Gems of TCS

Fine-grained Complexity

Sasha Golovnev

September 13, 2023

Fine-grained Complexity

- Efficient algorithms for important problems?

Fine-grained Complexity

- Efficient algorithms for important problems?
- For many of them, we couldn't find better algorithms in decades

Fine-grained Complexity

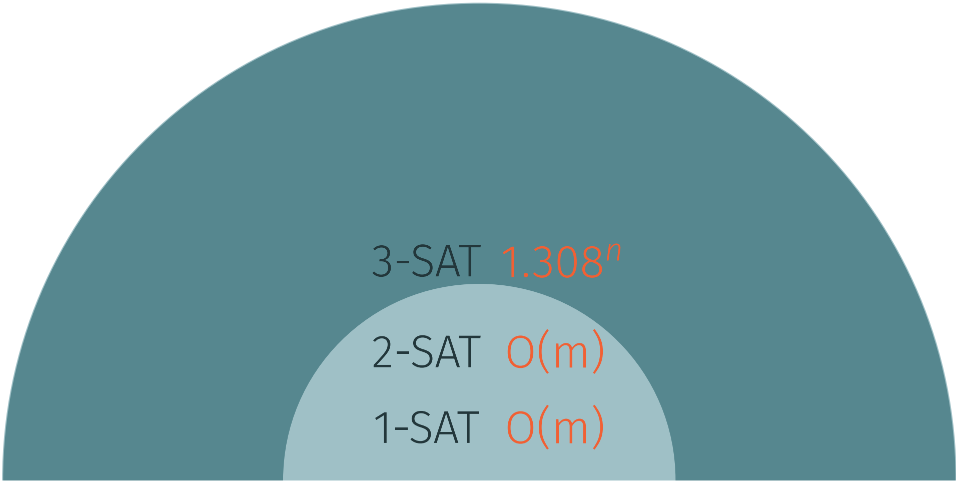
- Efficient algorithms for important problems?
- For many of them, we couldn't find better algorithms in decades
- Today: Identify reason why we're stuck

