satisfies all clauses?

The k-SAT problem

Input. A Boolean formula in *conjunctive normal form* (CNF) with n variables $x_1, \ldots, x_n$ and m clauses, each clause containing at most k literals.

Example (3-SAT).

$$\varphi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4).$$

Question. Is there an assignment $x \in \{0,1\}^n$ that satisfies all clauses?

Complexity.

- k-SAT is NP-complete for $k \geq 3$ (Cook–Levin).

The k-SAT problem

Input. A Boolean formula in *conjunctive normal form* (CNF) with n variables $x_1, \ldots, x_n$ and m clauses, each clause containing at most k literals.

Example (3-SAT).

$$\varphi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4).$$

Question. Is there an assignment $x \in \{0,1\}^n$ that satisfies all clauses?

Complexity.

- k-SAT is NP-complete for $k \geq 3$ (Cook–Levin).
- Brute force: try all 2^n assignments $\Rightarrow O(2^n \cdot m)$ time.

The k-SAT problem

Input. A Boolean formula in *conjunctive normal form* (CNF) with n variables $x_1, \ldots, x_n$ and m clauses, each clause containing at most k literals.

Example (3-SAT).

$$\varphi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4).$$

Question. Is there an assignment $x \in \{0,1\}^n$ that satisfies all clauses?

Complexity.

- k-SAT is NP-complete for $k \geq 3$ (Cook–Levin).
- Brute force: try all 2^n assignments $\Rightarrow O(2^n \cdot m)$ time.
- Best algorithms: $2^{n(1-c/k)}$ for a constant c (Schöning '99, PPSZ '05).

The k-SAT problem

Input. A Boolean formula in *conjunctive normal form* (CNF) with n variables $x_1, \ldots, x_n$ and m clauses, each clause containing at most k literals.

Example (3-SAT).

$$\varphi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee \bar{x}_4).$$

Question. Is there an assignment $x \in \{0,1\}^n$ that satisfies all clauses?

Complexity.

- k-SAT is NP-complete for $k \geq 3$ (Cook–Levin).
- Brute force: try all 2^n assignments $\Rightarrow O(2^n \cdot m)$ time.
- Best algorithms: $2^{n(1-c/k)}$ for a constant c (Schöning '99, PPSZ '05).
- **Key question:** Can we do $2^{(1-\epsilon)n}$ for a *fixed* $\epsilon > 0$ independent of k?

SETH: Strong Exponential Time Hypothesis

Impagliazzo–Paturi '01.

$$\text{SETH} \ : \ \forall \epsilon > 0, \ \exists k \text{ s.t. } k\text{-SAT requires time } 2^{(1-\epsilon)n}.$$

SETH: Strong Exponential Time Hypothesis

Impagliazzo–Paturi '01.

$$\text{SETH} \ : \ \forall \epsilon > 0, \ \exists k \text{ s.t. } k\text{-SAT requires time } 2^{(1-\epsilon)n}.$$

Equivalently: $\mathrm{CNF\text{-}SAT}$ on n variables, m clauses, cannot be solved in $2^{(1-\epsilon)n}\,\mathrm{poly}(m)$ for any $\epsilon > 0$.

SETH: Strong Exponential Time Hypothesis

Impagliazzo–Paturi '01.

$$\text{SETH} \ : \ \forall \epsilon > 0, \ \exists k \text{ s.t. } k\text{-SAT requires time } 2^{(1-\epsilon)n}.$$

Equivalently: $\mathrm{CNF\text{-}SAT}$ on n variables, m clauses, cannot be solved in $2^{(1-\epsilon)n}\,\mathrm{poly}(m)$ for any $\epsilon > 0$.

Status.

- Best algorithm (Schöning, PPSZ, CDLMNPSW): $2^{n(1-1/O(k))}$.

SETH: Strong Exponential Time Hypothesis

Impagliazzo–Paturi '01.

$$\text{SETH} \ : \ \forall \epsilon > 0, \ \exists k \text{ s.t. } k\text{-SAT requires time } 2^{(1-\epsilon)n}.$$

Equivalently: $\mathrm{CNF\text{-}SAT}$ on n variables, m clauses, cannot be solved in $2^{(1-\epsilon)n}\,\mathrm{poly}(m)$ for any $\epsilon > 0$.

Status.

- Best algorithm (Schöning, PPSZ, CDLMNPSW): $2^{n(1-1/O(k))}$.
- Not implied by $\mathrm{P} \neq \mathrm{NP}$.

SETH: Strong Exponential Time Hypothesis

Impagliazzo–Paturi '01.

$$\text{SETH} \ : \ \forall \epsilon > 0, \ \exists k \text{ s.t. } k\text{-SAT requires time } 2^{(1-\epsilon)n}.$$

Equivalently: $\mathrm{CNF\text{-}SAT}$ on n variables, m clauses, cannot be solved in $2^{(1-\epsilon)n}\,\mathrm{poly}(m)$ for any $\epsilon > 0$.

Status.

- Best algorithm (Schöning, PPSZ, CDLMNPSW): $2^{n(1-1/O(k))}$.
- Not implied by $\mathrm{P} \neq \mathrm{NP}$.
- "Tighter" than (Exponential Time Hypothesis): ETH only says $2^{\Omega(n)}$. SETH $\implies$ ETH.

SETH: Strong Exponential Time Hypothesis

Impagliazzo–Paturi '01.

$$\text{SETH} \ : \ \forall \epsilon > 0, \ \exists k \text{ s.t. } k\text{-SAT requires time } 2^{(1-\epsilon)n}.$$

Equivalently: $\mathrm{CNF\text{-}SAT}$ on n variables, m clauses, cannot be solved in $2^{(1-\epsilon)n}\,\mathrm{poly}(m)$ for any $\epsilon > 0$.

Status.

- Best algorithm (Schöning, PPSZ, CDLMNPSW): $2^{n(1-1/O(k))}$.
- Not implied by $\mathrm{P} \neq \mathrm{NP}$.
- "Tighter" than (Exponential Time Hypothesis): ETH only says $2^{\Omega(n)}$. SETH $\implies$ ETH.

Use. The *strongest* fine-grained hypothesis. Many conditional lower bounds depend on it.

Orthogonal Vectors (OV)

Problem. Given $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$ and $d = \omega(\log n)$, is there a pair $(a, b) \in A \times B$ with

$$\langle a, b \rangle \;=\; \sum_{i=1}^{d} a_i b_i \;=\; 0?$$

Orthogonal Vectors (OV)

Problem. Given $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$ and $d = \omega(\log n)$, is there a pair $(a, b) \in A \times B$ with

$$\langle a, b \rangle \;=\; \sum_{i=1}^{d} a_i b_i \;=\; 0?$$

Trivial. $O(n^2 d)$ by checking all pairs.

Orthogonal Vectors (OV)

Problem. Given $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$ and $d = \omega(\log n)$, is there a pair $(a, b) \in A \times B$ with

$$\langle a, b \rangle \;=\; \sum_{i=1}^{d} a_i b_i \;=\; 0?$$

Trivial. $O(n^2 d)$ by checking all pairs.

Conjecture (OVH). For every $\epsilon > 0$, OV requires $n^{2-\epsilon}$ time (even for $d = \omega(\log n)$).

Orthogonal Vectors (OV)

Problem. Given $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$ and $d = \omega(\log n)$, is there a pair $(a, b) \in A \times B$ with

$$\langle a, b \rangle \;=\; \sum_{i=1}^{d} a_i b_i \;=\; 0?$$

Trivial. $O(n^2 d)$ by checking all pairs.

Conjecture (OVH). For every $\epsilon > 0$, OV requires $n^{2-\epsilon}$ time (even for $d = \omega(\log n)$).

Theorem (Williams '05). SETH $\Rightarrow$ OVH.

Orthogonal Vectors (OV)

Problem. Given $A, B \subseteq \{0,1\}^d$ with $|A| = |B| = n$ and $d = \omega(\log n)$, is there a pair $(a, b) \in A \times B$ with

$$\langle a, b \rangle \;=\; \sum_{i=1}^{d} a_i b_i \;=\; 0?$$

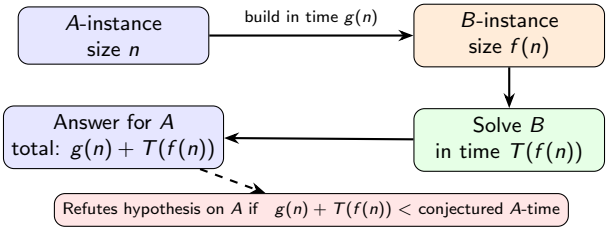
Trivial. $O(n^2 d)$ by checking all pairs.

Conjecture (OVH). For every $\epsilon > 0$, OV requires $n^{2-\epsilon}$ time (even for $d = \omega(\log n)$).

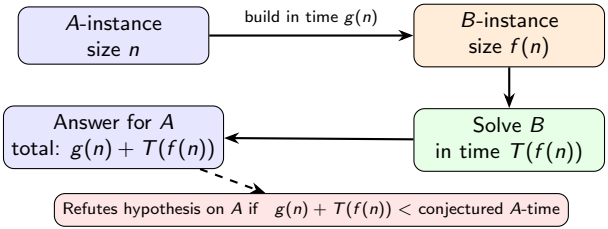
Theorem (Williams '05). SETH $\Rightarrow$ OVH.

This makes OV the "central station" of FGC: SETH lifts via OV to everything else.

