Introduction to Fine-grained Complexity

Mohan Paturi

Department of Computer Science and Engineering
University of California, San Diego

18th Eduard Čech Lecture
Institute of Mathematics
Czech Academy of Sciences
May 2022

Outline

Computational Complexity

- Study of computation resources, time, space, randomness, ..., required to compute problems
- Guide the design of efficient algorithms for concrete problems
- Goal: Present a theory of fine-grained computational complexity and its progress
- Notes: Complexity is a function of the problem instance size parameters.

Some Ingredients of a Complexity Theory

- Problems and classes of problems
- Algorithms and design techniques
- Notions of reduction and complexity relationships among problems
- Hard and complete problems
- Conjectures
- (Conditional) Lower Bounds

Reductions

- Problem A is reducible to problem B by a reduction f if $x \in A$ if and only if $f(x) \in B$.
- To obtain meaningful complexity relationships between A and B, we need to limit the computational power of f
- A reduction is polynomial-time if f is polynomial-time computable,
- Under polynomial-time reductions, if A reduces to B and B is polynomial-time computable, then so is A.
- Contrapositively, if A is not polynomial-time computable, then B is also not polynomial-time computable.

NP Theory

- Problems: SATISFIABILITY, MAX INDEPENDENT SET, HAMILTONIAN PATH, COLORABILITY, CLIQUE, FACTORING, GRAPH ISOMORPHISM, PRIMALITY, ...
- Classes: **P**, **NP**, **coNP**, **L**, ...
- Notions of complexity relationships: Polynomial time reductions
- Complexity relationships: The following problems (and many others) are polynomially equivalent.
  k-SAT for $k \geq 3$, COLORABILITY, VERTEX COVER, INDEPENDENT SET, CLIQUE, $\cdots$
- Completeness: 3-SAT is complete for **NP**.
- Complexity conjecture: $\mathbf{P} \neq \mathbf{NP}$.
- Conditional lower bounds: None of the **NP**-complete problems have a polynomial time algorithm (under the conjecture $\mathbf{P} \neq \mathbf{NP}$).

What is fine-grained complexity?

- Theory and techniques to reason about
  - exact worst-case complexities of deterministic or randomized algorithms that output exact solutions and
  - complexity relationships among them.
- What improvements can we expect over exhaustive search or standard algorithms?
- What are the obstructions that limit improvements?
- What principles explain the exact complexities of problems?
- Similar to **NP**-theory but differs from **NP**-theory in the following respects
  - Problem-centric rather than complexity class-centric.
  - Strives to determine the complexity as exactly as possible.
  - Requires fine-grained reductions

Problems in **P**

ORTHOGONAL VECTORS Problem

ORTHOGONAL VECTORS (Bipartite version): Given two sequences $A_1, \ldots, A_n$ and $B_1, \ldots, B_n$ of sets with elements from a universe of size d, do there exist i and j such that $A_i \cap B_j = \varnothing$.
If the sets are thought of as characteristic vectors in $\{0,1\}^d$, $A_i \cap B_j = \varnothing$ is equivalent to the proposition that the vectors A_i and A_j are orthogonal.

- Complexity parameters: n and d
- Straightforward algorithm solves it in time $O(n^2 \log d)$. Another straightforward algorithm takes $O(2^d n)$ time.
- $O(n^{2-\frac{1}{O(\log c)}})$ algorithm where $d = c \log n$ by Abboud and Williams, Yu (2015), Chan and Williams (2016).

3-SUM Problem

3-SUM: Given a sequence of integers $x_1, x_2, \ldots, x_n$ where $x_i \in [0, 1, \ldots, d-1]$, do there exist i, j and k such that $x_i + x_j = x_k$?

- Complexity parameters: n and d
- Straightforward algorithm solves it in time $O(n^2 \log d)$.
- $O(n^2 \log \log^2 d / \log^2 d)$ algorithm by Baran, Demaine, Pătrașcu, 2005

Algorithms

Polynomial Method for ORTHOGONAL VECTORS

ORTHOGONAL VECTORS (Bipartite version): Given two sequences $A_1, \ldots, A_n$ and $B_1, \ldots, B_n$ of sets with elements from a universe of size d, do there exist i and j such that $A_i \cap B_j = \varnothing$.

