(S)ETH and A Survey of Consequences

Mohan Paturi

Simons Institute, August 2015

Outline

1. Exact Algorithms and Complexity
2. Exponential-Time Hypothesis
3. Explanatory Value of **ETH** and **SETH**
4. Probabilistic Polynomial Time Algorithms
5. Open Problems

Exact Algorithms and Complexity

- All **NP**-complete problems are equivalent as far as polynomial time solvability is concerned.

Exact Algorithms and Complexity

- All **NP**-complete problems are equivalent as far as polynomial time solvability is concerned.
- However, some **NP**-complete problems have better exact algorithms than others

Exact Algorithms and Complexity

- All **NP**-complete problems are equivalent as far as polynomial time solvability is concerned.
- However, some **NP**-complete problems have better exact algorithms than others
- Exact algorithms — deterministic or randomized algorithms that produce exact solutions

Exact Algorithms and Complexity

- All **NP**-complete problems are equivalent as far as polynomial time solvability is concerned.
- However, some **NP**-complete problems have better exact algorithms than others
- Exact algorithms — deterministic or randomized algorithms that produce exact solutions
- Exact complexity — worst-case complexity of exact algorithms

Exact Algorithms and Complexity

- All **NP**-complete problems are equivalent as far as polynomial time solvability is concerned.
- However, some **NP**-complete problems have better exact algorithms than others
- Exact algorithms — deterministic or randomized algorithms that produce exact solutions
- Exact complexity — worst-case complexity of exact algorithms
- What improvements can we expect over exhaustive search or standard algorithms?

Exact Algorithms and Complexity

- All **NP**-complete problems are equivalent as far as polynomial time solvability is concerned.
- However, some **NP**-complete problems have better exact algorithms than others
- Exact algorithms — deterministic or randomized algorithms that produce exact solutions
- Exact complexity — worst-case complexity of exact algorithms
- What improvements can we expect over exhaustive search or standard algorithms?
- What are the obstructions that limit the improvements?

Exact Algorithms and Complexity

- All **NP**-complete problems are equivalent as far as polynomial time solvability is concerned.
- However, some **NP**-complete problems have better exact algorithms than others
- Exact algorithms — deterministic or randomized algorithms that produce exact solutions
- Exact complexity — worst-case complexity of exact algorithms
- What improvements can we expect over exhaustive search or standard algorithms?
- What are the obstructions that limit the improvements?
- What principles explain the exact complexities of **NP**-complete problems?

Exact Complexity — Parametrization of **NP** Problems

- Two (or more) parameters with each instance: m, the size of the input and n, a complexity parameter

Exact Complexity — Parametrization of **NP** Problems

- Two (or more) parameters with each instance: m, the size of the input and n, a complexity parameter
- Natural and robust complexity parameters
  1. k-SAT: formula $F \longrightarrow$ (formula size m, number of variables n)

Exact Complexity — Parametrization of **NP** Problems

- Two (or more) parameters with each instance: m, the size of the input and n, a complexity parameter
- Natural and robust complexity parameters
  1. k-SAT: formula $F \longrightarrow$ (formula size m, number of variables n)
  2. Also, k-SAT: formula $F \longrightarrow$ (formula size m, number of clauses)

Exact Complexity — Parametrization of **NP** Problems

- Two (or more) parameters with each instance: m, the size of the input and n, a complexity parameter
- Natural and robust complexity parameters
  1. k-SAT: formula $F \longrightarrow$ (formula size m, number of variables n)
  2. Also, k-SAT: formula $F \longrightarrow$ (formula size m, number of clauses)
  3. HAMILTONIAN PATH: graph $G = (V, E) \longrightarrow$ (size of the graph m, number of vertices n)

Exact Complexity — Parametrization of **NP** Problems

- Two (or more) parameters with each instance: m, the size of the input and n, a complexity parameter
- Natural and robust complexity parameters
  1. k-SAT: formula $F \longrightarrow$ (formula size m, number of variables n)
  2. Also, k-SAT: formula $F \longrightarrow$ (formula size m, number of clauses)
  3. Hamiltonian Path: graph $G = (V, E) \longrightarrow$ (size of the graph m, number of vertices n)
  4. Also, Hamiltonian Path: graph $G = (V, E) \longrightarrow$ (size of the graph m, $\log n!$)

NP Parametrization

- CIRCUIT SAT: circuit $F \longrightarrow$ (circuit size m, number of variables n)

NP Parametrization

- CIRCUIT SAT: circuit $F \longrightarrow$ (circuit size m, number of variables n)
- **NP** : Existentially quantified circuit $\exists y C(x, y) \rightarrow$ (circuit size m, number of existentially quantified Boolean variables $|y|$)

NP Parametrization

- Circuit Sat: circuit $F \longrightarrow$ (circuit size m, number of variables n)
- **NP** : Existentially quantified circuit $\exists y C(x, y) \rightarrow$ (circuit size m, number of existentially quantified Boolean variables $|y|$)
- Circuit Sat is the "mother" of all **NP**-complete problems under this natural parametrization.

