Genesis of **ETH** and **SETH**

Mohan Paturi

University of California, San Diego

19th Max Planck Advanced Course on the Foundations of Computer Science
August 2018, Saarbrücken, Germany

Outline

Central Role of Satisfiability

- Satisfiability problems seem to play a central role in complexity theory due to their expressive power.
    - as complete problems
    - in terms of connection to lower bounds
- Equally central in fine-grained complexity.
    - Any problem in **NP** can be reduced to $\textsc{Circuit Sat}$ preserving the natural complexity parameter.
    - Satisfiability conjectures are able to explain why we cannot improve algorithms for a wide variety of problems.
    - Reduction from the satisfiability problem of a more general circuit model to a given problem demonstrates the greater hardness of the problem.

Plan

- Satisfiability algorithms; PPZ algorithm
- Fine-grained reducibilities
- Completeness of 3-SAT under subexponential reductions
- Satisfiability conjectures and their explanatory power

Satisfiability Problems

- Input: a formula or circuit F on n variables.
- Check if F is satisfiable
- Examples for F : k-CNF, CNF, $\mathbf{AC}^0$ circuit, $\mathbf{NC}^1$ circuit, polynomial size circuit, or a formula
- Decidable in $|F|2^n$ time.
- 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.

Improved Algorithms for Circuit Sat

Circuit Sat — split and list, random restrictions, dynamic programming, algebraization, matrix multiplication
Impagliazzo, P, William (2012), Williams (2011), Santhanam (2011), Tamaki and Seto (2012), Impagliazzo, P, and Schneider (2013), Lokshtanov, Mikhailin, P, Pudlák (2018)

- **AC**^{0} Satisfiability for circuits of size cn and depth d — $2^{n(1-1/O(c^d))}$
- **ACC** Satisfiability — $2^{n-\Omega(n^\varepsilon)}$
- Formula Satisfiability for formulas of size cn — $2^{n(1-1/O(c^2))}$
- Formula Satisfiability for formulas of size cn over the full binary basis — $2^{n(1-1/O(2^{c^2}))}$
- Depth-2 Threshold Circuit Satisfiability for circuits with cn wires — $2^{n(1-1/O(c^{c^2}))}$
- Formula Satisfiability for formulas of size cn — $2^{n(1-1/O(c))}$
- Satisfiability for circuits of size cn and bounded treewidth ω — $2^{n(1-1/O(c\omega 4^\omega))}$

Improved Exponential Time Algorithms for k-SAT

- k-SAT, number of variables as the complexity parameter — backtracking and local search
  Hertli (2012), PPSZ (1998/2005), Schöning (1999), PPZ (1997), Rolf (2003), Iwama and Tamaki (2004), $\cdots$, Monien and Speckenmeyer (1985)
  - 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.

Approaches to k-SAT

- Backtracking algorithms (also known as DPLL algorithms)
- Local search algorithms
- Polynomial method
- Earlier (to 1997) results showed that μ is $\Omega(1/2^k)$
- PPZ is a DPLL-style algorithm with random ordering of variables

PPZ Algorithm

Algorithm **PPZ**:

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

Theorem

There is a randomized algorithm for k-SAT that finds a satisfying assignment in time $\texttt{poly}(n)2^{n(1-\frac{1}{k})}$ *with constant success probability.*

Isolated Solutions

- A satisfying solution for F is isolated if all its distance 1 neighbors are not solutions.
- What is the maximum number of isolated solutions for a k-CNF?
- We show that this number is at most $2^{n(1-1/k)}$

Critical Clauses

- 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.

Compressing Isolated Satisfying Solutions

- Let F be a k-CNF and σ a permutation of $\{1, \cdots, n\}$.
- Let $x \in \{0,1\}^n$ be an isolated satisfying solution of F
- Compression Function F_σ:
  1. Permute the bits of x according to σ
  2. For each i, delete the i'th bit of x if all other variables in a critical clause $C_{x,\sigma(i)}$ (for the variable $\sigma(i)$ at x) occur before the variable $\sigma(i)$ in the order σ.
  3. $F_\sigma(x)$ is the resulting compressed string.