Anatomy of a fine-grained reduction

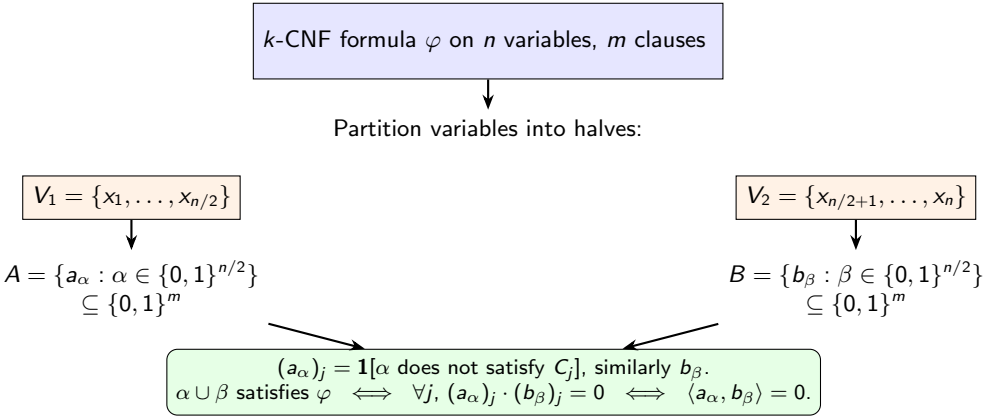


Anatomy of a fine-grained reduction

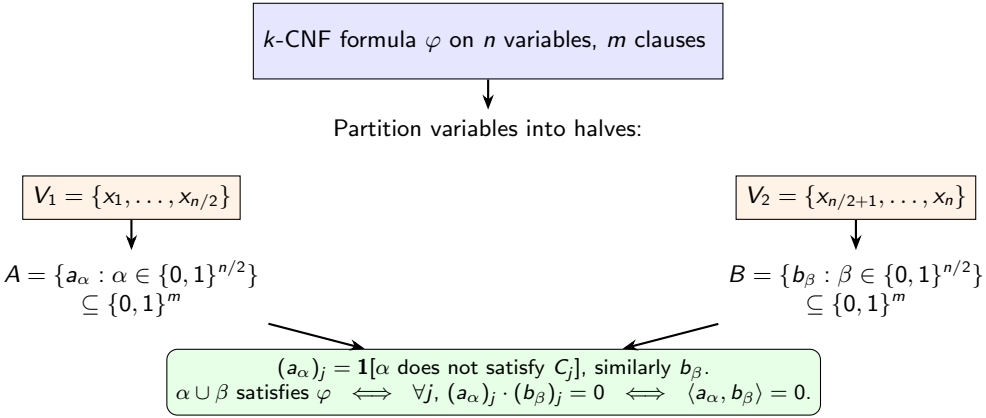


Fragility. Lose too much in the reduction (e.g. $f(n) = n^{1+\delta}$), and the bound becomes trivial. Fine-grained reductions must be *tight*.

The SETH $\Rightarrow$ OV reduction (Williams '05)

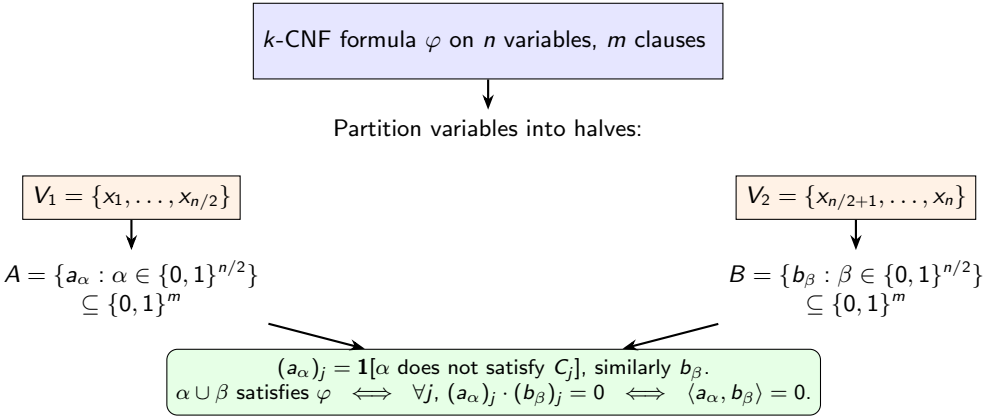


The SETH $\Rightarrow$ OV reduction (Williams '05)



$N := 2^{n/2}$ vectors of dimension $d := m = \text{poly}(n)$.

The SETH $\Rightarrow$ OV reduction (Williams '05)



$N := 2^{n/2}$ vectors of dimension $d := m = \text{poly}(n)$.

If OV solvable in $N^{2-\epsilon}\,\text{poly}(d)$, then k-SAT in $2^{(1-\epsilon/2)n}\,\text{poly}(n)$ — refuting SETH.

Why this reduction is “tight”

Three numerical facts make the reduction work:

Why this reduction is "tight"

Three numerical facts make the reduction work:

- **Square-root in instance size.** $N = 2^{n/2}$ vectors — splits the exponent neatly. $N^2 = 2^n$ matches brute-force SAT.

Why this reduction is "tight"

Three numerical facts make the reduction work:

- **Square-root in instance size.** $N = 2^{n/2}$ vectors — splits the exponent neatly. $N^2 = 2^n$ matches brute-force SAT.
- **Polynomial dimension.** $d = m = \text{poly}(n)$ stays polylogarithmic in N, exactly the regime where OVH is conjectured.

Why this reduction is "tight"

Three numerical facts make the reduction work:

- **Square-root in instance size.** $N = 2^{n/2}$ vectors — splits the exponent neatly. $N^2 = 2^n$ matches brute-force SAT.
- **Polynomial dimension.** $d = m = \text{poly}(n)$ stays polylogarithmic in N, exactly the regime where OVH is conjectured.
- **Linear-time vector construction.** Each vector is built in $\text{poly}(n)$ time from a partial assignment.

Why this reduction is "tight"

Three numerical facts make the reduction work:

- **Square-root in instance size.** $N = 2^{n/2}$ vectors — splits the exponent neatly. $N^2 = 2^n$ matches brute-force SAT.
- **Polynomial dimension.** $d = m = \text{poly}(n)$ stays polylogarithmic in N, exactly the regime where OVH is conjectured.
- **Linear-time vector construction.** Each vector is built in $\text{poly}(n)$ time from a partial assignment.

Consequence.

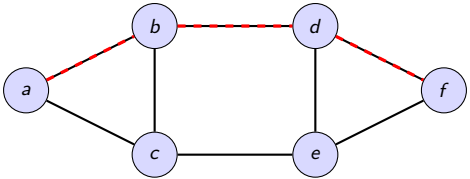
$$\text{OV in } n^{2-\epsilon} \cdot \text{poly}(d) \implies k\text{-SAT in } 2^{(1-\epsilon/2)\,n}.$$

The diameter of a graph

Definition. The *diameter* of a graph G is the longest shortest path between any two vertices:

$$\mathrm{diam}(G) = \max_{u,v \in V} \mathrm{dist}(u, v).$$

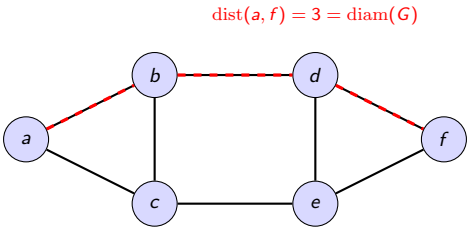
$\mathrm{dist}(a, f) = 3 = \mathrm{diam}(G)$



The diameter of a graph

Definition. The *diameter* of a graph G is the longest shortest path between any two vertices:

$$\mathrm{diam}(G) = \max_{u,v \in V} \mathrm{dist}(u, v).$$

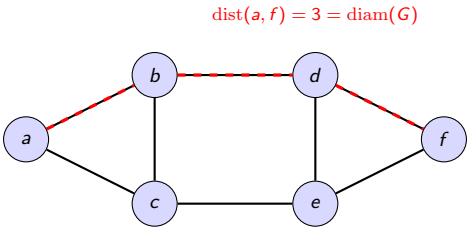


- **Trivial algorithm.** Run BFS/Dijkstra from every vertex: $O(n \cdot (n + m))$ for unweighted graphs, $O(n^2)$ for sparse, $O(n^3)$ for dense.

The diameter of a graph

Definition. The *diameter* of a graph G is the longest shortest path between any two vertices:

$$\mathrm{diam}(G) = \max_{u,v \in V} \mathrm{dist}(u, v).$$

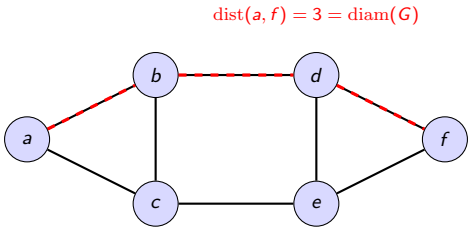


- **Trivial algorithm.** Run BFS/Dijkstra from every vertex: $O(n \cdot (n + m))$ for unweighted graphs, $O(n^2)$ for sparse, $O(n^3)$ for dense.
- **Key question.** Can we compute the diameter in truly sub-n^2 time (sparse) or sub-n^3 time (dense)?

The diameter of a graph

Definition. The *diameter* of a graph G is the longest shortest path between any two vertices:

$$\mathrm{diam}(G) = \max_{u,v \in V} \mathrm{dist}(u, v).$$



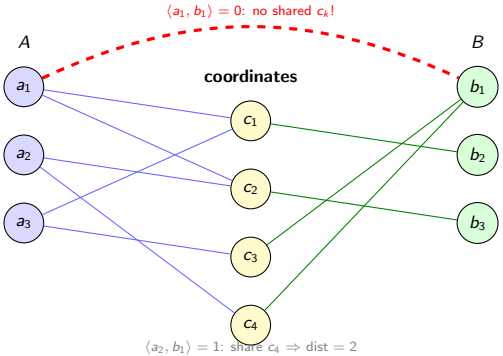
- **Trivial algorithm.** Run BFS/Dijkstra from every vertex: $O(n \cdot (n + m))$ for unweighted graphs, $O(n^2)$ for sparse, $O(n^3)$ for dense.
- **Key question.** Can we compute the diameter in truly sub-n^2 time (sparse) or sub-n^3 time (dense)?
- Fine-grained reductions show this is *unlikely* — diameter inherits hardness from OV.