NP Parametrization

- Circuit Sat: circuit $F \longrightarrow$ (circuit size m, number of variables n)
- **NP** : Existentially quantified circuit $\exists y C(x, y) \rightarrow$ (circuit size m, number of existentially quantified Boolean variables $|y|$)
- Circuit Sat is the "mother" of all **NP**-complete problems under this natural parametrization.
- Any **NP**-complete problem can be reduced to Circuit Sat preserving the complexity parameter exactly.

Improved Exact Algorithms

- Given an **NP** problem instance with size parameter m and complexity parameter n,

Improved Exact Algorithms

- Given an **NP** problem instance with size parameter m and complexity parameter n,
- Standard (deterministic or random) algorithms are those achieve worst-case time complexity $\mathtt{poly}(m)2^n$

Improved Exact Algorithms

- Given an **NP** problem instance with size parameter m and complexity parameter n,
- Standard (deterministic or random) algorithms are those achieve worst-case time complexity $\texttt{poly}(m)2^n$
- Improved exact algorithms are those that achieve worst-case time complexity $\texttt{poly}(m)2^{\mu n}$ for $\mu < 1$.

Improved Exact Algorithms

- Given an **NP** problem instance with size parameter m and complexity parameter n,
- Standard (deterministic or random) algorithms are those achieve worst-case time complexity $\texttt{poly}(m)2^n$
- Improved exact algorithms are those that achieve worst-case time complexity $\texttt{poly}(m)2^{\mu n}$ for $\mu < 1$.
- Also known as moderately exponential-time or nontrivial exponential-time algorithms

Improved Exponential Time Algorithms for k-SAT and k-COLORABILITY

- k-SAT, number of variables as the complexity parameter —
  backtracking and local search
  Hertli (2012), PPSZ, Schöning, PPZ, Rolf, Iwama, $\cdots$,
  Monien and Speckenmeyer (1985)

Improved Exponential Time Algorithms for k-SAT and k-COLORABILITY

- k-SAT, number of variables as the complexity parameter — backtracking and local search
  Hertli (2012), PPSZ, Schöning, PPZ, Rolf, Iwama, $\cdots$, Monien and Speckenmeyer (1985)
  - 3-SAT — $2^{0.386n}$

Improved Exponential Time Algorithms for k-SAT and k-COLORABILITY

- k-SAT, number of variables as the complexity parameter — backtracking and local search
  Hertli (2012), PPSZ, Schöning, PPZ, Rolf, Iwama, $\cdots$, Monien and Speckenmeyer (1985)
  - 3-SAT — $2^{0.386n}$
  - 4-SAT — $2^{0.554n}$

Improved Exponential Time Algorithms for k-SAT and k-COLORABILITY

- k-SAT, number of variables as the complexity parameter — backtracking and local search
  Hertli (2012), PPSZ, Schöning, PPZ, Rolf, Iwama, $\cdots$, 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.

Improved Exponential Time Algorithms for k-SAT and k-COLORABILITY

- k-SAT, number of variables as the complexity parameter — backtracking and local search
  Hertli (2012), PPSZ, Schöning, PPZ, Rolf, Iwama, $\cdots$, 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.
- k-COLORABILITY, number of vertices as the complexity parameter — backtracking
  Beigel and Eppstein (2005), Byskov (2004), $\cdots$

Improved Exponential Time Algorithms for k-SAT and k-COLORABILITY

- k-SAT, number of variables as the complexity parameter — backtracking and local search
  Hertli (2012), PPSZ, Schöning, PPZ, Rolf, Iwama, $\cdots$, 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.
- k-COLORABILITY, number of vertices as the complexity parameter — backtracking
  Beigel and Eppstein (2005), Byskov (2004), $\cdots$
  - 3-COLORABILITY — $2^{0.41n}$

Improved Exponential Time Algorithms for k-SAT and k-COLORABILITY

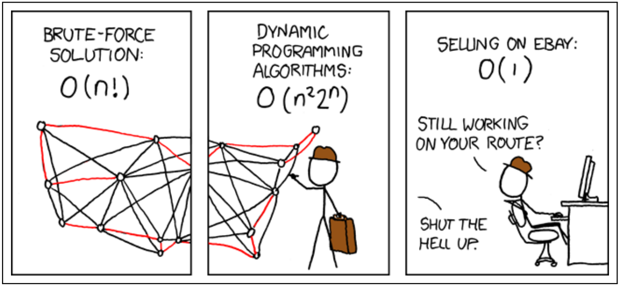
- k-SAT, number of variables as the complexity parameter — backtracking and local search
  Hertli (2012), PPSZ, Schöning, PPZ, Rolf, Iwama, $\cdots$, Monien and Speckenmeyer (1985)
  - 3-SAT — $2^{0.386n}$
  - 4-SAT — $2^{0.554n}$
  - k-SAT — $2^{(1-\mu_k/(k-1))n}$ where $\mu_k \approx 1.6$ for large k.
- k-COLORABILITY, number of vertices as the complexity parameter — backtracking
  Beigel and Eppstein (2005), Byskov (2004), $\cdots$
  - 3-COLORABILITY — $2^{0.41n}$
  - 4-COLORABILITY — $2^{0.807n}$

