On the Strong Exponential Time Hypothesis

Ryan Williams Stanford

On the Strong Exponential Time Hypothesis

~ 1%

Ryan Williams Stanford

Introduction

Satisfiability (SAT) needs no introduction

OK, I'm Not That Lazy

- SAT = { satisfiable boolean formulas in CNF }
- k-SAT = { SAT where all clauses have at most k literals}

Two measures of the size of a formula:

- n = number of variables
- m = number of clauses

Best known worst-case algorithms:

- SAT: **$2^{n - n/O(\log m/n)}$ poly(m) $\approx 2^{n - o(n)}$ time** [CIP'06]
- k-SAT: **$2^{n - n/k}$ poly(m) time** [PPZ'97]

Can we improve the exponents?

- **Is k-SAT always in $2^{\delta n}$ poly(m) time for a *universal* δ < 1 ?**

  Let **SETH** be the hypothesis that the answer is "no"

- **Is 3-SAT in $2^{\epsilon n}$ time, for every ϵ > 0?**

  Let **ETH** be the hypothesis that the answer is "no"

Theorem: SETH implies ETH

These hypotheses have been very useful in recent years.

For **many *polynomial time* problems**, improving the best known algorithms, even slightly, implies **¬SETH** or **¬ETH**

***Contrapositive**: If SAT needs exponential time, get strong polynomial lower bounds for interesting problems.*

The Point(s) of This Talk

I believe SETH is false.

My belief is the minority opinion.

(But the chances I'll be proved wrong in my lifetime are nil!)

Even if SETH is true, my belief in the opposite has led me to many ideas I'd have never found otherwise.

Will tell you about some of these ideas.

A Few Years Ago in Ithaca, NY

- **1998:** Thought I proved P=NP (3SAT in polytime)
- **1999:** Learned of Schoening's local search algorithm for k-SAT [FOCS'99] from J. Kleinberg
- **July '01:** Submitted a paper to SODA'02 on solving QBF
- **October '01:** Paper got in! G. Woeginger saw it. He was writing a survey about exact algorithms, sent a draft to me for comment
  - **Open Problem 4.4:** **Design an exact algorithm for Max-Cut with time complexity $O^*(c^n)$ for some $c < 2$**
  - **Open Problem 7.4:** **Assuming ETH, obtain evidence for SETH**
  - **I became obsessed with solving Woeginger's open problems…**

A Few Years Ago in Pittsburgh, PA

- **2002:** Became enamored of the $O(n^{\omega})$ time algorithm for finding a triangle in an n-node graph of Itai and Rodeh

Thm [IR78]
If m by m matrices can be multiplied in O(m^{ω}) additions and multiplications, then 3-CLIQUE on n-node graphs is in O(n^{ω}) time.

Proof:
Let A be the n by n adjacency matrix of graph G, and let B = A·A

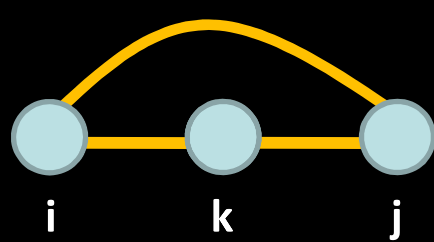
There is a 3-Clique in G

$\Leftrightarrow$

There are i, j=1,…,n such that A[i,j] $\neq$ 0 and B[i,j] $\neq$ 0

$B[i,j] = \sum_k A[i,k] \cdot A[k,j] \neq 0$

$A[i,j] \neq 0$



In fact, can count the 3-cliques!

A Few Years Ago in Pittsburgh, PA

- **2002:** Became enamored of the $O(n^\omega)$ time algorithm for finding a triangle in an n-node graph of Itai and Rodeh

  If we think of $n^3 = 2^k$ for some k, then $n^\omega = O\left(1.74^k\right)$

- Summer '03: **IDEA:** express CNF-SAT on k variables as an instance of triangle detection on $n = 2^{k/3}$ nodes.

  FAILED! Edges can only encode so much!

- Fall '03: Edges *can* encode constraints on two variables
  - **Max-Cut on n nodes is in $O(1.74^n)$ time (Open Problem 4.4)**
  - **Max-2Sat on n** variables **is in $O(1.74^n)$ time**

  Appeared in ICALP'04, generalized in my PhD thesis [2007]

- **2005-07:** Found *other* polytime problems whose faster solution would refute SETH **[Appeared in SODA'10]**

  (But FAILED to solve them faster)

Some Results [PW'10]

- ***k-Dominating Set*:** Given a graph (V,E), find a k-set of nodes S such that $S \cup N(S) = V$.