Algorithmic Complexity of SAT

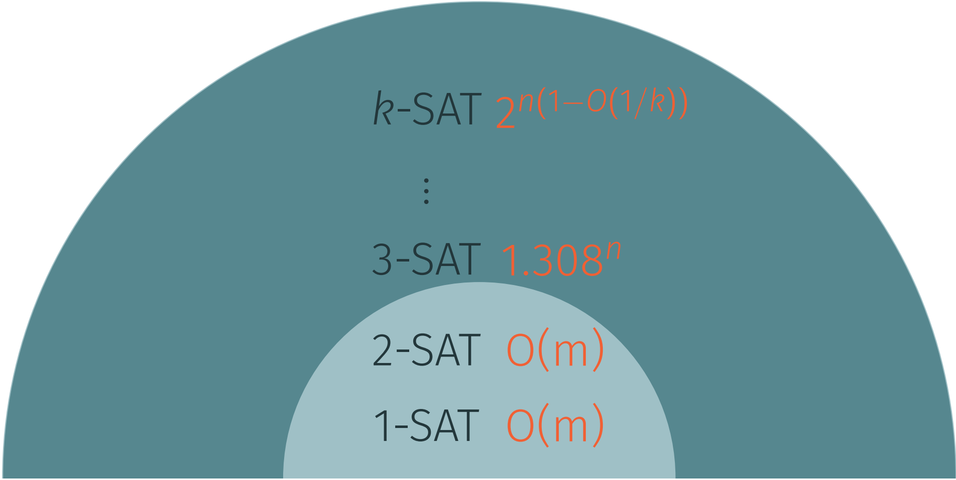
2-SAT $O(m)$

1-SAT $O(m)$

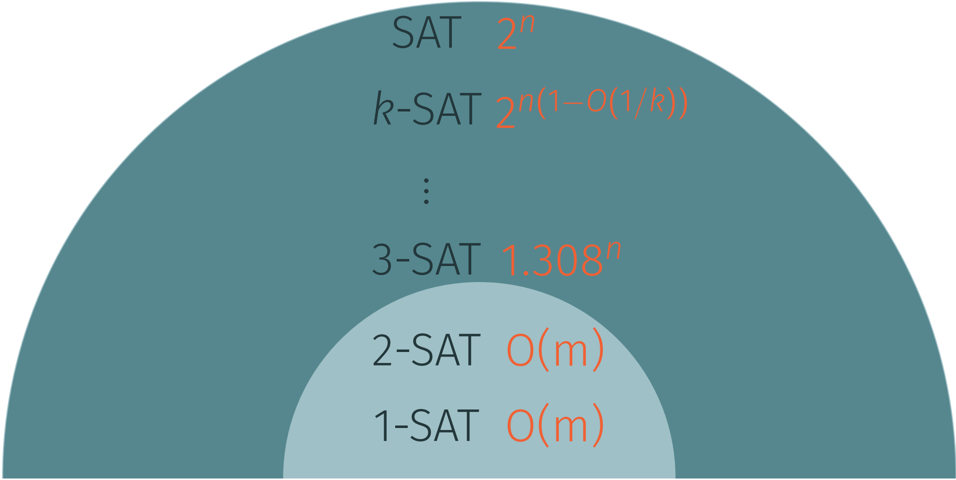
Algorithmic Complexity of SAT



Algorithmic Complexity of SAT



Algorithmic Complexity of SAT



Hardness of SAT

- SAT can be solved in time $2^n \operatorname{poly}(n)$
- We don't know how to solve SAT exponentially faster: in time 1.999^n

Hardness of SAT

- SAT can be solved in time $2^n \operatorname{poly}(n)$
- We don't know how to solve SAT exponentially faster: in time 1.999^n
- Strong Exponential Time Hypothesis (SETH)

SAT requires time 2^n

Edit Distance

Edit Distance

Edit Distance

e l e p h a n t

r e l e v a n t

Edit Distance

Edit Distance

e l e p h a n t

~~r~~ e l e v a n t

p

Edit Distance

Edit Distance

e l e p h a n t

~~r~~ e l e v a n t

p

A T A G T A C T

~~C~~ A T A C A C T

G

Edit Distance

Edit Distance

e l e p h a n t

~~r~~ e l e v a n t

p

A T A G T A C T

~~C~~ A T A C A C T

G

$$\widetilde{O}(n^2)$$

Other Problems

Longest Common Subsequence	Orthogonal Vectors	Edit Distance
Hamming Closest Pair	All Pairs Max Flow	RNA-Folding
Regular Expression Matching	Graph Diameter	Subset Sum

Conjectured Hardness

- A conjecture for each problem?

Conjectured Hardness

- A conjecture for each problem?
- One conjecture to rule them all?

Conjectured Hardness

- A conjecture for each problem?
- One conjecture to rule them all?
- Fine-grained Complexity: Better-than-known algorithms for one problem would imply better-than-known algorithms for other problems

Orthogonal Vectors (OV)

Orthogonal Vectors Problem

- S, T are sets of N vectors from $\{0,1\}^d$. Are there $s \in S$ and $t \in T$ such that $s \cdot t = \sum_{i=1}^{d} s_i \cdot t_i = 0$?

Orthogonal Vectors Problem

- S, T are sets of N vectors from $\{0,1\}^d$. Are there $s \in S$ and $t \in T$ such that $s \cdot t = \sum_{i=1}^{d} s_i \cdot t_i = 0$?
- Think of $d = \log^2 N$

Orthogonal Vectors Problem

- S, T are sets of N vectors from $\{0,1\}^d$. Are there $s \in S$ and $t \in T$ such that $s \cdot t = \sum_{i=1}^{d} s_i \cdot t_i = 0$?
- Think of $d = \log^2 N$
- Can solve in time $d \cdot N^2$

Orthogonal Vectors Problem

- S, T are sets of N vectors from $\{0,1\}^d$. Are there $s \in S$ and $t \in T$ such that $s \cdot t = \sum_{i=1}^{d} s_i \cdot t_i = 0$?
- Think of $d = \log^2 N$
- Can solve in time $d \cdot N^2$
- SETH implies that OV cannot be solved in time $N^{1.99}$