F_σ is Lossless

- We can recover x from $y = F_\sigma(x)$, F, and σ. Decompression Algorithm:

```
1  $F_1 = F$
2  **for** $i = 1, \cdots, n$
3    **if** $F_i$ has a clause of length one with the variable $\sigma(i)$,
4      **then** set the variable $\sigma(i)$ so that the clause is true
5      **else** set the variable $\sigma(i)$ to the next unused bit of $y$.
6    $F_{i+1}$ = substitute for $\sigma(i)$ in $F$ and simplify
```

Satisfiability Coding Lemma

Lemma (Satisfiability Coding Lemma)

If x is an isolated solution of a k-CNF F, then its average (over all permutations σ) compressed length $|F_\sigma(x)|$ is at most $n(1-1/k)$.

Proof Sketch: For each variable i with a critical clause at x, the probability (under a random permutation) i appears last among all the variables in its critical clause is at least $1/k$.

The compression algorithm deletes n/k bits on average.

Maximum Number of Isolated Solutions

Lemma

A k-CNF can have at most $2^{n(1-1/k)}$ isolated solutions.

Proof Sketch:

- For every isolated solution, the average (over permutations) compressed length is at most $n - n/k$
- There exists a permutation such that the average (over all isolated solutions) compressed length is at most $n - n/k$.
- Hence, the number of isolated solutions is at most $2^{n(1-1/k)}$ using a convexity argument.

Fact

If $\Phi : S \rightarrow \{0,1\}^*$ is a prefix-free encoding (one-to-one function) with average code length l, the $|S| \leq 2^l$.

Lower Bounds for Parity

Theorem

Computing the parity function requires $2^{n/k}$ size $\Sigma\Pi\Sigma_k$ circuits.

Theorem

Computing the parity function requires $\Omega(n^{1/4}2^{\sqrt{n}})$ size depth-3 circuits.

k-SAT Algorithm

Algorithm **PPZ**:

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

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: $\sum_{x \in S} value(x) \geq 1$
- $$\begin{aligned} \mathbf{P}(x \text{ is output by } \mathbf{PPZ}) &\geq \sum_{x \in S} \frac{1}{n} 2^{-n+I(x)/k} \\ &= \frac{1}{n} 2^{-n+n/k} \sum_{x \in S} 2^{(-n+I(x))/k} \\ &\geq \frac{1}{n} 2^{-n+n/k} \end{aligned}$$

Dense Case

Theorem

If $S \neq \emptyset$ is the set of satisfying solutions of a k-CNF F, then **PPZ** *finds a satisfying assignment with probability at least* $\frac{1}{n}(\frac{2^n}{|S|})^{(1-1/k)}$

Proof Sketch: Use the edge isoperimetric inequality for the hypercube to conclude that among all sets $S \subseteq \{0,1\}^n$ of a given size, the subcube of dimension $\log|S|$ minimizes the number of edges between S and $\bar{S}$.

Towards a Theory of Fine-Grained Complexity — Motivating Questions

- Can the improvements for 3-SAT extend to arbitrarily small exponents?
- Is 3-SAT solvable in subexponential-time?
- How about 3-COLORABILITY?
  Do improved algorithms for 3-SAT imply improved algorithms for 3-COLORABILITY or vice versa?

An Obstacle for Developing a Theory of Exact 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 $1 \leq i \leq 2^{\varepsilon n}$ where $|y_i| \leq \texttt{poly}(|x_i|)$ and $p'(y_i) \leq c(\varepsilon)p(x)$.
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 $\texttt{poly}(|x|)2^{\varepsilon p(x)}$.

Worst-case Instances of MAX INDEPENDENT SET

Theorem (Johnson and Szegedy, 1998)

MAX INDEPENDENT SET *problem has a subexponential time algorithm (in the number of vertices) iff* MAX INDEPENDENT SET *problem when restricted to graphs of degree at most three has one.*

In other words, graphs with maximum degree three are the worst-case instances for the MAX INDEPENDENT SET problem up to subexponential time reductions.

Proof Sketch for Johnson/Szegedy Theorem

- Let d be large enough. Given a graph G on n vertices, execute the following backtracking algorithm:
    1. As long as G has a vertex of degree more than d, select a vertex v such that $deg(v) > d$.
    2. Solve the instances $G - v$ (major branch) and $G - N[v]$ (minor branch) where $N[v]$ is the neighbourhood of v in G including v.
- Transform each G_i output by the previous algorithm to G_i' of degree at most 3 where $n(G_i') \leq (2d-1)n(G_i)$ and such that a maximum independent set of G_i can be recovered from a maximum independent set of G_i' in linear time.
- Number of root-leaf paths with i minor branches is at most $\binom{n}{i}$ since the maximum path length is at most n.
- $i \leq n/(d+1)$. $\binom{n}{n/(d+1)} \approx 2^{h(1/(d+1))n}$ where h is the binary entropy function.