Theorem (Abboud, Williams, and Yu, 2015)

For vectors of dimension $d = c \log n$, the bipartite ORTHOGONAL VECTORS *problems can solved in $n^{2-\frac{1}{O(\log c)}}$ time by a randomized algorithm that is correct with high probability.*

Sketch:

- Find a suitable meta problem which has an improved algorithm over straightforward evaluation
- Reduce the problem to the meta problem by approximating it as a polynomial.
- Optimize the parameters.

Orthogonal Vectors Algorithm: Details

- Partition inputs A and B into $\frac{n}{s}$ blocks $A_1, \ldots, A_p, B_1, \ldots, B_q$ of size s d-dimensional vectors each where $d = c \log n$.
- Let $\mathrm{OV}(x_1, \ldots, x_s, y_1, \ldots, y_s) = 1$ iff $\exists i, j$ x_i and y_j are orthogonal.
- Construct a polynomial P with a small number of monomials to approximate OV. $\mathbf{P}[P$ produces the correct output$] \geq 2/3$.
- For each pair of blocks of inputs A_i and B_j, construct an approximate polynomial $h_{i,j}$ for computing OV on A_j and B_j.
- Evaluate each polynomial $h_{j,j}$ in time $s^2 \texttt{poly} \log(s)$ time using fast matrix-multiplication.
- Choose $s = 2^{\varepsilon \log n / \log d}$
- Overall time complexity of $n^{2 - \frac{1}{O(\log c)}} \texttt{poly} \log n$
- Probability of correctness greater than $2/3$.
- Repeat the experiment $O(\log n)$ times to get the probability of correctness arbitrarily close to 1.

Boolean Expressions to GF(2) Polynomials

- Let $a_1, \ldots, a_s, b_1, \ldots, b_s$ be d-dimensional 0,1 vectors.
- Let $\mathrm{OV}(a_1, \ldots, a_s, b_1, \ldots, b_s) = 1$ if and only if $\exists i, j$ a_i and b_j are orthogonal .

$$\begin{aligned} \mathrm{OV}(a_1, \ldots, a_s, b_1, \ldots, b_s) &= \bigvee_{i,j} \bigwedge_{p} (\bar{a}_{i,p} \vee \bar{b}_{j,p}) \\ &= \bigvee_{i,j} \bigwedge_{p} (1 \oplus a_{i,p} b_{j,p}) \end{aligned}$$

- Approximate $\bigvee$ and $\bigwedge$ by GF(2) polynomials, controlling for the number of monomials.
- Small circuit complexity of OV yields a polynomial approximation with a small number of monomials.

Approximating $\bigwedge$ and $\bigvee$ by GF(2) Polynomials

- Let $g(\vec{x}) = \bigvee_{i=1}^{q} f_i(\vec{x})$
- Let $\tilde{g}(\vec{x}) = \bigoplus_{i=1}^{q} r_i f_i(\vec{x})$ where r_i are randomly chosen 0,1 values.
- $\mathbf{P}_{r_i}[\tilde{g}(\vec{x}) = g(\vec{x})] \geq 1/2$ for any $\vec{x}$.
    - If $g(\vec{x}) = 0$, then $\tilde{g}(\vec{x}) = 0$.
    - If $g(\vec{x}) = 1$, then $\mathbf{P}[\tilde{g}(\vec{x}) = 1] = 1/2$.
- Let $h(\vec{x}) = \bigvee_{j=1}^{t} \tilde{g}_j(\vec{x})$
- $\mathbf{P}[g(\vec{x}) = h(\vec{x})] \geq 1 - 2^{-t}$.
- $h(\vec{x})$ can be written as a degree-t GF(2) polynomial in terms of $\tilde{g}_j$.

Approximating $\bigwedge$ and $\bigvee$ by GF(2) Polynomials

- Approximation of $\bigwedge$ is similar due to De Morgan's law: $g(\vec{x}) = \bigwedge_{i=1}^{q} f_i(\vec{x}) = 1 \oplus (\bigvee_{i=1}^{q} \bar{f}_i(\vec{x}))$
- Note: $\mathrm{OV}(a_1, \ldots, a_s, b_1, \ldots, b_s) = \bigvee_{i,j} \bigwedge_p (1 \oplus a_{i,p} b_{j,p})$
- Approximate the inner $\bigwedge$ with $t = 2\log s$ and the outer $\bigvee$ with $t = 2$ to obtain the final polynomial h.
- Verify $\mathbf{P}[h \text{ is correct}] \geq 2/3$.
- Verify that the number of monomials in h is $s^4(d+1)^{2\log s} < n^{\varepsilon}$ when $s = \varepsilon \log n / \log d$.