Fine-grained Reductions

formula ϕ of SAT

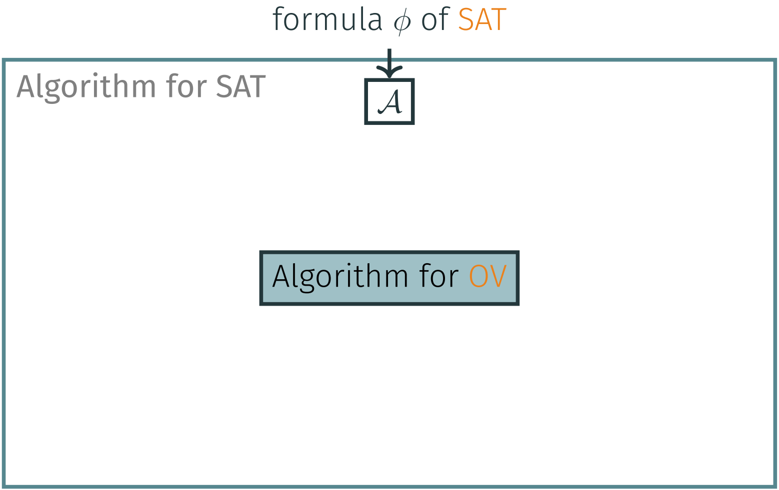
Fine-grained Reductions

formula ϕ of SAT

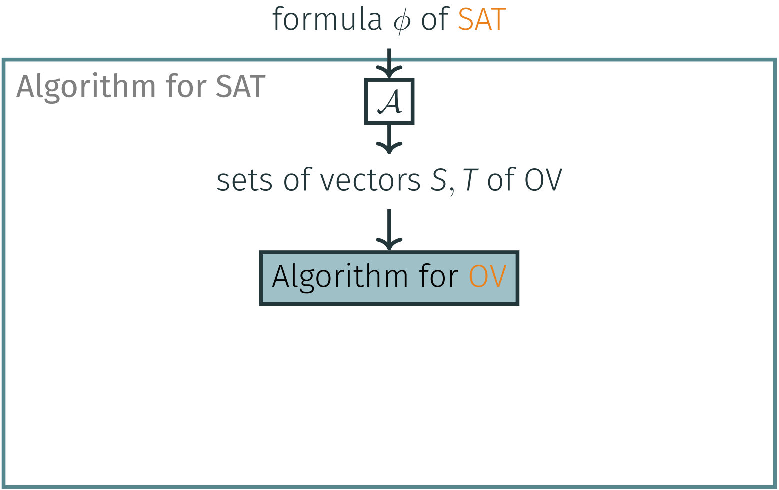
Algorithm for SAT

Algorithm for OV

Fine-grained Reductions



Fine-grained Reductions



Fine-grained Reductions

formula ϕ of SAT

Algorithm for SAT

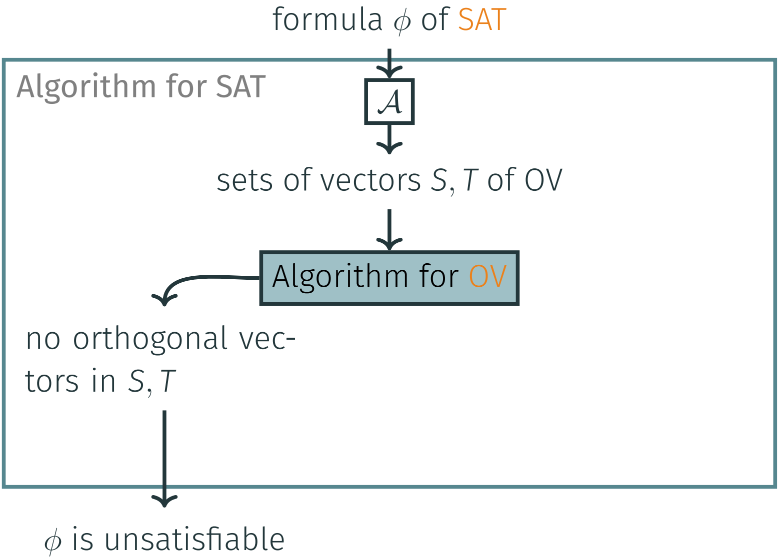
$\mathcal{A}$

sets of vectors S, T of OV

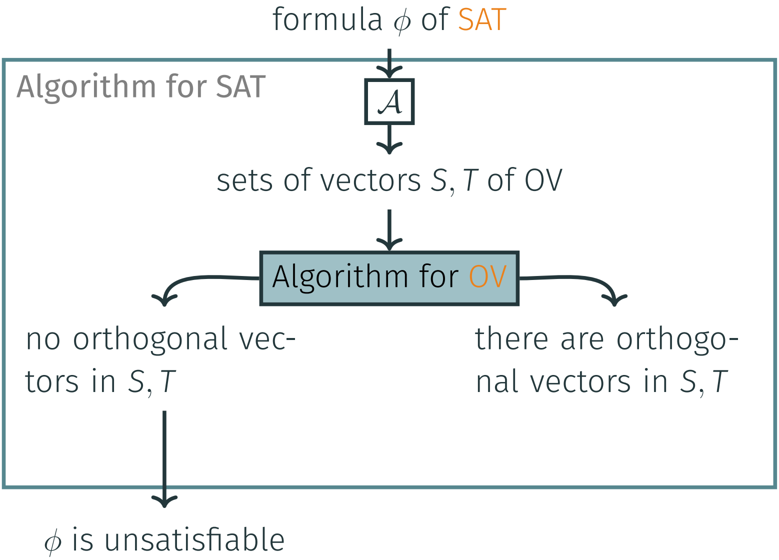
Algorithm for OV

no orthogonal vectors in S, T

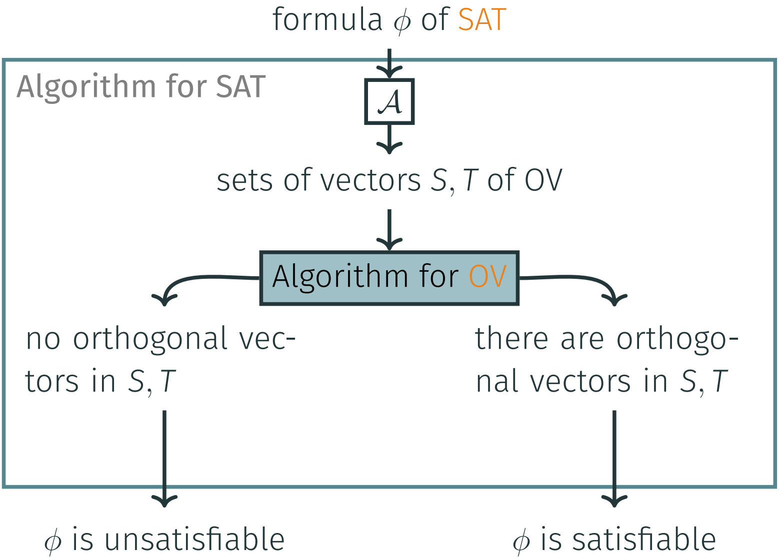
Fine-grained Reductions



Fine-grained Reductions



Fine-grained Reductions



SETH $\implies$ OV

- Given a SAT formula ϕ, split its n input variables into two sets of size $n/2$

SETH $\implies$ OV

- Given a SAT formula ϕ, split its n input variables into two sets of size $n/2$
- For each assignment to the first group — a vector in S, for each assignment to the second — a vector in T

SETH $\implies$ OV

- Given a SAT formula ϕ, split its n input variables into two sets of size $n/2$
- For each assignment to the first group — a vector in S, for each assignment to the second — a vector in T
- $N = 2^{n/2}$