  Solvable in $n^{k+o(1)}$ time [EG'04]

  If solvable in $O(n^{k-\epsilon})$ time for some $k > 2$, $\epsilon > 0$ $\Longrightarrow$ **¬SETH**

- ***2SAT2*:** Given a 2CNF on $n^{1+o(1)}$ clauses with two extra clauses of arbitrary length, is it satisfiable? Solvable in $n^{2+o(1)}$ time

  If solvable in $O(n^{2-\epsilon})$ time for some $\epsilon > 0$ $\Longrightarrow$ **¬SETH**

- **d-SUM:** Given n numbers, are there d that sum to zero?

  ETH $\Longrightarrow$ d-SUM requires $n^{\Omega(d)}$ time

- ***OV:*** Given a set of n binary d-dimensional vectors, are there two with inner product equal to zero?

  If solvable in $n^{2-\epsilon} 2^{o(d)}$ time for some $\epsilon > 0$ $\Longrightarrow$ **¬SETH**

Faster K-Dominating Set $\Rightarrow$ $\neg$SETH

Theorem. k-Dominating Set in **O($n^{k-\epsilon}$)** time
$\Rightarrow$ SAT in **$2^{(1-\epsilon/k)n}$ poly(m)** time

Proof. Given F with n variables and m clauses, we construct a graph G on **O(k $2^{n/k}$ + m)** nodes, where

G has a dominating set of size k $\Leftrightarrow$ F is satisfiable

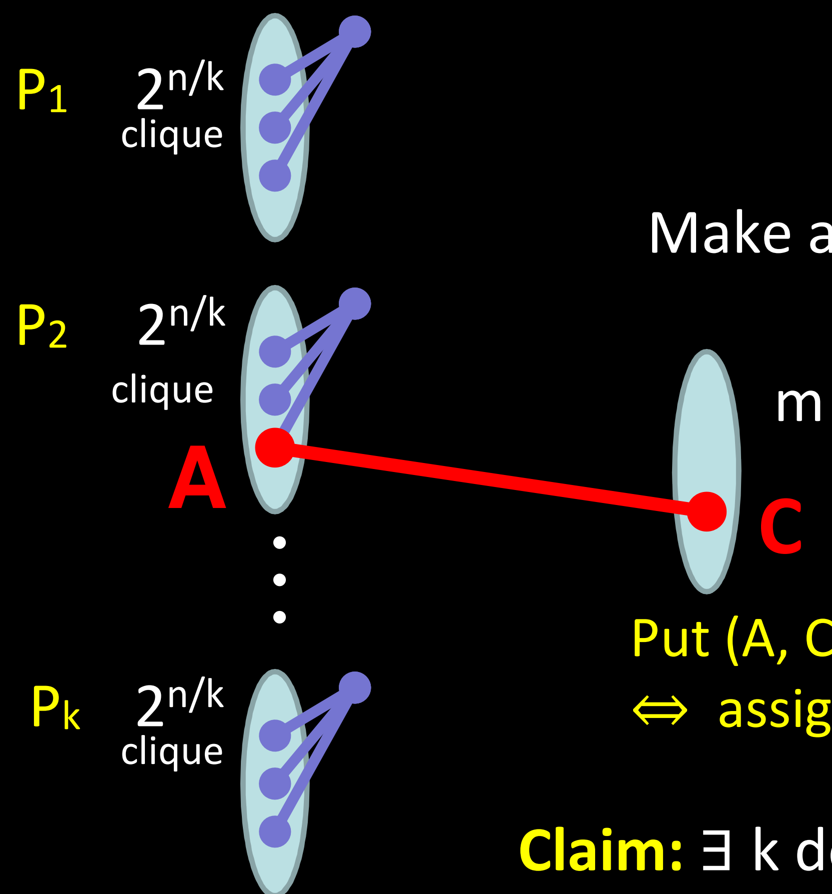
Note: Theorem shows that even tiny improvements in solving k-DS imply tiny SAT improvements

We construct a graph G on **$O(k\, 2^{n/k} + m)$** nodes, where

G has a dominating set of size k $\Leftrightarrow$ F is satisfiable

Split n vars into k parts $P_1, \ldots, P_k$ with $\leq n/k+1$ variables each.