Efficient Polynomial Evaluation on a Rectangle of Inputs

Lemma (Williams 2014)

Given a polynomial $P(x_1, \ldots, x_k, y_1, \ldots, y_k)$ over $\mathbb{F}_2$ with at most $m^{0.1}$ monomials and inputs $A = \{a_1, \ldots, a_m\} \subseteq \{0,1\}^k$ and $B = \{b_1, \ldots, b_m\} \subseteq \{0,1\}^k$, P can be evaluated on all pairs $(a_i, b_j) \in A \times B$ in $O(m^2 \texttt{poly}(\log m))$ time.

Proof:

- Reduce the problem to fast rectangular matrix multiplication. by constructing two matrices M_A and M_B from $P = \sum_u m_u$.
- Rows of M_A are indexed by a_i and the columns are indexed by $m_u = \prod_l x_l \prod_p y_p = m_u^x m_u^y$
- $M_A(a_i, m_u) = m_u^x(a_i)$.
- Rows of M_B are indexed by m_u of P and the columns are indexed by b_i.
- $M_B(m_u, b_q) = m_u(b_q)$.
- Observe $M_A M_B(a_i, b_q) = \sum_u m_u^x(a_i) m_u^y(b_q) = P(a_j, b_q)$.

Fast Matrix Multiplication

Lemma (Coppersmith 1982)

For all sufficiently large N, multiplication of an $N \times N^{0.172}$ matrix with an $N^{0.172} \times N$ can be performed in $O(N^2 \log^2 N)$ arithmetic operations.

Polynomial Method for ORTHOGONAL VECTORS— Key Ideas

- Load balancing
- Approximation by polynomials
- Fast matrix multiplication - meta algorithm for evaluating polynomials

Problems in **NP**

Satisfiability Problem

- Input: a formula or circuit F on n Boolean variables, $x_1, x_2, \ldots, x_n$.
- Conjunctive formulas: $\bigwedge_i C_i$ where C_i is a disjunction of literals. $(x_i \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_6 \vee x_8 \vee \bar{x}_3) \wedge \cdots \wedge (\bar{x}_6 \vee x_5 \vee x_2)$.
- $\bar{x}_i$ is the Boolean complement of x_i. $x_i, \bar{x}_i$ are called literals.
- Check if F is satisfiable: does there exist an assignment of Boolean values for $x_1, \ldots, x_n$ which satisfies all the clauses.
- Decidable in $|F|2^n$ time by exhaustive search.
- Can we improve upon the exhaustive search? Can we obtain a $|F|2^{n(1-\mu)}$ bound for $\mu > 0$?
- μ is a called the satisfiability savings. μ can be a function of the parameters of the class of formulas/circuits and n, the number of variables.

Satisfiability for Conjunctive Formulas

- CNF-SAT: Conjunction of disjunctions of literals
- k-SAT: Conjunction of disjunctions of literals where each disjunction contains at most k literals.
- k-SAT is **NP**-complete for $k \geq 3$. How fast can we solve k-SAT for $k \geq 3$? What is the savings over exhaustive search?
- Several algorithmic approaches have been developed.
  - Backtracking algorithms (also known as DPLL algorithms)
  - Local search algorithms
  - Polynomial method
- Best known results are due to PPSZ-style algorithms which themselves is based on PPZ algorithm.‘
  PPZ — Paturi, Pudlák and Zane (1997)
  PPSZ — Paturi, Pudlák, Saks and Zane (1998/2005)
- PPZ is a DPLL - style algorithm with random ordering of variables

PPZ Algorithm

Algorithm **PPZ**:

Let F be a k-CNF and σ a random permutation on variables
for $i = 1, \cdots, n$
  if there is a unit clause for the variable $\sigma(i)$
    then set the variable $\sigma(i)$ so that the clause true
    else set the variable $\sigma(i)$ randomly
  Simplify F
if F is satisfied, output the assignment

Theorem

PPZ finds a satisfying assignment in time $\texttt{poly}(n)2^{n(1-\frac{1}{k})}$ *with constant success probability.*