The OV $\Rightarrow$ graph diameter reduction (Roditty–V.Williams '13)

Given. OV instance: sets $A, B \subseteq \{0,1\}^d$, $|A| = |B| = n$.

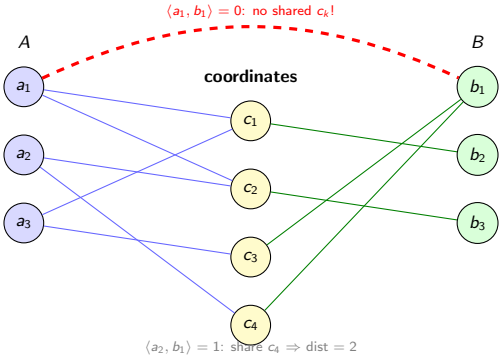
The OV $\Rightarrow$ graph diameter reduction (Roditty–V.Williams '13)

Given. OV instance: sets $A, B \subseteq \{0,1\}^d$, $|A| = |B| = n$.
Build. A tripartite graph:



The OV $\Rightarrow$ graph diameter reduction (Roditty–V.Williams '13)

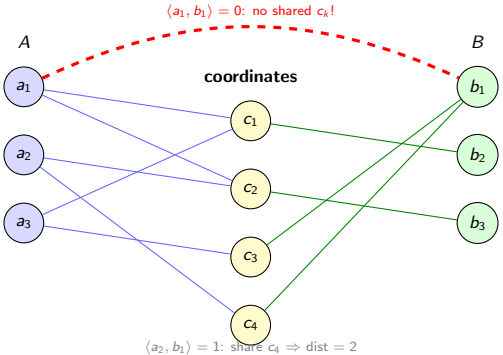
Given. OV instance: sets $A, B \subseteq \{0,1\}^d$, $|A| = |B| = n$.
Build. A tripartite graph:



- a_i—c_k if $(a_i)_k = 1$; $\quad b_j$—c_k if $(b_j)_k = 1$.

The OV $\Rightarrow$ graph diameter reduction (Roditty–V.Williams '13)

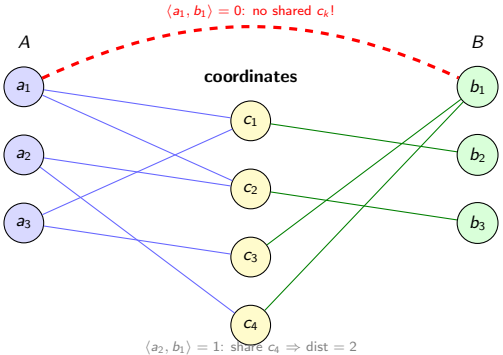
Given. OV instance: sets $A, B \subseteq \{0,1\}^d$, $|A| = |B| = n$.
Build. A tripartite graph:



- a_i—c_k if $(a_i)_k = 1$; $\quad b_j$—c_k if $(b_j)_k = 1$.
- $\langle a_i, b_j \rangle > 0 \Rightarrow$ share a neighbor $\Rightarrow \mathrm{dist}(a_i, b_j) = 2$.

The OV $\Rightarrow$ graph diameter reduction (Roditty–V.Williams '13)

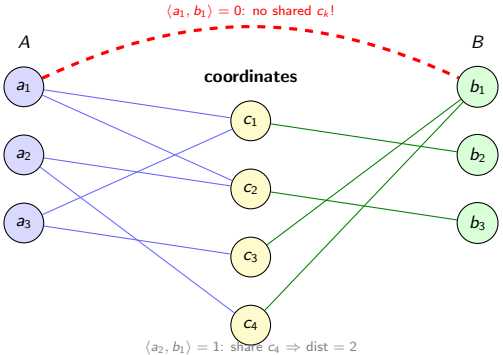
Given. OV instance: sets $A, B \subseteq \{0,1\}^d$, $|A| = |B| = n$.
Build. A tripartite graph:



- a_i—c_k if $(a_i)_k = 1$; $\quad$ b_j—c_k if $(b_j)_k = 1$.
- $\langle a_i, b_j \rangle > 0 \Rightarrow$ share a neighbor $\Rightarrow \mathrm{dist}(a_i, b_j) = 2$.
- $\langle a_i, b_j \rangle = 0 \Rightarrow$ no shared neighbor $\Rightarrow \mathrm{dist}(a_i, b_j) \geq 4$.

The OV $\Rightarrow$ graph diameter reduction (Roditty–V.Williams '13)

Given. OV instance: sets $A, B \subseteq \{0,1\}^d$, $|A| = |B| = n$.
Build. A tripartite graph:



- a_i—c_k if $(a_i)_k = 1$; $\quad$ b_j—c_k if $(b_j)_k = 1$.
- $\langle a_i, b_j \rangle > 0 \Rightarrow$ share a neighbor $\Rightarrow \mathrm{dist}(a_i, b_j) = 2$.
- $\langle a_i, b_j \rangle = 0 \Rightarrow$ no shared neighbor $\Rightarrow \mathrm{dist}(a_i, b_j) \geq 4$.

Example: OV $\Rightarrow$ graph diameter (cont.)

Conclusion.

$$\exists \text{ orthogonal pair} \iff \text{diameter} \geq 4.$$

Example: OV $\Rightarrow$ graph diameter (cont.)

Conclusion.

$$\exists \text{ orthogonal pair} \iff \text{diameter} \geq 4.$$

Reduction analysis.

Example: OV $\Rightarrow$ graph diameter (cont.)

Conclusion.

$$\exists \text{ orthogonal pair} \iff \text{diameter} \geq 4.$$

Reduction analysis.

- Graph has $2n + d = O(n)$ vertices (since $d = \text{poly}(\log n)$).

Example: OV $\Rightarrow$ graph diameter (cont.)

Conclusion.

$$\exists \text{ orthogonal pair} \iff \text{diameter} \geq 4.$$

Reduction analysis.

- Graph has $2n + d = O(n)$ vertices (since $d = \text{poly}(\log n)$).
- Construction time: $g(n) = O(nd)$ (linear in input size).

Example: OV $\Rightarrow$ graph diameter (cont.)

Conclusion.

$$\exists \text{ orthogonal pair} \iff \text{diameter} \geq 4.$$

Reduction analysis.

- Graph has $2n + d = O(n)$ vertices (since $d = \text{poly}(\log n)$).
- Construction time: $g(n) = O(nd)$ (linear in input size).
- Instance size preserved: $f(n) = O(n)$.

Example: OV $\Rightarrow$ graph diameter (cont.)

Conclusion.

$$\exists \text{ orthogonal pair} \iff \text{diameter} \geq 4.$$

Reduction analysis.

- Graph has $2n + d = O(n)$ vertices (since $d = \text{poly}(\log n)$).
- Construction time: $g(n) = O(nd)$ (linear in input size).
- Instance size preserved: $f(n) = O(n)$.

If diameter in $O(n^{2-\epsilon})$ time, then OV in $O(n^{2-\epsilon})$, refuting OVH (and hence SETH).

Example: OV $\Rightarrow$ graph diameter (cont.)

Conclusion.

$$\exists \text{ orthogonal pair} \iff \text{diameter} \geq 4.$$

Reduction analysis.

- Graph has $2n + d = O(n)$ vertices (since $d = \text{poly}(\log n)$).
- Construction time: $g(n) = O(nd)$ (linear in input size).
- Instance size preserved: $f(n) = O(n)$.

If diameter in $O(n^{2-\epsilon})$ time, then OV in $O(n^{2-\epsilon})$, refuting OVH (and hence SETH).

A purely *graph-theoretic* problem (diameter) inherits the hardness of a *vector* problem (OV), via a simple gadget graph. This is the power of fine-grained reductions.

3SUM and APSP

3SUM. Given n integers, decide if any three sum to zero. Best: $n^2/\operatorname{polylog} n$.

3SUM and APSP

3SUM. Given n integers, decide if any three sum to zero. Best: $n^2/\operatorname{polylog} n$.

3SUM-Hyp. For every $\epsilon > 0$, 3SUM requires $n^{2-\epsilon}$.

3SUM and APSP

3SUM. Given n integers, decide if any three sum to zero. Best: $n^2/\operatorname{polylog} n$.

3SUM-Hyp. For every $\epsilon > 0$, 3SUM requires $n^{2-\epsilon}$.

Consequences: most computational geometric problems: point-line incidences, dynamic connectivity, conflict-free coloring.

3SUM and APSP

3SUM. Given n integers, decide if any three sum to zero. Best: $n^2/\operatorname{polylog} n$.

3SUM-Hyp. For every $\epsilon > 0$, 3SUM requires $n^{2-\epsilon}$.

Consequences: most computational geometric problems: point-line incidences, dynamic connectivity, conflict-free coloring.

APSP. All-pairs shortest paths in a weighted graph. Best: $n^3/2^{\sqrt{\log n}}$.

3SUM and APSP

3SUM. Given n integers, decide if any three sum to zero. Best: $n^2/\operatorname{polylog} n$.

3SUM-Hyp. For every $\epsilon > 0$, 3SUM requires $n^{2-\epsilon}$.

Consequences: most computational geometric problems: point-line incidences, dynamic connectivity, conflict-free coloring.

APSP. All-pairs shortest paths in a weighted graph. Best: $n^3/2^{\sqrt{\log n}}$.

APSP-Hyp. For every $\epsilon > 0$, APSP requires $n^{3-\epsilon}$.

3SUM and APSP

3SUM. Given n integers, decide if any three sum to zero. Best: $n^2/\operatorname{polylog} n$.

3SUM-Hyp. For every $\epsilon > 0$, 3SUM requires $n^{2-\epsilon}$.

Consequences: most computational geometric problems: point-line incidences, dynamic connectivity, conflict-free coloring.