Make nodes for *all* assignments to the variables in a part.



Make a node for every clause in F

Put (A, C) as an edge

$\Leftrightarrow$ assignment A satisfies clause C

Claim: $\exists$ k dom set in G $\Leftrightarrow$ F is SAT

Faster O.V. $\Rightarrow$ ¬SETH

Theorem. **Orthogonal Vectors with n vectors and d dimensions in $n^{2-\epsilon} 2^{o(d)}$ time**

$\Rightarrow$ SAT in $2^{(1-\epsilon/2)n} 2^{o(m)}$ time

[Sparsification Lemma] $\Rightarrow$ k-SAT in $2^{(1-\epsilon/2)n}$ time, for all k

Proof Sketch. Given F with n variables and m clauses, we construct a set S of 2n vectors in m+2 dimensions s.t.

S has an orthogonal pair $\Leftrightarrow$ F is satisfiable

Split n variables into two parts P_1, P_2 with $\leq n/2$ variables each.

Make vectors for all assignments to the variables in a part.

For all assignments A in P_1 define a vector v_A

$v_A[i] := 1$ iff A doesn't satisfy ith clause of F, $v_A[m+1] := 1$, $v_A[m+2] := 0$

For all assignments A in P_2

$v_A[i] := 1$ iff A doesn't satisfy ith clause of F, $v_A[m+1] := 0$, $v_A[m+2] := 1$

A Few Years Ago in San Jose, CA

- **Summer 2010:** Another approach to solving CNF-SAT

  IDEA: Try to CNF-SAT by expressing it as some multivariate polynomial problem, then using algebraic algorithms like Fast Fourier Transform

  FAILED! CNF-SAT algs which were much *worse* than [CIP06]

  – But they worked not only for CNF, but also AC0, and ACC0…

- **Fall '10:** SAT of ACC0 circuits is in **$O(2^n/n^{\log n})$ time**

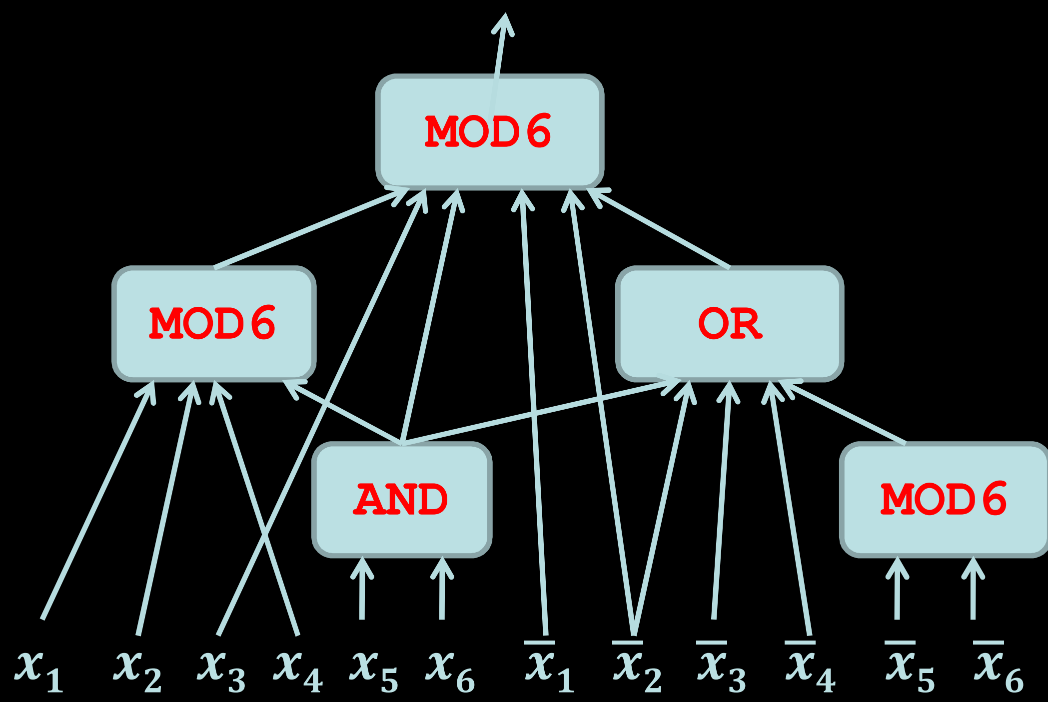
- **Next day:** Proved that this implies NEXP not in ACC0.