Isolated Solutions and Critical Clauses

- A satisfying solution for F is isolated if all its distance 1 neighbors are not solutions.
- Let F be a k-CNF and x be an isolated satisfying solution of x.
- For each variable i and isolated solution x, F must have a clause with exactly one true literal corresponding to the variable i at solution x.
- Such clause is called a critical clause for the variable i at the solution x.
- $F = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$
- For the isolated solution $x_1 = 0, x_2 = 0, x_3 = 0$, $F = (0 \vee 1 \vee 0) \wedge (1 \vee 0 \vee 0) \wedge (0 \vee 0 \vee 1) \wedge (1 \vee 1 \vee 1)$

PPZ Analysis

Lemma

Algorithm **PPZ** *outputs x with probability at least $\frac{1}{n}2^{-n+I(x)/k}$ for any satisfying solution x with $I(x)$ many neighbors which are not solutions.*

Proof Sketch:

- E_1 — for at least $I(x)/k$ variables, the critical variable appears as the last variable among the variables in the critical clause
- E_2 — values assigned to the variables in the **for** loop agree with x
- $\mathbf{P}(E_1) \geq 1/n$
- $\mathbf{P}(E_2|E_1) \geq 2^{-n+I(x)/k}$
- $\mathbf{P}(x \text{ is output by } \mathbf{PPZ}) \geq \frac{1}{n}2^{-n+I(x)/k}$

PPZ Analysis

- Let S be the set of satisfying solutions of F.
- For $x \in S$, define $value(x) = 2^{-n+I(x)}$
- Fact: $\sum_{x \in S} value(x) \geq 1$
- $$\mathbf{P}(\exists x \in S x \text{ is output by } \mathbf{PPZ}) \geq \sum_{x \in S} \frac{1}{n} 2^{-n+I(x)/k}$$
$$= \frac{1}{n} 2^{-n+n/k} \sum_{x \in S} 2^{(-n+I(x))/k}$$
$$\geq \frac{1}{n} 2^{-n+n/k}$$

Improved Exponential Time Algorithms for k-SAT

- A lot of effort has gone into improving k-SAT.
  Scheder (2021), Hertli (2012), P, Pudák, Saks, Zane (1998/2005), Schöning (1999), P, Pudlák, Zane (1997), Rolf (2003), Iwama and Tamaki (2004), $\cdots$, Monien and Speckenmeyer (1985)
- Current best approach— PPSZ: PPZ combined with resolution. Nontrivial analysis.
  - 3-SAT — $2^{0.386n}$
  - 4-SAT — $2^{0.554n}$
  - k-SAT — $2^{(1-\mu_k/(k-1))n}$ where $\mu_k \approx 1.6$ for large k.
- Under mild assumptions, $\mu_k \leq 2$ for PPSZ-style algorithms
  Scheder and Talebanfard (2020)

NP-complete Graph Problems

- HAMILTONIAN PATH: Given a graph $G = (V, E)$, is there a hamiltonian path?
- k-COLORABILITY: Given a graph $G = (V, E)$, is G colorable with k or fewer colors?
- COLORABILITY: Given a graph $G = (V, E)$ and an integer k, is G colorable with k or fewer colors?
- MAX INDEPENDENT SET: Given a graph $G = (V, E)$ and an integer k, does G have an independent set of size at least k?
- Complexity parameters: the number of vertices: $n = |V|$, the number of edges: $m = |E|$, the range of edge weights: d

Motivating Questions for Fine-grained Complexity

- Is there an $\varepsilon > 0$ such that ORTHOGONAL VECTORS problem can be computed in time $n^{2-\varepsilon}$ for $d = \omega(\log n)$?
- Is there an $\varepsilon > 0$ such that 3-SUM problem can be computed in time $n^{2-\varepsilon}$ for $d = \omega(\log n)$?
- Is $s_3 = 0$, where $s_k = \inf\{\delta | \exists \quad 2^{\delta n}$ algorithm for k-SAT$\}$?
- Is $s_\infty = 1$, where $s_\infty = \lim_{k\to\infty} s_k$?.