APSP. All-pairs shortest paths in a weighted graph. Best: $n^3/2^{\sqrt{\log n}}$.

APSP-Hyp. For every $\epsilon > 0$, APSP requires $n^{3-\epsilon}$.

Consequences: most graph problems: negative triangle detection/counting, graph radius/diameter, betweenness centrality, second shortest path, replacement paths.

3SUM and APSP

3SUM. Given n integers, decide if any three sum to zero. Best: $n^2/\operatorname{polylog} n$.

3SUM-Hyp. For every $\epsilon > 0$, 3SUM requires $n^{2-\epsilon}$.

Consequences: most computational geometric problems: point-line incidences, dynamic connectivity, conflict-free coloring.

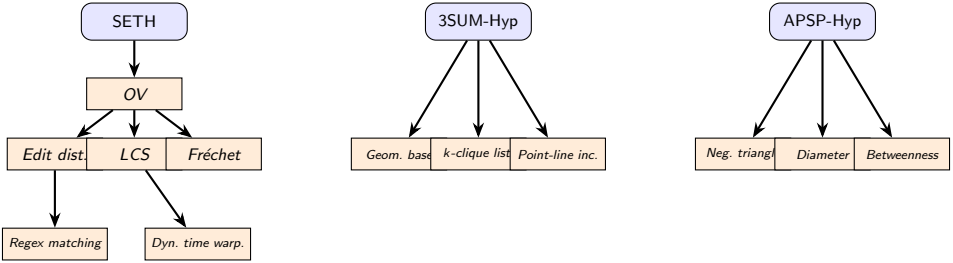
APSP. All-pairs shortest paths in a weighted graph. Best: $n^3/2^{\sqrt{\log n}}$.

APSP-Hyp. For every $\epsilon > 0$, APSP requires $n^{3-\epsilon}$.

Consequences: most graph problems: negative triangle detection/counting, graph radius/diameter, betweenness centrality, second shortest path, replacement paths.

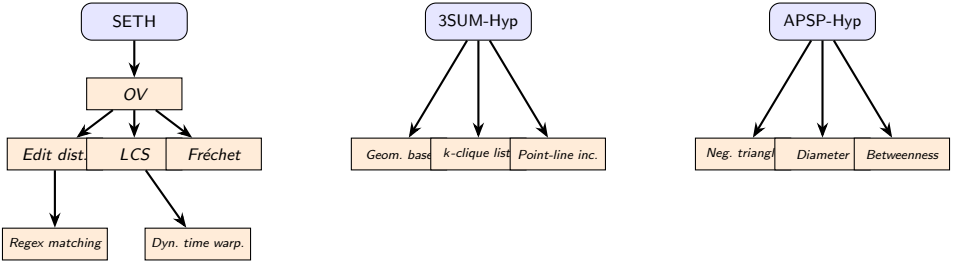
Independent. SETH, 3SUM-Hyp and APSP-Hyp are indepdent by so far.

The classical fine-grained landscape



Arrows point from *hard problem* to *problem it makes hard*. Each arrow is a tight reduction.

The classical fine-grained landscape



Arrows point from *hard problem* to *problem it makes hard*. Each arrow is a tight reduction.

On some fine-grained questions in algorithms and complexity

Virginia Vassilevska Williams
Massachusetts Institute of Technology, EECS and CSAIL

What changes in the quantum world?

Quantum speedups for the three pillars:

Pillar	Classical	Quantum	How
OV (SETH)	$n^2 \operatorname{poly}(d)$	$n \cdot \operatorname{poly}(d)$	nested Grover
3SUM	$n^2 / \operatorname{polylog} n$	$\tilde{O}(n)$	sort + Grover
APSP	n^3	$\tilde{O}(n^{5/2})$	quantum min-finding

What changes in the quantum world?

Quantum speedups for the three pillars:

Pillar	Classical	Quantum	How
OV (SETH)	$n^2 \operatorname{poly}(d)$	$n \cdot \operatorname{poly}(d)$	nested Grover
3SUM	$n^2 / \operatorname{polylog} n$	$\tilde{O}(n)$	sort + Grover
APSP	n^3	$\tilde{O}(n^{5/2})$	quantum min-finding

Pattern. Roughly a $\sqrt{\cdot}$ speedup from Grover / quantum minimum finding.

What changes in the quantum world?

Quantum speedups for the three pillars:

Pillar	Classical	Quantum	How
OV (SETH)	$n^2 \operatorname{poly}(d)$	$n \cdot \operatorname{poly}(d)$	nested Grover
3SUM	$n^2 / \operatorname{polylog} n$	$\tilde{O}(n)$	sort + Grover
APSP	n^3	$\tilde{O}(n^{5/2})$	quantum min-finding

Pattern. Roughly a $\sqrt{\cdot}$ speedup from Grover / quantum minimum finding.

Two parallel questions.

- *Algorithms.* How much can quantum gain? Are these optimal?

What changes in the quantum world?

Quantum speedups for the three pillars:

Pillar	Classical	Quantum	How
OV (SETH)	$n^2 \operatorname{poly}(d)$	$n \cdot \operatorname{poly}(d)$	nested Grover
3SUM	$n^2 / \operatorname{polylog} n$	$\tilde{O}(n)$	sort + Grover
APSP	n^3	$\tilde{O}(n^{5/2})$	quantum min-finding

Pattern. Roughly a $\sqrt{\cdot}$ speedup from Grover / quantum minimum finding.

Two parallel questions.

- *Algorithms.* How much can quantum gain? Are these optimal?
- *Lower bounds.* Can we formulate quantum analogues of SETH / 3SUM-Hyp / APSP-Hyp and prove these speedups are the best possible?

Quantum algorithmic subroutines

Grover ('96). Search $f : [N] \to \{0, 1\}$ for a marked i:

$$\widetilde{O}(\sqrt{N/|f^{-1}(1)|}) \text{ time vs. classical } \Theta(N/|f^{-1}(1)|).$$

Quantum algorithmic subroutines

Grover ('96). Search $f : [N] \to \{0, 1\}$ for a marked i:

$$\widetilde{O}(\sqrt{N/|f^{-1}(1)|}) \text{ time vs. classical } \Theta(N/|f^{-1}(1)|).$$

Amplitude amplification (BHMT '02). Given any quantum procedure $\mathcal{A}$ that succeeds with probability p, $O(1/\sqrt{p})$ uses of $\mathcal{A}$ and $\mathcal{A}^\dagger$ suffice.

Quantum algorithmic subroutines

Grover ('96). Search $f : [N] \to \{0, 1\}$ for a marked i:

$$\widetilde{O}(\sqrt{N/|f^{-1}(1)|}) \text{ time vs. classical } \Theta(N/|f^{-1}(1)|).$$

Amplitude amplification (BHMT '02). Given any quantum procedure $\mathcal{A}$ that succeeds with probability p, $O(1/\sqrt{p})$ uses of $\mathcal{A}$ and $\mathcal{A}^\dagger$ suffice.
Nest Grover: subroutine with time T and success prob $p \Rightarrow$ wrapped cost $T/\sqrt{p}$.

Quantum algorithmic subroutines

Grover ('96). Search $f : [N] \to \{0,1\}$ for a marked i:

$$\widetilde{O}(\sqrt{N/|f^{-1}(1)|}) \text{ time vs. classical } \Theta(N/|f^{-1}(1)|).$$

Amplitude amplification (BHMT '02). Given any quantum procedure $\mathcal{A}$ that succeeds with probability p, $O(1/\sqrt{p})$ uses of $\mathcal{A}$ and $\mathcal{A}^\dagger$ suffice.
Nest Grover: subroutine with time T and success prob $p \Rightarrow$ wrapped cost $T/\sqrt{p}$.

Quantum minimum finding (Dürr–Høyer '96). Given $f : [N] \to \mathbb{R}$, find $\arg\min_i f(i)$ in $\widetilde{O}(\sqrt{N})$ time.

Quantum algorithmic subroutines

Grover ('96). Search $f : [N] \to \{0,1\}$ for a marked i:

$$\widetilde{O}(\sqrt{N/|f^{-1}(1)|}) \text{ time vs. classical } \Theta(N/|f^{-1}(1)|).$$

Amplitude amplification (BHMT '02). Given any quantum procedure $\mathcal{A}$ that succeeds with probability p, $O(1/\sqrt{p})$ uses of $\mathcal{A}$ and $\mathcal{A}^\dagger$ suffice.
Nest Grover: subroutine with time T and success prob $p \Rightarrow$ wrapped cost $T/\sqrt{p}$.

Quantum minimum finding (Dürr–Høyer '96). Given $f : [N] \to \mathbb{R}$, find $\arg\min_i f(i)$ in $\widetilde{O}(\sqrt{N})$ time.
Idea.

Quantum algorithmic subroutines

Grover ('96). Search $f : [N] \to \{0,1\}$ for a marked i:

$$\widetilde{O}(\sqrt{N/|f^{-1}(1)|}) \text{ time vs. classical } \Theta(N/|f^{-1}(1)|).$$

Amplitude amplification (BHMT '02). Given any quantum procedure $\mathcal{A}$ that succeeds with probability p, $O(1/\sqrt{p})$ uses of $\mathcal{A}$ and $\mathcal{A}^\dagger$ suffice.
Nest Grover: subroutine with time T and success prob $p \Rightarrow$ wrapped cost $T/\sqrt{p}$.

Quantum minimum finding (Dürr–Høyer '96). Given $f : [N] \to \mathbb{R}$, find $\arg\min_i f(i)$ in $\widetilde{O}(\sqrt{N})$ time.
Idea.

- Pick a random threshold y.