  (Was a notorious open problem in circuit complexity)

ACC-SAT Algorithm

- **ACC-SAT** Constant-depth AND/OR/NOT/MODm
  MODm($x_1, \ldots, x_t$) = 1 iff $\sum_i x_i$ is divisible by m

[W '11] ACC-SAT is in $2^{n - n^e}$ time for circuits of size $2^{n^{o(1)}}$



Algorithm for ACC-SAT [W'11]

The ingredients:

1. **A known representation of ACC via polynomials**
   [Yao '90, Beigel-Tarui'94] Every ACC function
   $\mathbf{f : \{0,1\}^n \rightarrow \{0,1\}}$ can be put in the form

   $$f(x_1,\ldots,x_n) = g(h(x_1,\ldots,x_n))$$

   - **h** is a multilinear polynomial with K monomials,
   and over all 0-1 assignments, $\mathbf{h(x_1,\ldots,x_n) \in \{0,\ldots,K\}}$
   - **K** is not "too large" *(quasipolynomial in circuit size)*
   - $\mathbf{g : \{0,\ldots,K\} \rightarrow \{0,1\}}$ can be arbitrary.

2. **Fast Fourier Transform for multilinear polynomials to quickly evaluate h on all its possible assignments**

Fast Multipoint Evaluation

Theorem: Given the 2^n coefficients of a multilinear polynomial **h** in **n** variables, the value **h(x)** can be computed on all points $\mathbf{x} \in \{0,1\}^n$ in 2^n **poly(n)** time.

Can write **$h(x_1, \ldots, x_n) = x_1\, h_1(x_2, \ldots, x_n) + h_2(x_2, \ldots, x_n)$**

Want a 2^n table T that contains the value of h on all 2^n points.

Algorithm: If n = 1 then return T = [h(0), h(1)]

Recursively compute the 2^{n-1} table T_1 for the values of h_1, and the 2^{n-1} table T_2 for the values of h_2

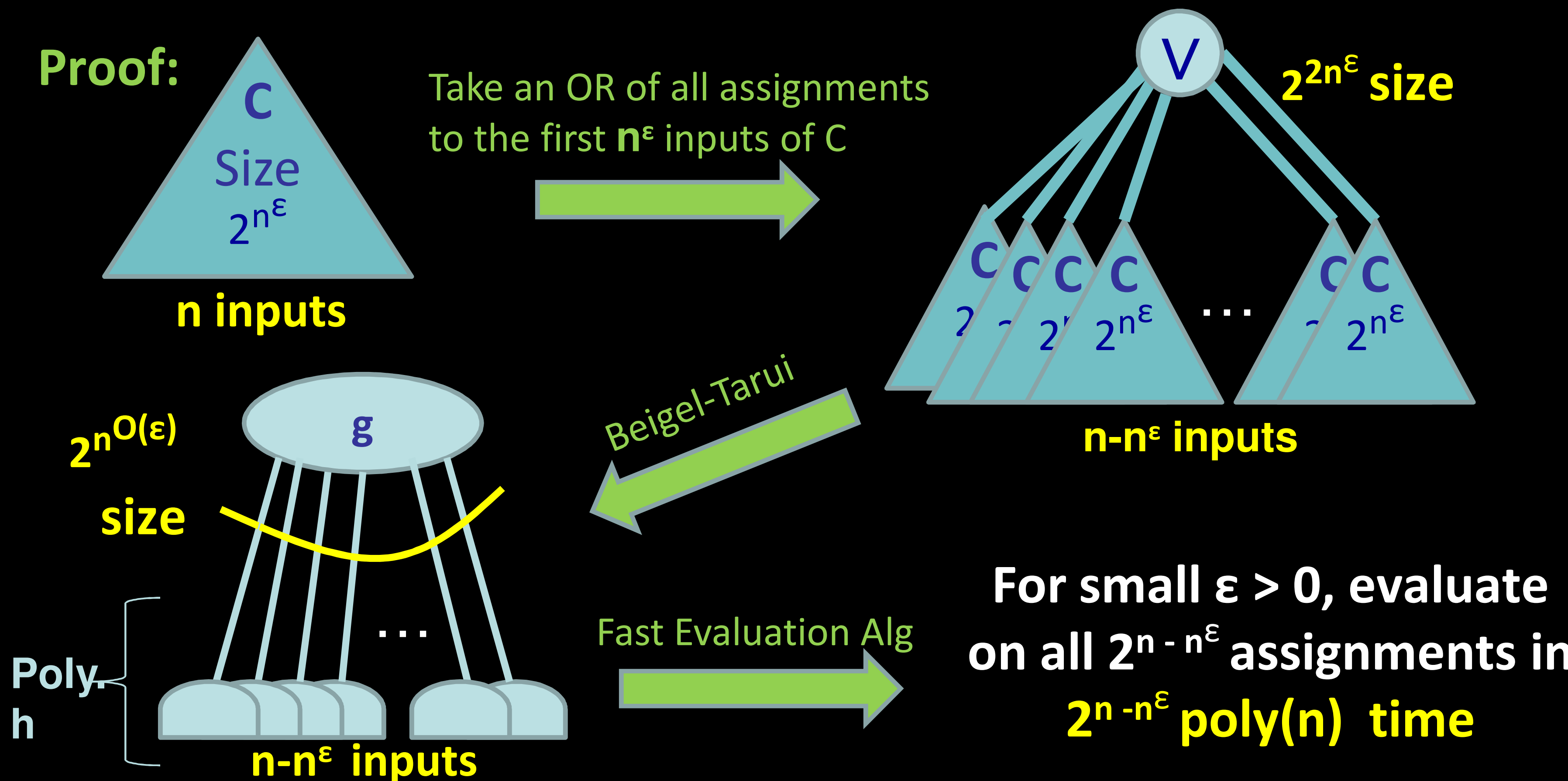
Return the table $T = (T_2)(T_1 + T_2)$ of 2^n entries