Relationships among Problems

- If ORTHOGONAL VECTORS problem can be solved in time $n^{2-\varepsilon}$ for some $\varepsilon > 0$ for $d = \omega(\log n)$, does there exists $\delta > 0$ such that k-SAT can be solved in time $2^{(1-\delta)n}$ for all k?
- If 3-SAT is solvable in subexponential time, is 4-SAT solvable in subexponential time?
- Do improved algorithms for 3-SAT imply improved algorithms for 3-COLORABILITY or vice versa?

Fine-grained Reductions

- More generally, what 'fine-grained' reductions are possible among these problems?
  - Assume that problem A has a conjectured complexity $T_A(n)$ and problem B $T_B(n)$.
  - Assume that the complexity of A improved to $T_A^{(1-\varepsilon)}(n)$ for $\varepsilon > 0$.
  - Can we infer if there will be an improvement in the complexity of B?
  - Reductions from B to A that enable the transfer of the improvement are fine-grained reductions.

An Obstacle for Developing a Theory of Fine-grained Complexity

- Lack of reductions that preserve the complexity parameter
- In the least, we need reductions that preserve the complexity parameter linearly.

Example: An Obstacle for a Reduction from 3-SAT to 3-COLORABILITY

- If 3-COLORABILITY has a subexponential time ($2^{\varepsilon n}$ for arbitrarily small ε) algorithm, does it imply a subexponential time algorithms for 3-SAT?
- In the standard reduction from 3-SAT of n variables and m clauses to 3-COLORABILITY, we get a graph on $O(n+m)$ vertices and $O(n+m)$ edges.
- Complexity parameter increases polynomially, thus preventing any useful conclusion about 3-SAT.

Subexponential Time

Definition (Subexponential Time)

A problem is computable in time subexponential in the complexity parameter n if there is an effectively computable monotone increasing function $g(n) = \omega(1)$ such that the problem on instance x with complexity parameter n is computable in time $\texttt{poly}(|x|)2^{n/g(n)}$.

Subexponential Time Reductions

Definition (Subexponential Time Reductions)

Let $\mathcal{P}$ and $\mathcal{P}'$ be problems with complexity parameters p and p' respectively. $\mathcal{P}$ is subexponential time reducible to $\mathcal{P}'$ if there exists a collection of reductions $\{R_\varepsilon\}$ such that $\forall \varepsilon > 0$, $\exists c(\varepsilon)$ such that

1. R_ε takes an instance x of $\mathcal{P}$ and outputs instances y_i of $\mathcal{P}'$ for $1 \le i \le 2^{\varepsilon n}$ where $|y_i| \le \texttt{poly}(|x_i|)$ and $p'(y_i) \le c(\varepsilon)p(x)$.
2. $x \in \mathcal{L}(\mathcal{P})$ if and only if $y_i \in \mathcal{L}(\mathcal{P}')$ for some i.
3. R_ε runs in time $\texttt{poly}(|x|)2^{\varepsilon p(x)}$.

Sparsification Lemma

Lemma (Sparsification Lemma)

$\exists$ algorithm A $\forall k \geq 2, \epsilon \in (0,1], \phi \in k\text{-CNF}$ with n variables, $A_{k,\epsilon}(\phi)$ outputs $\phi_1, \ldots, \phi_s \in k\text{-CNF}$ in $2^{\epsilon n}$ time such that

1. *$s \leq 2^{\epsilon n}$; $\mathrm{SAT}(\phi) = \bigcup_i \mathrm{SAT}(\phi_i)$, where $\mathrm{SAT}(\phi)$ is the set of satisfying assignments of ϕ*
2. *$\forall i \in [s]$ each literal occurs $\leq O(\frac{k}{\epsilon})^{3k}$ times in ϕ_i.*

- Branching on variables alone would require setting almost all the variables resulting in a large tree.
- Branch on frequently occurring subclauses rather than just on variables.
- Clause branching results in less information, and as a result the tree does not grow too much.
- To control for the growth of new clauses, start with small clauses and look for longest subclauses with required

Reducing 3-SAT to 3-COLORABILITY under SERF

- Apply Sparsification Lemma to the given 3-CNF ϕ.
- Consider each 3-CNF ϕ_i with linearly many clauses and reduce it to a graph with linearly many vertices.
- Now, a subexponential time algorithm for 3-COLORABILITY implies a subexponential time algorithm for 3-SAT.