Johnson/Szegedy Continued

- Let $g(n) = \omega(1)$ be a monotone function such that that maximum independent set can be computed in time $\texttt{poly}(|G|)2^{n/g(n)}$ on graphs G of degree at most 3 with n vertices.
- We reduced G on n vertices to at most $2^{h(1/(d+1))n}$ many graph instances each with at most $(2d-1)n$ vertices in time $2^{h(1/(d+1))n}$.
- Total time for solving the $\textsc{Max Independent Set}$ problem is bounded by

$$2^{h(\frac{1}{(d+1)})n} + 2^{h(\frac{1}{(d+1)})n} 2^{\frac{(2d-1)n}{g((2d-1)n)}} \approx 2^{n(h(\frac{1}{(d+1)}) + \frac{(2d-1)}{g((2d-1)n)})}$$

- Choose d large enough so that the exponent is at most εn for all sufficiently large n.

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. *$s \leq 2^{\epsilon n}$; $\textsc{Sat}(\phi) = \bigcup_i \textsc{Sat}(\phi_i)$, where $\textsc{Sat}(\phi)$ 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

Reduced Clause Sets

- A *k-clause* is a clause of size (exactly) k. A k-CNF ϕ is a set of clauses each of size $\leq k$.
- View each clause as a set of literals.
- ϕ is reduced iff no clause is a subset of any other. The reduction of $\phi = \textbf{red}\ \phi = \{\subseteq$-minimal elements of $\phi\}$.

Frequency Parameters

- Define

$$\begin{aligned}
\beta_1 &= 2, \theta_0 = 2, \\
\beta_c &= \sum_{h=1}^{c-1} 4\alpha\beta_{c-h}\beta_h \text{ for } c \geq 2 \\
\theta_c &= \beta_c\alpha \text{ for } c \geq 1, \text{ and} \\
\alpha &= \frac{2(k-1)^2}{\varepsilon} \lg \frac{32(k-1)^2}{\varepsilon}.
\end{aligned}$$

- Solving the recurrence, we get

$$\begin{aligned}
\beta_i &\leq 4(32\alpha)^{i-1} \text{ for } i \geq 2 \\
\theta_i &= \alpha\beta_i \leq 4(32)^{i-1}\alpha^i \text{ for } i \geq 1.
\end{aligned}$$

- Define $\beta = \sum_{i=1}^{k-1} \beta_i \leq 4(32\alpha)^k$

Sunflowers, Hearts, and Petals

- ϕ_c — set of c-clauses of ϕ;
- Sunflower $\mathcal{F}$ of c-clauses — a collection of distinct c-clauses of size $\geq \theta_{c-h}$ that share a common subset H of size $h \geq 1$
- Sunflowers consist of clauses of the same size.
- Petals of a sunflower $\mathcal{F}$: $\{C - H\}$ where C is a clause of $\mathcal{F}$ and H its heart.
- All petals have the same size and they need not be disjoint.

Sparsification Algorithm

```
$A_{k,\epsilon}(\phi \in$ k-CNF$)$
  $\phi \leftarrow$ **red** $\phi$
  if $\exists$ a sunflower {
    select a sunflower consisting of clauses of the smallest size $c$ and among them one with a heart $H$ of the largest size
    $P \leftarrow \{C - H \mid H \subseteq C \in \phi_c\}$ /* set of petals */
    /* branch: if we set $H$ to 1, we call this a heart branch */
    /* if we set $H$ to 0, we call this a petal branch */
    $A_{k,\epsilon}(\phi \cup \{H\})$
    $A_{k,\epsilon}(\phi \cup P)$
  }
  output $\phi$    /* $\phi$ is sparsified */
```

Figure: Sparsification Algorithm

Analysis of Sparsification Algorithm

- Consider the binary tree T generated by the recursive calls of the algorithm.
- Each node is associated with a reduced k-CNF.
- Each leaf is sparse: No literal occurs with frequency more than $O(\frac{k}{\varepsilon})^{3k}$. If a literal (heart of size 1) with frequency θ_{c-1} among c-clauses, we would have branched.
- Goal: to bound the number of leaves of T.
- Plan:
  1. Bound the number of root-leaf paths.
  2. Bound the maximum length (βn) of any path
  3. Bound the maximum number ($\frac{(k-1)n}{\alpha}$) of petal branches along any path.
  4. Conclude that the number of leaves $\leq \binom{\beta n}{\frac{(k-1)n}{\alpha}} \leq 2^{\varepsilon n}$.