Running time has the recurrence **$R(2^n) \leq 2\,R(2^{n-1}) + 2^n\, \mathrm{poly}(n)$**

Corollary: We can compute g of h on all $x \in \{0,1\}^n$ in only 2^n poly(n) time

ACC Satisfiability Algorithm

Theorem For all d, there's an $\varepsilon > 0$ such that ACC-SAT with depth d, n inputs, 2^{n^ε} size can be solved in $2^{n - \Omega(n^\varepsilon)}$ time



A Year Ago in Stanford, CA

- **Summer 2013:** Yet another attack on SETH

  IDEA: Try to solve O.V. in sub-quadratic time…

  By applying a polynomial reduction to a circuit expressing a group of orthogonal vector queries,
  then use matrix multiply/FFT

  FAILED!

  But later [SODA'15] got CNF-SAT algorithms as good as [CIP06]

  – However, the idea there could be used to compute *another* kind of inner product instead…

All-Pairs Shortest Paths (APSP)

Let $u, v \in \mathbb{N}^d$. Define the min-plus inner product of u and v to be

$$(u \circ v) := \min_k (u_k + v_k)$$

Theorem [Fischer-Meyer, Munro '71]

To solve APSP, it suffices to compute the *min-plus matrix product of* $A, B \in \mathbb{R}^{n \times n}$

$$(A \circ B)[i, j] = \min_k (A[i, k] + B[k, j])$$

Key Idea 1: ***Min-plus inner products*** **are EASY wrt circuit complexity!**

Computable with AC0 circuits: *constant depth, AND/OR/NOT, polynomial size*

Key Idea 2: **EASY inner products can be reduced to polynomials over $\mathbb{F}_2$**

[Razborov-Smolensky'87]

Randomized reduction from AC0 circuits to polylog-degree polynomials over $\mathbb{F}_2$:

for every input, the probability the polynomial agrees with the circuit is > ¾.

Key Idea 3: **Polynomials can be eff. evaluated on many pairs of points**

[Coppersmith'82] (Very) fast rectangular matrix multiplication

All-Pairs Shortest Paths (APSP)

Theorem 1: There is a randomized algorithm for APSP on n-node weighted graphs running in $\frac{n^3}{2^{L(n)}}$ time where $L(n) \geq \Omega(\log n)^{1/2}$.

Was open for 40 years whether $\frac{n^3}{log^c(n)}$ time was possible for every constant c.

Open Problems

- Give more evidence that **SETH** is true?
  - **Prove that ETH is equivalent to SETH?**
    [Cygan et al. CCC'12] Equivalences
  - **Prove that ¬SETH implies an unlikely collapse of complexity classes?**
- Give more evidence that **SETH** is false? ☺

(Make future talks more satisfying?)

Thank you!