Reducing 4-SAT to 3-SAT under Subexponential-time Reductions

- Let $\varepsilon > 0$ be arbitrary.
- Apply Sparsification Lemma to the given 4-CNF ϕ to obtain a disjunction of $2^{\varepsilon n}$ ϕ_i in time $2^{\varepsilon n}$ where each ϕ_i has linearly many clauses.
- Reduce 4-CNF ϕ_i to a 3-SAT formula with only linearly many new variables.
  $(l_1 \vee l_2 \vee l_3 \vee l_4) = \exists y (l_1 \vee l_2 \vee y)(\bar{y} \vee l_3 \vee l_4)$
- Now, a subexponential time algorithm for 3-SAT implies a subexponential time algorithm for ϕ_i.
- Since $\varepsilon > 0$ is arbitrary, ϕ can be solved in subexponential-time.

SNP

- **SNP** — class of properties expressible by a series of second order existential quantifiers, followed by a series of first order universal quantifiers, followed by a basic formula —Papadimitriou and Yannakakis 1991
- **SNP** includes k-SAT and k-COLORABILITY for $k \geq 3$. $\exists S \forall (y_1, \ldots, y_k) \forall (s_1, \ldots, s_k) [R_{(s_1,\ldots,s_k)}(y_1, \ldots, y_k) \implies \wedge_{1 \leq i \leq k} S_{s_i}(y_i)$, where $s_i \in \{+, -\}$ and S is a subset of $[n]$.
- VERTEX COVER,CLIQUE, INDEPENDENT SET and k-SET COVER are in size-constrained **SNP**.
- HAMILTONIAN PATH is **SNP**-hard.

Completeness of 3-SAT in **SNP**

Theorem (IPZ 1997)

3-SAT *admits a subexponential-time algorithm if and only if every problem in (size-constrained)* **SNP** *admits one.*

- Proof Sketch: Show that every problem in **SNP** is strongly many-one reducible to k-SAT for some k. Complexity parameter is the number of Boolean existential quantifiers.
- Reduce k-SAT to the union of subexponentially many linear-size k-SAT using Sparsification Lemma.
- Reduce each linear-size k-SAT to 3-SAT with linearly many variables.

Exponential Time Hypothesis (**ETH**)

- Let $s_k = \inf\{\delta | \exists 2^{\delta n}$ algorithm for k-SAT$\}$;
- 3-SAT has a subexponential time algorithm $\implies s_k = 0$ for all k and $s_\infty = 0$. Moreover, all problems in **SNP** have subexponential time algorithms.
- Our plan is to make progress by assuming this statement
- Exponential Time Hypothesis (**ETH**) — $s_3 > 0$

Explanatory Burden of **ETH**

- We have very little understanding of exponential time algorithms.
- For **ETH** to be useful,
  - it must be able to provide an explanation for the known complexities of various problems,
  - ideally, by providing lower bounds that match the upper bounds from the best known algorithms.
- **ETH** will be useful if it helps factor out the essential difficulty of dealing with exponential time algorithms for **NP**-complete problems.

Explanatory Value of **ETH** — I

- All the following results assume **ETH**.
- None of the problems in (size-constrained) **SNP** have a subexponential time algorithm
- Furthermore, **SNP**-hard problems such as HAMILTONIAN PATH cannot have a subexponential time algorithm.

Explanatory Value of **ETH** — II

- We follow the nice summary provided by Lokshtanov, Marx and Saurabh (2011).
- Subexponential time lower bounds: There is no $2^{o(\sqrt{n})}$ algorithm for VERTEX COVER, 3-COLORABILITY, and HAMILTONIAN PATH for planar graphs.
- Lower bounds for FPT problems: There is no $2^{o(k)}n^{O(1)}$ algorithm to decide whether the graph has a vertex cover of size at most k.
  Similar results hold for the problems FEEDBACK VERTEX SET and LONGEST PATH. Cai and Juedes (2003)
- Lower bounds for $W[1]$-complete problems: There is no $f(k)n^{o(k)}$ algorithm for CLIQUE or INDEPENDENT SET. Chen, Chor, Fellows, Huang, Juedes, Kanj, and Xia (2005, 2006)

SETH — Strong Exponential Time Hypothesis

Theorem (IP, 1999)

If **ETH** *is true, s_k increases infinitely often*

- Let $s_\infty = \lim_{k\to\infty} s_k$.
- Conjecture:
  Strong Exponential Time Hypothesis (**SETH**): $s_\infty = 1$