Quantum algorithmic subroutines

Grover ('96). Search $f : [N] \to \{0,1\}$ for a marked i:

$$\widetilde{O}(\sqrt{N/|f^{-1}(1)|}) \text{ time vs. classical } \Theta(N/|f^{-1}(1)|).$$

Amplitude amplification (BHMT '02). Given any quantum procedure $\mathcal{A}$ that succeeds with probability p, $O(1/\sqrt{p})$ uses of $\mathcal{A}$ and $\mathcal{A}^\dagger$ suffice.
Nest Grover: subroutine with time T and success prob $p \Rightarrow$ wrapped cost $T/\sqrt{p}$.

Quantum minimum finding (Dürr–Høyer '96). Given $f : [N] \to \mathbb{R}$, find $\arg\min_i f(i)$ in $\widetilde{O}(\sqrt{N})$ time.
Idea.

- Pick a random threshold y.
- Grover-search for i with $f(i) < y$.

Quantum algorithmic subroutines

Grover ('96). Search $f : [N] \to \{0,1\}$ for a marked i:

$$\widetilde{O}(\sqrt{N/|f^{-1}(1)|}) \text{ time vs. classical } \Theta(N/|f^{-1}(1)|).$$

Amplitude amplification (BHMT '02). Given any quantum procedure $\mathcal{A}$ that succeeds with probability p, $O(1/\sqrt{p})$ uses of $\mathcal{A}$ and $\mathcal{A}^\dagger$ suffice.
Nest Grover: subroutine with time T and success prob $p \Rightarrow$ wrapped cost $T/\sqrt{p}$.

Quantum minimum finding (Dürr–Høyer '96). Given $f : [N] \to \mathbb{R}$, find $\arg\min_i f(i)$ in $\widetilde{O}(\sqrt{N})$ time.
Idea.

- Pick a random threshold y.
- Grover-search for i with $f(i) < y$.
- update y and repeat.

Quantum algorithmic subroutines

Grover ('96). Search $f : [N] \to \{0,1\}$ for a marked i:

$$\widetilde{O}(\sqrt{N/|f^{-1}(1)|}) \text{ time vs. classical } \Theta(N/|f^{-1}(1)|).$$

Amplitude amplification (BHMT '02). Given any quantum procedure $\mathcal{A}$ that succeeds with probability p, $O(1/\sqrt{p})$ uses of $\mathcal{A}$ and $\mathcal{A}^\dagger$ suffice.
Nest Grover: subroutine with time T and success prob $p \Rightarrow$ wrapped cost $T/\sqrt{p}$.

Quantum minimum finding (Dürr–Høyer '96). Given $f : [N] \to \mathbb{R}$, find $\arg\min_i f(i)$ in $\widetilde{O}(\sqrt{N})$ time.
Idea.

- Pick a random threshold y.
- Grover-search for i with $f(i) < y$.
- update y and repeat.

Quantum algorithmic subroutines

Grover ('96). Search $f : [N] \to \{0,1\}$ for a marked i:

$$\widetilde{O}(\sqrt{N/|f^{-1}(1)|}) \text{ time vs. classical } \Theta(N/|f^{-1}(1)|).$$

Amplitude amplification (BHMT '02). Given any quantum procedure $\mathcal{A}$ that succeeds with probability p, $O(1/\sqrt{p})$ uses of $\mathcal{A}$ and $\mathcal{A}^\dagger$ suffice.
Nest Grover: subroutine with time T and success prob $p \Rightarrow$ wrapped cost $T/\sqrt{p}$.

Quantum minimum finding (Dürr–Høyer '96). Given $f : [N] \to \mathbb{R}$, find $\arg\min_i f(i)$ in $\widetilde{O}(\sqrt{N})$ time.
Idea.

- Pick a random threshold y.
- Grover-search for i with $f(i) < y$.
- update y and repeat.

Why it matters for FGC. APSP uses min-finding ($\sqrt{n}$ per pair $\times$ n^2 pairs $= n^{5/2}$). This is the main quantum tool beyond Grover for fine-grained problems.

Quantum algorithms for the three pillars

OV. Nested Grover: $n^2\,\mathrm{poly}(d) \to n \cdot \mathrm{poly}(d)$. (Worked example next slide.)

Quantum algorithms for the three pillars

OV. Nested Grover: $n^2\,\mathrm{poly}(d) \to n \cdot \mathrm{poly}(d)$. (Worked example next slide.)

3SUM.

Quantum algorithms for the three pillars

OV. Nested Grover: $n^2\,\mathrm{poly}(d) \to n \cdot \mathrm{poly}(d)$. (Worked example next slide.)

3SUM.

- Sort the n numbers classically in $O(n \log n)$.

Quantum algorithms for the three pillars

OV. Nested Grover: $n^2\,\mathrm{poly}(d) \to n \cdot \mathrm{poly}(d)$. (Worked example next slide.)

3SUM.

- Sort the n numbers classically in $O(n \log n)$.
- For each a_i, Grover-search for a_j with $a_i + a_j = -a_k$ (binary search check): $\tilde{O}(\sqrt{n})$ per i, $\tilde{O}(n^{3/2})$ total.

Quantum algorithms for the three pillars

OV. Nested Grover: $n^2\,\mathrm{poly}(d) \to n \cdot \mathrm{poly}(d)$. (Worked example next slide.)

3SUM.

- Sort the n numbers classically in $O(n \log n)$.
- For each a_i, Grover-search for a_j with $a_i + a_j = -a_k$ (binary search check): $\tilde{O}(\sqrt{n})$ per i, $\tilde{O}(n^{3/2})$ total.
- With a tighter approach (hash + Grover): $\tilde{O}(n)$.

Quantum algorithms for the three pillars

OV. Nested Grover: $n^2\,\mathrm{poly}(d) \to n \cdot \mathrm{poly}(d)$. (Worked example next slide.)

3SUM.

- Sort the n numbers classically in $O(n \log n)$.
- For each a_i, Grover-search for a_j with $a_i + a_j = -a_k$ (binary search check): $\tilde{O}(\sqrt{n})$ per i, $\tilde{O}(n^{3/2})$ total.
- With a tighter approach (hash + Grover): $\tilde{O}(n)$.

APSP.

Quantum algorithms for the three pillars

OV. Nested Grover: $n^2\,\mathrm{poly}(d) \to n \cdot \mathrm{poly}(d)$. (Worked example next slide.)

3SUM.

- Sort the n numbers classically in $O(n \log n)$.
- For each a_i, Grover-search for a_j with $a_i + a_j = -a_k$ (binary search check): $\tilde{O}(\sqrt{n})$ per i, $\tilde{O}(n^{3/2})$ total.
- With a tighter approach (hash + Grover): $\tilde{O}(n)$.

APSP.

- For each source-target pair (i, k), find $\min_j(d(i,j) + d(j,k))$ via quantum minimum finding in $O(\sqrt{n})$.

Quantum algorithms for the three pillars

OV. Nested Grover: $n^2\,\mathrm{poly}(d) \to n \cdot \mathrm{poly}(d)$. (Worked example next slide.)

3SUM.

- Sort the n numbers classically in $O(n \log n)$.
- For each a_i, Grover-search for a_j with $a_i + a_j = -a_k$ (binary search check): $\tilde{O}(\sqrt{n})$ per i, $\tilde{O}(n^{3/2})$ total.
- With a tighter approach (hash + Grover): $\tilde{O}(n)$.

APSP.

- For each source-target pair (i, k), find $\min_j(d(i,j) + d(j,k))$ via quantum minimum finding in $O(\sqrt{n})$.
- Over all n^2 pairs: $\tilde{O}(n^{5/2})$.

Quantum algorithms for the three pillars

OV. Nested Grover: $n^2\,\mathrm{poly}(d) \to n \cdot \mathrm{poly}(d)$. (Worked example next slide.)

3SUM.

- Sort the n numbers classically in $O(n \log n)$.
- For each a_i, Grover-search for a_j with $a_i + a_j = -a_k$ (binary search check): $\tilde{O}(\sqrt{n})$ per i, $\tilde{O}(n^{3/2})$ total.
- With a tighter approach (hash + Grover): $\tilde{O}(n)$.

APSP.

- For each source-target pair (i, k), find $\min_j(d(i,j) + d(j,k))$ via quantum minimum finding in $O(\sqrt{n})$.
- Over all n^2 pairs: $\tilde{O}(n^{5/2})$.

Key question for each.

- OV: Is $n^{1-o(1)}$ a quantum barrier?
- 3SUM: Is $n^{1-o(1)}$ a quantum barrier?
- APSP: Is $n^{5/2-o(1)}$ a quantum barrier?

Quantum algorithms for the three pillars

OV. Nested Grover: $n^2\,\mathrm{poly}(d) \to n \cdot \mathrm{poly}(d)$. (Worked example next slide.)

3SUM.

- Sort the n numbers classically in $O(n \log n)$.
- For each a_i, Grover-search for a_j with $a_i + a_j = -a_k$ (binary search check): $\tilde{O}(\sqrt{n})$ per i, $\tilde{O}(n^{3/2})$ total.
- With a tighter approach (hash + Grover): $\tilde{O}(n)$.

APSP.

- For each source-target pair (i, k), find $\min_j(d(i,j) + d(j,k))$ via quantum minimum finding in $O(\sqrt{n})$.
- Over all n^2 pairs: $\tilde{O}(n^{5/2})$.

Key question for each.

- OV: Is $n^{1-o(1)}$ a quantum barrier?
- 3SUM: Is $n^{1-o(1)}$ a quantum barrier?
- APSP: Is $n^{5/2-o(1)}$ a quantum barrier?

$\Rightarrow$ Motivates quantum fine-grained hypotheses (QSETH, Q3SUM, QAPSP).

Quantum OV in $\tilde{O}(n \cdot d)$

Algorithm. Nested Grover, with amplitude amplification on each layer.

Quantum OV in $\tilde{O}(n \cdot d)$

Algorithm. Nested Grover, with amplitude amplification on each layer.

1. *Outer search* over $a \in A$ for an a that has an orthogonal partner in B. Each oracle call to the outer Grover invokes.