Bounding the Max Length of a Path in T

- A clause C is new at node u in T if C is either a petal or a heart in the branching at the parent of u.
- In other words, C is new at u if the parent of u introduced C along the branch leading to u.
- Consider the new clauses introduced along a path. Each branch introduces at least one new clause.
- Maximum path length is bounded by the maximum number of new clauses introduced along any path.
  1. Show that the number of new c-clauses $\leq \beta_c n$
  2. Conclude that the maximum path length $\leq \beta n = \sum_{c=1}^{k-1} \beta_c n$

Bounding the Number of New Clauses

- Petals are compact: no subclause of length h occurs more than $\theta_{c-h} - 1$ among petals of size c.
- Indirect argument to count new clauses: new clauses either remain till the leaf or get eliminated by another new clause
- Number of new clauses along a path $\leq$ number of clauses eliminated $+$ number remaining at the leaf
- Number of new clauses of length c at any leaf is at most $2n\theta_{c-1}/c$ since no variable occurs with frequency more than $\theta_{c-1} - 1$.

Bounding the Number of Eliminated New Clauses – I

- Only new clauses can eliminate new clauses.
- Any clauses eliminated by a new clause are eliminated at the time of its introduction.
- Argue that a new clause cannot eliminate many new clauses.

Bounding the Number of Eliminated New Clauses – II

- We will show that a a newly introduced h-clause can eliminate at most $2(\theta_{c-h} - 1)$ new c-clauses.
- Definition: A set of clauses of uniform length c is sparsified if no subclause of length $h \geq 1$ occurs with frequency more than $\theta_{c-h} - 1$. In other words, no sunflowers exist.

Bounding the Number of Eliminated New Clauses – III

- If the branching at a nonleaf node is made based on a sunflower of c-clauses, then all the clause sets of length less than c are sparsified at the node.
- The set of petal clauses introduced by a petal branch is sparsified
- At any node, new c-clauses are almost sparsified: no subclause of length h occurs more than $2(\theta_{c-h} - 1)$ times.
- Therefore, a newly introduced h-clause can eliminate at most $2(\theta_{c-h} - 1)$ new c-clauses.

Bounding the Number of New Clauses

Number of new 1-clauses $\leq 2n = \beta_1 n$. For $c > 1$,

# of new ($\leq c$)-clauses

$$\leq \sum_{h=1}^{c-1} \#\text{ of new } c\text{-clauses eliminated}$$

by a new h-clause

$+ \#$ c-clauses at the leaf

$+ \#$ new clauses of length $\leq c - 1$

$$\leq \sum_{h=1}^{c-1} (2\theta_{c-h} - 2)\beta_h n + \theta_{c-1}\frac{2n}{c} + \beta_{c-1} n$$

$$\leq \sum_{h=1}^{c-1} 4\alpha\beta_{c-h}\beta_h n = \beta_c n$$

Bounding the Max Number of Petal Branches in T

- Each petal branch introduces at least θ_c new clauses for some c.
- Number of petal branches that introduce petals of size c $\leq \frac{\text{number of new } c\text{-clauses}}{\theta_c} \leq \frac{\beta_c n}{\theta_c} \leq \frac{n}{\alpha}$
- Total number of petal branches $\leq \sum_{c=1}^{k-1} \beta_c n/\theta_c = (k-1)n/\alpha$
- Conclusion: total number of leaves $\leq \binom{\beta n}{(k-1)n/\alpha} \leq 2^{\varepsilon n}$ by the choice of parameters

Open Problem

Lemma (Sparsification Lemma)

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

1. *$s \leq 2^{\epsilon n}$; $\textsc{Sat}(\phi) = \bigcup_i \textsc{Sat}(\phi_i)$, where $\textsc{Sat}(\phi)$ 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.*

Can we improve the sparsification constant (for a given ε) to $O(\frac{c}{\varepsilon})^{O(k)}$ for some absolute constant c?