SETH and Its Equivalent Statements

Theorem

The following statements are equivalent:

- $\forall \varepsilon < 1$, $\exists k$, k-SAT, *the satisfiability problems for n-variable k-CNF formulas, cannot be computed in time $O(2^{\varepsilon n})$ time.*
- $\forall \varepsilon < 1$, $\exists k$, k-HITTING SET, *the* HITTING SET *problem for set systems over $[n]$ with sets of size at most k, cannot be computed in time $O(2^{\varepsilon n})$ time.*
- $\forall \varepsilon < 1$, $\exists k$, k-SET SPLITTING, *the* SET SPLITTING *problem for set systems over $[n]$ with sets of size at most k, cannot be computed in time $O(2^{\varepsilon n})$ time.*

— Cygan, Dell, Lokshtanov, Marx, Nederlof, Okamoto, P, Saurabh, Wahlstrom, 2012

Explanatory Power of **SETH**

- Under **SETH**, we will show that there is no $\varepsilon > 0$ such that ORTHOGONAL VECTORSproblem has a $n^{2-\varepsilon}$ algorithm for $d = \omega(\log n)$.
- Reduce k-SAT to ORTHOGONAL VECTORS
- Each clause of a k-CNF corresponds to a coordinate of the vector.
- Assume that the variables come in two colors. For each color and for each setting of variables of the color, we will derive a vector from the set of clauses.
- Let $C_i = (x_1 \vee \bar{x}_2 \vee y_1 \vee \vee y_2)$. α is a setting for the x variables and β a setting for the y variables.
- Define $a_i(\alpha) = \neg(x_1 \vee \bar{x}_2)(\alpha)$ and $b_i = \neg(x_1 \vee \bar{x}_2)(\beta)$
- C_i is false under (α, β) if and only if $a_i(\alpha)b_i(\beta) = 1$

Reducing k-SAT to ORTHOGONAL VECTORS

- Assume that the ORTHOGONAL VECTORS problem can be solved in time $n^{2-\varepsilon}$ for $\varepsilon > 0$ for $d = \omega(\log n)$.
- Let $\varepsilon' = \varepsilon/3$ and ϕ be a k-CNF with n variables for $k > 0$.
- Sparsify ϕ in $2^{\varepsilon' n}$ time into $2^{\varepsilon' n}$ many k-SAT instances ϕ_i with at most $c_{\varepsilon'} n$ many clauses.
- For each ϕ_i, construct two families L and R of sets which are subsets of a universe of size $c_{\varepsilon'} n$ where $|L| = |R| = N = 2^{n/2}$.
- ϕ_i is satisfiable if and only if there is a pair of sets $A \in L$ and $B \in R$ such that $A \cap B = \varnothing$.
- Total time for solving the satisfiability of ϕ is $2^{\varepsilon' n} + N^{2-\epsilon} 2^{\varepsilon' n} \approx 2^{(1-\varepsilon/6)n}$
- Since k is arbitrary, this implies that **SETH** is false.
- Conclusion: If **SETH**, there is no $\epsilon > 0$ such that ORTHOGONAL VECTORS problem can be solved in time $n^{2-\varepsilon}$ for a universe of size $\omega(\log n)$.

Conclusions

- If Orthogonal Vectors problem can be solved in time $n^{2-\varepsilon}$ for some $\varepsilon > 0$ for $d = \omega(\log n)$, does there exists $\delta > 0$ such that k-SAT can be solved in time $2^{(1-\delta)n}$ for all k?
  Yes
- If 3-SAT is solvable in subexponential time, is 4-SAT solvable in subexponential time?
  Yes
- Do improved algorithms for 3-SAT imply improved algorithms for 3-Colorability or vice versa?
  Yes (due to Sparsification Lemma)
- Is there an $\varepsilon > 0$ such that Orthogonal Vectors problem can be computed in time $n^{2-\varepsilon}$ for $d = \omega(\log n)$?
  No, if **SETH** is true.

Open Problems

- Prove or disprove **ETH**
- Prove or disprove **SETH**
- Assuming **ETH** or other suitable assumption, prove
  - a specific lower bound on s_3
  - $s_\infty = 1$ (**SETH**)
- Assuming **SETH**, can we prove a 2^n lower bound on COLORABILITY?

Thank You