Quantum OV in $\tilde{O}(n \cdot d)$

Algorithm. Nested Grover, with amplitude amplification on each layer.

1. *Outer search* over $a \in A$ for an a that has an orthogonal partner in B. Each oracle call to the outer Grover invokes.
2. *Inner search* over $b \in B$ for $\langle a, b \rangle = 0$. Time per inner step: $O(d)$ to compute inner product. Inner cost: $\sqrt{n} \cdot d$.

Quantum OV in $\tilde{O}(n \cdot d)$

Algorithm. Nested Grover, with amplitude amplification on each layer.

1. *Outer search* over $a \in A$ for an a that has an orthogonal partner in B. Each oracle call to the outer Grover invokes.
2. *Inner search* over $b \in B$ for $\langle a, b \rangle = 0$. Time per inner step: $O(d)$ to compute inner product. Inner cost: $\sqrt{n} \cdot d$.
3. *Total cost:* $\sqrt{n}$ outer queries $\times$ ($\sqrt{n} \cdot d$ per query) $= n \cdot d$.

Quantum OV in $\tilde{O}(n \cdot d)$

Algorithm. Nested Grover, with amplitude amplification on each layer.

1. *Outer search* over $a \in A$ for an a that has an orthogonal partner in B. Each oracle call to the outer Grover invokes.
2. *Inner search* over $b \in B$ for $\langle a, b \rangle = 0$. Time per inner step: $O(d)$ to compute inner product. Inner cost: $\sqrt{n} \cdot d$.
3. *Total cost:* $\sqrt{n}$ outer queries $\times$ ($\sqrt{n} \cdot d$ per query) $= n \cdot d$.

Subtlety: bounded error. Each level of Grover is bounded-error. Wrapping inner search with amp. amp. keeps the error $O(1/n)$ per outer query — total error $O(1)$ after union bound.

Quantum OV in $\tilde{O}(n \cdot d)$

Algorithm. Nested Grover, with amplitude amplification on each layer.

1. *Outer search* over $a \in A$ for an a that has an orthogonal partner in B. Each oracle call to the outer Grover invokes.
2. *Inner search* over $b \in B$ for $\langle a, b \rangle = 0$. Time per inner step: $O(d)$ to compute inner product. Inner cost: $\sqrt{n} \cdot d$.
3. *Total cost:* $\sqrt{n}$ outer queries $\times$ ($\sqrt{n} \cdot d$ per query) $= n \cdot d$.

Subtlety: bounded error. Each level of Grover is bounded-error. Wrapping inner search with amp. amp. keeps the error $O(1/n)$ per outer query — total error $O(1)$ after union bound.

Comparison.

$$\underbrace{n^2 \cdot d}_{\text{classical}} \longrightarrow \underbrace{n \cdot d \cdot \operatorname{polylog} n}_{\text{quantum}}.$$

Quantum OV in $\tilde{O}(n \cdot d)$

Algorithm. Nested Grover, with amplitude amplification on each layer.

1. *Outer search* over $a \in A$ for an a that has an orthogonal partner in B. Each oracle call to the outer Grover invokes.
2. *Inner search* over $b \in B$ for $\langle a, b \rangle = 0$. Time per inner step: $O(d)$ to compute inner product. Inner cost: $\sqrt{n} \cdot d$.
3. *Total cost:* $\sqrt{n}$ outer queries $\times$ ($\sqrt{n} \cdot d$ per query) $= n \cdot d$.

Subtlety: bounded error. Each level of Grover is bounded-error. Wrapping inner search with amp. amp. keeps the error $O(1/n)$ per outer query — total error $O(1)$ after union bound.

Comparison.

$$\underbrace{n^2 \cdot d}_{\text{classical}} \longrightarrow \underbrace{n \cdot d \cdot \operatorname{polylog} n}_{\text{quantum}}.$$

Question. Can quantum do better than n? Or is $n^{1-o(1)}$ a quantum barrier?

The strawman QSETH (and why it fails)

First attempt. "k-SAT requires $\Omega(2^{n/2})$ quantum time."

The strawman QSETH (and why it fails)

First attempt. "k-SAT requires $\Omega(2^{n/2})$ quantum time."

Motivation: Grover gives $O(2^{n/2})$ for k-SAT (search the assignment space). So the "Grover barrier" should be the analogue of SETH.

The strawman QSETH (and why it fails)

First attempt. "k-SAT requires $\Omega(2^{n/2})$ quantum time."

Motivation: Grover gives $O(2^{n/2})$ for k-SAT (search the assignment space). So the "Grover barrier" should be the analogue of SETH.

Issue 1. Too strong. For sparse k-SAT (few clauses), better algorithms exist — $\tilde{O}(2^{n/2})$ is not the only quantum bound to worry about.

The strawman QSETH (and why it fails)

First attempt. "k-SAT requires $\Omega(2^{n/2})$ quantum time."

Motivation: Grover gives $O(2^{n/2})$ for k-SAT (search the assignment space). So the "Grover barrier" should be the analogue of SETH.

Issue 1. Too strong. For sparse k-SAT (few clauses), better algorithms exist — $\tilde{O}(2^{n/2})$ is not the only quantum bound to worry about.

Issue 2. Restricted versions of SAT behave *very* differently:

- Quantum #k-SAT (count satisfying assignments) needs 2^n, not $2^{n/2}$.

The strawman QSETH (and why it fails)

First attempt. "k-SAT requires $\Omega(2^{n/2})$ quantum time."

Motivation: Grover gives $O(2^{n/2})$ for k-SAT (search the assignment space). So the "Grover barrier" should be the analogue of SETH.

Issue 1. Too strong. For sparse k-SAT (few clauses), better algorithms exist — $\tilde{O}(2^{n/2})$ is not the only quantum bound to worry about.

Issue 2. Restricted versions of SAT behave *very* differently:

- Quantum $\#k$-SAT (count satisfying assignments) needs 2^n, not $2^{n/2}$.
- Quantum $\oplus$-k-SAT (parity of satisfying assignments) also needs 2^n.

The strawman QSETH (and why it fails)

First attempt. "k-SAT requires $\Omega(2^{n/2})$ quantum time."

Motivation: Grover gives $O(2^{n/2})$ for k-SAT (search the assignment space). So the "Grover barrier" should be the analogue of SETH.

Issue 1. Too strong. For sparse k-SAT (few clauses), better algorithms exist — $\tilde{O}(2^{n/2})$ is not the only quantum bound to worry about.

Issue 2. Restricted versions of SAT behave *very* differently:

- Quantum $\#k$-SAT (count satisfying assignments) needs 2^n, not $2^{n/2}$.
- Quantum $\oplus$-k-SAT (parity of satisfying assignments) also needs 2^n.

$\Rightarrow$ Need a more careful framework.

Buhrman–Patro–Speelman QSETH (STACS 2021)

Idea. For a k-CNF φ on n variables, let $T_\varphi \in \{0,1\}^{2^n}$ be its truth table.

Buhrman–Patro–Speelman QSETH (STACS 2021)

Idea. For a k-CNF φ on n variables, let $T_\varphi \in \{0,1\}^{2^n}$ be its truth table. Computing φ on any specific assignment is cheap; computing *global properties* of T_φ is the hard part.

Let $f : \{0,1\}^{2^n} \to \{0,1\}$ be a Boolean property.

Buhrman–Patro–Speelman QSETH (STACS 2021)

Idea. For a k-CNF φ on n variables, let $T_\varphi \in \{0,1\}^{2^n}$ be its truth table. Computing φ on any specific assignment is cheap; computing *global properties* of T_φ is the hard part.

Let $f : \{0,1\}^{2^n} \to \{0,1\}$ be a Boolean property.
Recall $\mathrm{Q}(f)$ be its bounded-error quantum query complexity.

Buhrman–Patro–Speelman QSETH (STACS 2021)

Idea. For a k-CNF φ on n variables, let $T_\varphi \in \{0,1\}^{2^n}$ be its truth table. Computing φ on any specific assignment is cheap; computing *global properties* of T_φ is the hard part.

Let $f : \{0,1\}^{2^n} \to \{0,1\}$ be a Boolean property.
Recall $\mathrm{Q}(f)$ be its bounded-error quantum query complexity.

QSETH **(informal).** For every $\epsilon > 0$, there exists k such that no quantum algorithm computes $f(T_\varphi)$ for k-CNF φ in time

$$2^{(\log_2 \mathrm{Q}(f)/n - \epsilon)\, n} \;=\; \mathrm{Q}(f)^{1-o(1)} \quad \text{(treating } N = 2^n\text{)}.$$

Buhrman–Patro–Speelman QSETH (STACS 2021)

Idea. For a k-CNF φ on n variables, let $T_\varphi \in \{0,1\}^{2^n}$ be its truth table. Computing φ on any specific assignment is cheap; computing *global properties* of T_φ is the hard part.

Let $f : \{0,1\}^{2^n} \to \{0,1\}$ be a Boolean property.
Recall $\mathrm{Q}(f)$ be its bounded-error quantum query complexity.

QSETH **(informal).** For every $\epsilon > 0$, there exists k such that no quantum algorithm computes $f(T_\varphi)$ for k-CNF φ in time

$$2^{(\log_2 \mathrm{Q}(f)/n - \epsilon)\, n} \;=\; \mathrm{Q}(f)^{1-o(1)} \quad \text{(treating } N = 2^n\text{)}.$$

Examples.

Buhrman–Patro–Speelman QSETH (STACS 2021)

Idea. For a k-CNF φ on n variables, let $T_\varphi \in \{0,1\}^{2^n}$ be its truth table. Computing φ on any specific assignment is cheap; computing *global properties* of T_φ is the hard part.

Let $f : \{0,1\}^{2^n} \to \{0,1\}$ be a Boolean property.
Recall $\mathrm{Q}(f)$ be its bounded-error quantum query complexity.