SETH $\implies$ OV

- For an assignment $x \in \{0,1\}^{n/2}$, add $s \in \{0,1\}^m$ to S:

$$s_i = 1 \text{ iff } x \text{ doesn't satisfy clause } C_i$$

SETH $\implies$ OV

- For an assignment $x \in \{0,1\}^{n/2}$, add $s \in \{0,1\}^m$ to S:

$$s_i = 1 \text{ iff } x \text{ doesn't satisfy clause } C_i$$

- ϕ is SAT iff $\exists s \in S, t \in T$:

$$\forall i \in \{1, \ldots, m\}: \quad s_i \cdot t_i = 0$$

SETH $\implies$ OV

- For an assignment $x \in \{0,1\}^{n/2}$, add $s \in \{0,1\}^m$ to S:

$$s_i = 1 \text{ iff } x \text{ doesn't satisfy clause } C_i$$

- ϕ is SAT iff $\exists s \in S, t \in T$:

$$\forall i \in \{1, \ldots, m\}: \quad s_i \cdot t_i = 0$$

- An $N^{1.99}$ algorithm for OV gives an algorithm for SAT with run time

$$N^{1.99}$$

SETH $\implies$ OV

- For an assignment $x \in \{0,1\}^{n/2}$, add $s \in \{0,1\}^m$ to S:

$$s_i = 1 \text{ iff } x \text{ doesn't satisfy clause } C_i$$

- ϕ is SAT iff $\exists s \in S, t \in T$:

$$\forall i \in \{1, \ldots, m\}: \quad s_i \cdot t_i = 0$$

- An $N^{1.99}$ algorithm for OV gives an algorithm for SAT with run time

$$N^{1.99} = (2^{n/2})^{1.99} = 2^{0.995n}$$

The Dominating Set Problem

Dominating Set

- *k*-Dominating Set: Given $G = (V, E), |V| = n$, find an $S \subseteq V, |S| = k$ such that

$$\forall v \in V\colon v \in S \text{ or } \exists u \in S\colon \ (v, u) \in E$$

Dominating Set

- *k*-Dominating Set: Given $G = (V, E), |V| = n$, find an $S \subseteq V, |S| = k$ such that

$$\forall v \in V\colon v \in S \text{ or } \exists u \in S\colon\ (v, u) \in E$$

- For $k \geq 7$, solvable in n^k

Dominating Set

- *k*-Dominating Set: Given $G = (V, E), |V| = n$, find an $S \subseteq V, |S| = k$ such that

$$\forall v \in V \colon v \in S \text{ or } \exists u \in S \colon \; (v, u) \in E$$

- For $k \geq 7$, solvable in n^k
- SETH implies that k-DS cannot be solved in time $n^{k-0.01}$ for any k

SETH $\implies$ DS

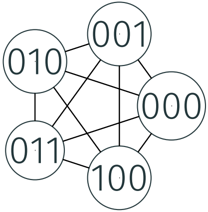
Partition vars in k groups:

$\{x_1, \ldots, x_{n/k}\}, \ldots, \{x_{n-n/k+1}, \ldots, x_n\}, |\mathrm{DS}| = k$

SETH $\implies$ DS

Partition vars in k groups:
$\{x_1, \ldots, x_{n/k}\}, \ldots, \{x_{n-n/k+1}, \ldots, x_n\}, |\mathrm{DS}| = k$

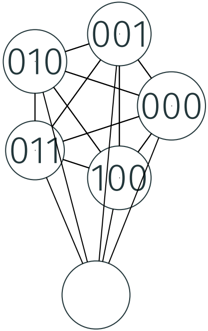
$2^{n/k}$ vertices



SETH $\implies$ DS

Partition vars in k groups:
$\{x_1, \ldots, x_{n/k}\}, \ldots, \{x_{n-n/k+1}, \ldots, x_n\}, |\mathrm{DS}| = k$

$2^{n/k}$ vertices

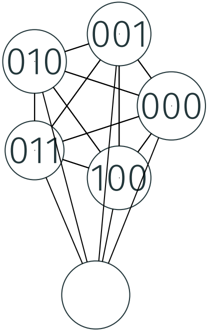


SETH $\implies$ DS

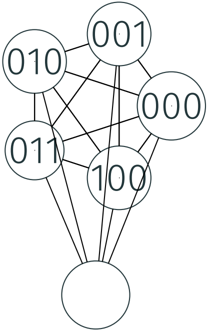
Partition vars in k groups:

$\{x_1, \ldots, x_{n/k}\}, \ldots, \{x_{n-n/k+1}, \ldots, x_n\}, |\mathrm{DS}| = k$

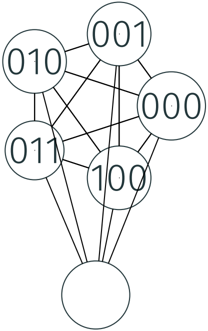
$2^{n/k}$ vertices



$2^{n/k}$ vertices



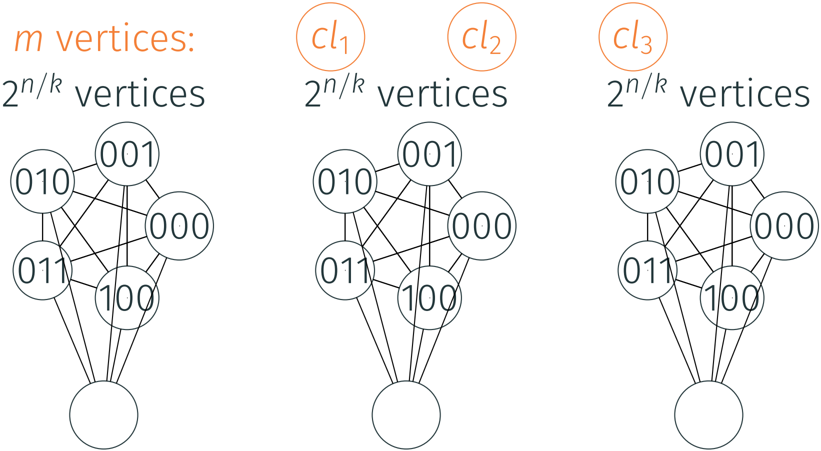
$2^{n/k}$ vertices



SETH $\implies$ DS

Partition vars in k groups:

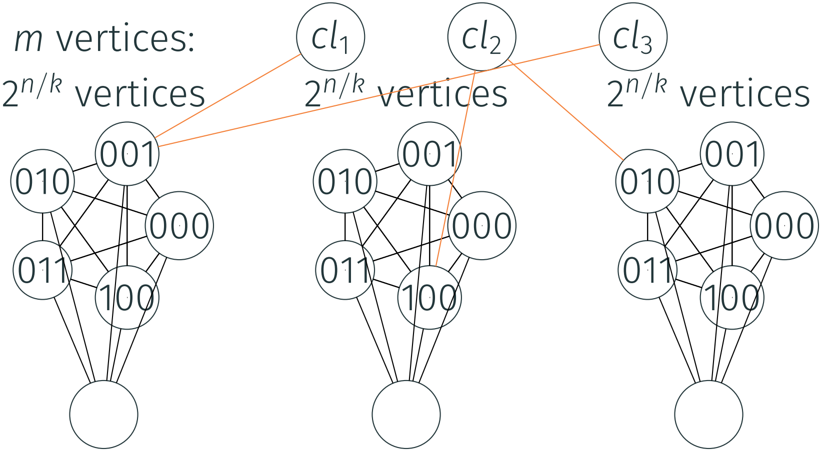
$\{x_1, \ldots, x_{n/k}\}, \ldots, \{x_{n-n/k+1}, \ldots, x_n\}, |\mathrm{DS}| = k$



SETH $\implies$ DS

Partition vars in k groups:

$\{x_1, \ldots, x_{n/k}\}, \ldots, \{x_{n-n/k+1}, \ldots, x_n\}, |\mathrm{DS}| = k$



SETH $\implies$ DS

- For every k, we reduce SAT on n vertices k-DS with

$$\approx 2^{n/k}$$

vertices

SETH $\implies$ DS

- For every k, we reduce SAT on n vertices k-DS with
$$\approx 2^{n/k}$$
vertices
- If k-DS on N vertices can be solved in time $N^{k-0.1}$, then SAT can be solved in time
$$N^{k-0.1} = 2^{(n/k)(k-0.1)} = 2^{n-0.1n/k}$$