Random k-SAT

Conjecture (Satisfiability Threshold Conjecture)

For every $k \geq 3$, there exists a constant $r_k > 0$ such that,

$$\lim_{n\to\infty} \mathbf{P}[F_k(n, rn) \textit{ is satisfiable}] = \begin{cases} 1, & \textit{if } r < r_k \\ 0, & \textit{if } r > r_k \end{cases}$$

It is known that $2^k \ln 2 - \Theta(k) \leq r_k \leq 2^k \ln 2 - \Theta(1)$ for $k \geq 3$.

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.

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$\}$;
- Define $s_\infty = \lim_{k \to \infty} s_k$
- 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)

Explanatory Value of **ETH** — III

- Lower bounds for $W[2]$-complete problems: There is no $f(k)n^{o(k)}$ algorithm for DOMINATING SET. Fellows (2011), Lokshtanov (2009)
- Lower bounds for problems parameterized by treewidth CHROMATIC NUMBER parameterized by treewidth t does not admit an algorithm that runs in time $2^{o(t \lg t)} n^{O(1)}$. Lokshtanov, Marx, and Saurabh (2011), Cygan, Nederlof, Pilipczuk, van Rooij, Wojtaszczyk (2011)
- LIST COLORING parameterized by treewidth does not admit algorithms that run in $f(t)n^{o(t)}$. Fellows, Fomin, Lokshtanov, Rosamond, Saurabh, Szeider, and Thomassen (2011)
- Workflow Satisfiability Problem parameterized by the number of steps k cannot have a $2^{o(k \lg k)} n^{O(1)}$ algorithm. Crampton, Cohen, Gutin, and Jones (2013)
- Many others $\cdots$

Explanatory Value of **ETH** — IV

- Can the **ETH** provide any information about the specific values of the constants in the exponents?
- Can we prove a specific non-zero constant lower bound on s_3 assuming **ETH**?

Explanatory Value of **ETH** — V

- Practical experience with SAT heuristics shows that the performance degrades as the clause width increases.
- Worst-case analysis of SAT algorithms also shows a degradation in performance with increasing clause width.
- Can **ETH** explain this behavior?

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 Value of **SETH** - I

- If **SETH** holds, k-Dominating Set does not have a $f(k)n^{k-\varepsilon}$ time algorithm. — Patrascu and Williams, 2009
- **SETH** implies that Independent Set parameterized by treewidth cannot be solved faster than $2^{tw}n^{O(1)}$ — Lokshtanov, Marx, and Saurabh 2010
- **SETH** implies that Dominating Set parameterized by treewidth cannot be solved faster than $3^{tw}n^{O(1)}$ — Lokshtanov, Marx, and Saurabh 2010
- Many others $\cdots$

Explanatory Value of **SETH** - II

Theorem

SETH *determines the exact complexities of the following problems in* **P**.

- $\forall \varepsilon > o$, *the* ORTHOGONAL VECTORS *problem for* n *binary vectors of dimension* $\omega(\log n)$ *cannot be solved in time* $O(n^{2-\varepsilon})$. — *Williams - 2004*
- $\forall \varepsilon > o$, *the* VECTOR DOMINATION *problem for* n *vectors of dimension* $\omega \log n$ *cannot be solved in time* $O(n^{2-\varepsilon})$. — *Williams - 2004, Impagliazzo, Paturi, Schneider - 2013*
- $\forall \varepsilon > o$, *the* FRÉCHET DISTANCE *problem for two piece-wise linear curves with* n *pieces* n *cannot be solved in time* $O(n^{2-\varepsilon})$. — *Bringmann - 2014*
- *Many others* $\cdots$ — *Borassi, Crescenzi, Habib - 2014, Abboud, Vassilevska Williams, 2014*

Explanatory Value of **SETH** - III

- Assuming **SETH**, can we prove a 2^n lower bound on CHROMATIC NUMBER?
- Assuming **SETH**, can we prove that $s_3 > c$ for some $c > 0$?

Open Problems

- 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?
- Are there better non-**OPP** algorithms for k-SAT or CIRCUIT SAT?
- Does there exist a c^{-n} success probability **OPP** algorithm for HAMILTONIAN PATH?

Thank You