QSETH **(informal).** For every $\epsilon > 0$, there exists k such that no quantum algorithm computes $f(T_\varphi)$ for k-CNF φ in time

$$2^{(\log_2 \mathrm{Q}(f)/n-\epsilon)\, n} \;=\; \mathrm{Q}(f)^{1-o(1)} \quad \text{(treating } N = 2^n\text{)}.$$

Examples.

- $f = \mathrm{OR}$ (k-SAT): $\mathrm{Q}(\mathrm{OR}) = \Theta(\sqrt{N}) \;\Rightarrow\; 2^{n/2}$.

Buhrman–Patro–Speelman QSETH (STACS 2021)

Idea. For a k-CNF φ on n variables, let $T_\varphi \in \{0,1\}^{2^n}$ be its truth table. Computing φ on any specific assignment is cheap; computing *global properties* of T_φ is the hard part.

Let $f : \{0,1\}^{2^n} \to \{0,1\}$ be a Boolean property.
Recall $\mathrm{Q}(f)$ be its bounded-error quantum query complexity.

QSETH **(informal).** For every $\epsilon > 0$, there exists k such that no quantum algorithm computes $f(T_\varphi)$ for k-CNF φ in time

$$2^{(\log_2 \mathrm{Q}(f)/n-\epsilon)\, n} \;=\; \mathrm{Q}(f)^{1-o(1)} \quad \text{(treating } N = 2^n\text{)}.$$

Examples.

- $f = \mathrm{OR}$ (k-SAT): $\mathrm{Q}(\mathrm{OR}) = \Theta(\sqrt{N}) \;\Rightarrow\; 2^{n/2}$.
- $f = \mathrm{MAJ}$ (MAJ-k-SAT): $\mathrm{Q}(\mathrm{MAJ}) = \Theta(N) \;\Rightarrow\; 2^n$.

Buhrman–Patro–Speelman QSETH (STACS 2021)

Idea. For a k-CNF φ on n variables, let $T_\varphi \in \{0,1\}^{2^n}$ be its truth table. Computing φ on any specific assignment is cheap; computing *global properties* of T_φ is the hard part.

Let $f : \{0,1\}^{2^n} \to \{0,1\}$ be a Boolean property.
Recall $\mathrm{Q}(f)$ be its bounded-error quantum query complexity.

QSETH **(informal).** For every $\epsilon > 0$, there exists k such that no quantum algorithm computes $f(T_\varphi)$ for k-CNF φ in time

$$2^{(\log_2 \mathrm{Q}(f)/n - \epsilon)\, n} \;=\; \mathrm{Q}(f)^{1-o(1)} \quad \text{(treating } N = 2^n\text{)}.$$

Examples.

- $f = \mathrm{OR}$ (k-SAT): $\mathrm{Q}(\mathrm{OR}) = \Theta(\sqrt{N}) \;\Rightarrow\; 2^{n/2}$.
- $f = \mathrm{MAJ}$ (MAJ-k-SAT): $\mathrm{Q}(\mathrm{MAJ}) = \Theta(N) \;\Rightarrow\; 2^n$.
- $f = \mathrm{PARITY}$ ($\oplus$-k-SAT): $\mathrm{Q}(\mathrm{PARITY}) = N/2 \;\Rightarrow\; 2^n$.

Buhrman–Patro–Speelman QSETH (STACS 2021)

Idea. For a k-CNF φ on n variables, let $T_\varphi \in \{0,1\}^{2^n}$ be its truth table. Computing φ on any specific assignment is cheap; computing *global properties* of T_φ is the hard part.

Let $f : \{0,1\}^{2^n} \to \{0,1\}$ be a Boolean property.
Recall $\mathrm{Q}(f)$ be its bounded-error quantum query complexity.

QSETH **(informal).** For every $\epsilon > 0$, there exists k such that no quantum algorithm computes $f(T_\varphi)$ for k-CNF φ in time

$$2^{(\log_2 \mathrm{Q}(f)/n - \epsilon)\, n} \;=\; \mathrm{Q}(f)^{1-o(1)} \quad \text{(treating } N = 2^n\text{)}.$$

Examples.

- $f = \mathrm{OR}$ (k-SAT): $\mathrm{Q}(\mathrm{OR}) = \Theta(\sqrt{N}) \;\Rightarrow\; 2^{n/2}$.
- $f = \mathrm{MAJ}$ (MAJ-k-SAT): $\mathrm{Q}(\mathrm{MAJ}) = \Theta(N) \;\Rightarrow\; 2^n$.
- $f = \mathrm{PARITY}$ ($\oplus$-k-SAT): $\mathrm{Q}(\mathrm{PARITY}) = N/2 \;\Rightarrow\; 2^n$.

Anatomy of a quantum fine-grained reduction

Classical reduction:

$$A \text{ on size } n \longrightarrow B \text{ on size } f(n).$$

Quantum analog has *three* subtleties not present classically:

Anatomy of a quantum fine-grained reduction

Classical reduction:

$$A \text{ on size } n \longrightarrow B \text{ on size } f(n).$$

Quantum analog has *three* subtleties not present classically:

1. **Oracle compatibility.** Reduction must convert quantum queries to B into quantum queries to A's instance, *coherently*. "Classical query reduction" doesn't suffice if the A-algorithm is in superposition. "on-the-fly"

Anatomy of a quantum fine-grained reduction

Classical reduction:

$$A \text{ on size } n \longrightarrow B \text{ on size } f(n).$$

Quantum analog has *three* subtleties not present classically:

1. **Oracle compatibility.** Reduction must convert quantum queries to B into quantum queries to A's instance, *coherently*. "Classical query reduction" doesn't suffice if the A-algorithm is in superposition. "on-the-fly"
2. **Bounded error and amplitude.** Errors amplify under composition. Need $O(1/\text{poly})$ inner error to maintain $O(1)$ outer success, or rely on amp. amp. with exact-counting variants.

Anatomy of a quantum fine-grained reduction

Classical reduction:

$$A \text{ on size } n \longrightarrow B \text{ on size } f(n).$$

Quantum analog has *three* subtleties not present classically:

1. **Oracle compatibility.** Reduction must convert quantum queries to B into quantum queries to A's instance, *coherently*. "Classical query reduction" doesn't suffice if the A-algorithm is in superposition. "on-the-fly"
2. **Bounded error and amplitude.** Errors amplify under composition. Need $O(1/\text{poly})$ inner error to maintain $O(1)$ outer success, or rely on amp. amp. with exact-counting variants.
3. **Square-roots in the right places.** A classical reduction time must shrink quantumly.

Worked: QSETH $\Rightarrow$ QOV

Claim. If OV is solvable in $n^{1-\epsilon}$ quantum time, then QSETH is false.

Proof.

Worked: QSETH $\Rightarrow$ QOV

Claim. If OV is solvable in $n^{1-\epsilon}$ quantum time, then QSETH is false.

Proof.

1. Take k-CNF φ on n variables. Apply Williams' reduction: build OV instance (A, B) with $|A| = |B| = N := 2^{n/2}$, dimension $d = m = \text{poly}(n)$.

Worked: QSETH $\Rightarrow$ QOV

Claim. If OV is solvable in $n^{1-\epsilon}$ quantum time, then QSETH is false.

Proof.

1. Take k-CNF φ on n variables. Apply Williams' reduction: build OV instance (A, B) with $|A| = |B| = N := 2^{n/2}$, dimension $d = m = \text{poly}(n)$.
2. OV vectors are poly(n)-time computable from indices. So we can implement an oracle that on input α (label of a vector) outputs a_α in poly(n) quantum time — *coherently*.

Worked: QSETH $\Rightarrow$ QOV

Claim. If OV is solvable in $n^{1-\epsilon}$ quantum time, then QSETH is false.

Proof.

1. Take k-CNF φ on n variables. Apply Williams' reduction: build OV instance (A, B) with $|A| = |B| = N := 2^{n/2}$, dimension $d = m = \text{poly}(n)$.
2. OV vectors are poly(n)-time computable from indices. So we can implement an oracle that on input α (label of a vector) outputs a_α in poly(n) quantum time — *coherently*.
3. Run the assumed QOV algorithm: time $N^{1-\epsilon} \cdot \text{poly}(d) = 2^{(1/2-\epsilon/2)n}\,\text{poly}(n)$.

Worked: QSETH $\Rightarrow$ QOV

Claim. If OV is solvable in $n^{1-\epsilon}$ quantum time, then QSETH is false.

Proof.

1. Take k-CNF φ on n variables. Apply Williams' reduction: build OV instance (A, B) with $|A| = |B| = N := 2^{n/2}$, dimension $d = m = \text{poly}(n)$.
2. OV vectors are poly(n)-time computable from indices. So we can implement an oracle that on input α (label of a vector) outputs a_α in poly(n) quantum time — *coherently*.
3. Run the assumed QOV algorithm: time $N^{1-\epsilon} \cdot \text{poly}(d) = 2^{(1/2-\epsilon/2)n}\,\text{poly}(n)$.
4. OV decides the existence of an orthogonal pair, which decides the satisfiability of φ.

Worked: QSETH $\Rightarrow$ QOV

Claim. If OV is solvable in $n^{1-\epsilon}$ quantum time, then QSETH is false.

Proof.

1. Take k-CNF φ on n variables. Apply Williams' reduction: build OV instance (A, B) with $|A| = |B| = N := 2^{n/2}$, dimension $d = m = \text{poly}(n)$.
2. OV vectors are poly(n)-time computable from indices. So we can implement an oracle that on input α (label of a vector) outputs a_α in poly(n) quantum time — *coherently*.
3. Run the assumed QOV algorithm: time $N^{1-\epsilon} \cdot \text{poly}(d) = 2^{(1/2-\epsilon/2)n}\,\text{poly}(n)$.
4. OV decides the existence of an orthogonal pair, which decides the satisfiability of φ.
5. Time for k-SAT: $2^{(1/2-\epsilon/2)n}$.

Worked: QSETH $\Rightarrow$ QOV

Claim. If OV is solvable in $n^{1-\epsilon}$ quantum time, then QSETH is false.

Proof.

1. Take k-CNF φ on n variables. Apply Williams' reduction: build OV instance (A, B) with $|A| = |B| = N := 2^{n/2}$, dimension $d = m = \text{poly}(n)$.
2. OV vectors are poly(n)-time computable from indices. So we can implement an oracle that on input α (label of a vector) outputs a_α in poly(n) quantum time — *coherently*.
3. Run the assumed QOV algorithm: time $N^{1-\epsilon} \cdot \text{poly}(d) = 2^{(1/2-\epsilon/2)n}\,\text{poly}(n)$.
4. OV decides the existence of an orthogonal pair, which decides the satisfiability of φ.
5. Time for k-SAT: $2^{(1/2-\epsilon/2)n}$.

This is faster than $2^{n/2}$, contradicting QSETH instantiated with $f = \text{OR}$ ($\text{Q(OR)} = \sqrt{N}$). □

Where the quantum picture is incomplete

- **Edit distance / Longest Common Sequence** No quantum algorithm beats $n^{2-o(1)}$ meaningfully; no quantum lower bound rules out $n^{1+o(1)}$.

Where the quantum picture is incomplete

- **Edit distance / Longest Common Sequence** No quantum algorithm beats $n^{2-o(1)}$ meaningfully; no quantum lower bound rules out $n^{1+o(1)}$.
- k-**SUM for** $k \geq 4$**.** Classical $n^{\lceil k/2 \rceil}$; quantum walks give $n^{k/3}$ (Belovs–Špalek).

Where the quantum picture is incomplete

- **Edit distance / Longest Common Sequence** No quantum algorithm beats $n^{2-o(1)}$ meaningfully; no quantum lower bound rules out $n^{1+o(1)}$.
- k-**SUM for** $k \geq 4$**.** Classical $n^{\lceil k/2 \rceil}$; quantum walks give $n^{k/3}$ (Belovs–Špalek).
- **APSP vs.Negative Triangle Finding** Classcially sub-cubic equivalent but quantumly NTF can be solved in $\widetilde{O}(n^{3/2})$ time.

Where the quantum picture is incomplete

- **Edit distance / Longest Common Sequence** No quantum algorithm beats $n^{2-o(1)}$ meaningfully; no quantum lower bound rules out $n^{1+o(1)}$.
- **k-SUM for $k \geq 4$.** Classical $n^{\lceil k/2 \rceil}$; quantum walks give $n^{k/3}$ (Belovs–Špalek).
- **APSP vs.Negative Triangle Finding** Classcially sub-cubic equivalent but quantumly NTF can be solved in $\widetilde{O}(n^{3/2})$ time.
- **APSP vs. Maximum Subarrary** Classcially sub-cubic equivalent but quantumly MS can be olved in $\widetilde{O}(n^2)$ time.

Where the quantum picture is incomplete

- **Edit distance / Longest Common Sequence** No quantum algorithm beats $n^{2-o(1)}$ meaningfully; no quantum lower bound rules out $n^{1+o(1)}$.
- **k-SUM for $k \geq 4$.** Classical $n^{\lceil k/2 \rceil}$; quantum walks give $n^{k/3}$ (Belovs–Špalek).
- **APSP vs.Negative Triangle Finding** Classcially sub-cubic equivalent but quantumly NTF can be solved in $\widetilde{O}(n^{3/2})$ time.
- **APSP vs. Maximum Subarrary** Classcially sub-cubic equivalent but quantumly MS can be olved in $\widetilde{O}(n^2)$ time.

Average-case and approximation

Average-case quantum FGC. For cryptographically relevant inputs (random, structured), do we still get tight lower bounds?

Average-case and approximation

Average-case quantum FGC. For cryptographically relevant inputs (random, structured), do we still get tight lower bounds?

Compressed-oracle / recording techniques (Zhandry '18) give average-case lower bounds in the QROM.

Average-case and approximation

Average-case quantum FGC. For cryptographically relevant inputs (random, structured), do we still get tight lower bounds?

Compressed-oracle / recording techniques (Zhandry '18) give average-case lower bounds in the QROM. Connecting these to fine-grained *worst-case* problems is an active area.

Average-case and approximation

Average-case quantum FGC. For cryptographically relevant inputs (random, structured), do we still get tight lower bounds?

Compressed-oracle / recording techniques (Zhandry '18) give average-case lower bounds in the QROM. Connecting these to fine-grained *worst-case* problems is an active area.

Average-case and approximation

Average-case quantum FGC. For cryptographically relevant inputs (random, structured), do we still get tight lower bounds?

Compressed-oracle / recording techniques (Zhandry '18) give average-case lower bounds in the QROM. Connecting these to fine-grained *worst-case* problems is an active area.

Quantum SETH-hardness of approximation.

- Classically: SETH-hardness of approximating closest pair, bichromatic LCS, max-k-OV, etc.

Average-case and approximation

Average-case quantum FGC. For cryptographically relevant inputs (random, structured), do we still get tight lower bounds?

Compressed-oracle / recording techniques (Zhandry '18) give average-case lower bounds in the QROM. Connecting these to fine-grained *worst-case* problems is an active area.

Quantum SETH-hardness of approximation.

- Classically: SETH-hardness of approximating closest pair, bichromatic LCS, max-k-OV, etc.
- Quantumly: No results. The reductions for inapproximability blow up gadget sizes, and the $\sqrt{\ }$ savings get spent on the blow-up.

Average-case and approximation

Average-case quantum FGC. For cryptographically relevant inputs (random, structured), do we still get tight lower bounds?

Compressed-oracle / recording techniques (Zhandry '18) give average-case lower bounds in the QROM. Connecting these to fine-grained *worst-case* problems is an active area.

Quantum SETH-hardness of approximation.

- Classically: SETH-hardness of approximating closest pair, bichromatic LCS, max-k-OV, etc.
- Quantumly: No results. The reductions for inapproximability blow up gadget sizes, and the $\sqrt{\ }$ savings get spent on the blow-up.

Comparison: classical vs. quantum FGC

Problem	Classical (cond.)	Quantum (cond.)	Status
k-SAT	$2^{(1-o(1))n}$	$2^{(1/2-o(1))n}$	SETH / QSETH
OV	$n^{2-o(1)}$	$n^{1-o(1)}$	under QSETH
3SUM	$n^{2-o(1)}$	$n^{1-o(1)}$	under Q3SUM (Grover after sort)
APSP	$n^{3-o(1)}$	$n^{5/2-o(1)}$	under QAPSP

Comparison: classical vs. quantum FGC

Problem	Classical (cond.)	Quantum (cond.)	Status
k-SAT	$2^{(1-o(1))n}$	$2^{(1/2-o(1))n}$	SETH / QSETH
OV	$n^{2-o(1)}$	$n^{1-o(1)}$	under QSETH
3SUM	$n^{2-o(1)}$	$n^{1-o(1)}$	under Q3SUM (Grover after sort)
APSP	$n^{3-o(1)}$	$n^{5/2-o(1)}$	under QAPSP

Take-aways.

- Quantum FGC conjectures require less time — the reduction needs real care about coherence and bounded error.

Comparison: classical vs. quantum FGC

Problem	Classical (cond.)	Quantum (cond.)	Status
k-SAT	$2^{(1-o(1))n}$	$2^{(1/2-o(1))n}$	SETH / QSETH
OV	$n^{2-o(1)}$	$n^{1-o(1)}$	under QSETH
3SUM	$n^{2-o(1)}$	$n^{1-o(1)}$	under Q3SUM (Grover after sort)
APSP	$n^{3-o(1)}$	$n^{5/2-o(1)}$	under QAPSP

Take-aways.

- Quantum FGC conjectures require less time — the reduction needs real care about coherence and bounded error.
- The right hypothesis is not "SETH with $2^{n/2}$" but QSETH parameterized by query complexity.

Comparison: classical vs. quantum FGC

Problem	Classical (cond.)	Quantum (cond.)	Status
k-SAT	$2^{(1-o(1))n}$	$2^{(1/2-o(1))n}$	SETH / QSETH
OV	$n^{2-o(1)}$	$n^{1-o(1)}$	under QSETH
3SUM	$n^{2-o(1)}$	$n^{1-o(1)}$	under Q3SUM (Grover after sort)
APSP	$n^{3-o(1)}$	$n^{5/2-o(1)}$	under QAPSP

Take-aways.

- Quantum FGC conjectures require less time — the reduction needs real care about coherence and bounded error.
- The right hypothesis is not "SETH with $2^{n/2}$" but QSETH parameterized by query complexity.
- For many problems (like sequence problems), the algorithmic side lags far behind the lower-bound side. Closing this gap is a defining open problem.

Further reading

- V. Vassilevska Williams. *On some fine-grained questions in algorithms and complexity.* ICM 2018. (Best classical survey.)
- Buhrman, Patro, Speelman. *A framework of quantum strong exponential-time hypotheses.* STACS 2021.
- Aaronson, Chia, Lin, Wang, Zhang. *On the quantum complexity of closest pair and related problems.* CCC 2020.
- Williams. *A new algorithm for optimal* 2*-constraint satisfaction and its implications.* TCS 2005. (SETH→OV.)
- Backurs, Indyk. *Edit distance cannot be computed in strongly subquadratic time (unless SETH is false).* STOC 2015.
- Bringmann, Künnemann. *Quadratic conditional lower bounds for string problems.* FOCS 2015.
- Le Gall. *Improved quantum algorithm for the all-pairs shortest paths problem.* QIC 2014.
- Magniez, Nayak, Roland, Santha. *Search via quantum walk.* STOC 2007.
- Dürr, Høyer, Mhalla, Lei. *Quantum query complexity of some graph problems.* SICOMP 2006.
- Belovs, Špalek. *Adversary lower bound for the k-sum problem.* ITCS 2013.

Questions?