Improved Algorithms for HAMILTONIAN PATH

- HAMILTONIAN PATH, number of vertices as the complexity parameter — dynamic programming, inclusion-exclusion, determinant summation formulas, algebraic sieving
Björklund (2010), Bax (1993), Karp (1982), Kohn, Gottlieb, and Kohn (1977), Held and Karp (1962), Bellman (1962)

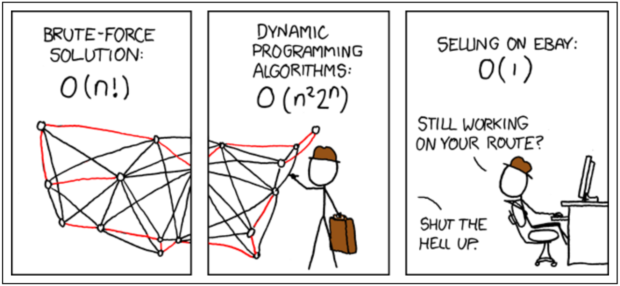
Improved Algorithms for HAMILTONIAN PATH

- HAMILTONIAN PATH, number of vertices as the complexity parameter — dynamic programming, inclusion-exclusion, determinant summation formulas, algebraic sieving
  Björklund (2010), Bax (1993), Karp (1982), Kohn, Gottlieb, and Kohn (1977), Held and Karp (1962), Bellman (1962)
  - HAMILTONIAN PATH — 2^n



Improved Algorithms for HAMILTONIAN PATH

- HAMILTONIAN PATH, number of vertices as the complexity parameter — dynamic programming, inclusion-exclusion, determinant summation formulas, algebraic sieving
  Björklund (2010), Bax (1993), Karp (1982), Kohn, Gottlieb, and Kohn (1977), Held and Karp (1962), Bellman (1962)
  - HAMILTONIAN PATH — 2^n



  - undirected HAMILTONIAN PATH — $2^{0.67n}$

Improved Algorithms for COLORABILITY

- COLORABILITY, number of vertices as the complexity parameter — dynamic programming, Möbius inversion
  Björklund, Husfeldt, Kaski, Koivisto (2006-2008), $\cdots$, Byskov (2004), Eppstein (2003), Lawler (1976)
    - COLORABILITY — 2^n in exponential space

Improved Algorithms for Max Independent Set

- Max Independent Set, number of vertices as the complexity parameter — backtracking, measure and conquer Fomin, Grandoni, and Kratsch (2005), Robson (1986), Jian (1986), Tarjan and Trojanowski (1977)

Improved Algorithms for MAX INDEPENDENT SET

- MAX INDEPENDENT SET, number of vertices as the complexity parameter — backtracking, measure and conquer Fomin, Grandoni, and Kratsch (2005), Robson (1986), Jian (1986), Tarjan and Trojanowski (1977)
  - $2^{0.287n}$ in polynomial space

Improved Algorithms for MAX INDEPENDENT SET

- MAX INDEPENDENT SET, number of vertices as the complexity parameter — backtracking, measure and conquer Fomin, Grandoni, and Kratsch (2005), Robson (1986), Jian (1986), Tarjan and Trojanowski (1977)
  - $2^{0.287n}$ in polynomial space
  - $2^{0.276n}$ in exponential space

Improved Algorithms for MAX INDEPENDENT SET

- MAX INDEPENDENT SET, number of vertices as the complexity parameter — backtracking, measure and conquer
Fomin, Grandoni, and Kratsch (2005), Robson (1986), Jian (1986), Tarjan and Trojanowski (1977)
    - $2^{0.287n}$ in polynomial space
    - $2^{0.276n}$ in exponential space
- 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)

Improved Algorithms for MAX INDEPENDENT SET

- MAX INDEPENDENT SET, number of vertices as the complexity parameter — backtracking, measure and conquer
  Fomin, Grandoni, and Kratsch (2005), Robson (1986), Jian (1986), Tarjan and Trojanowski (1977)
  - $2^{0.287n}$ in polynomial space
  - $2^{0.276n}$ in exponential space
- 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)
  - $\mathbf{AC}^0$ Satisfiability for circuits of size cn and depth d — $2^{n(1-1/\Theta(c^d))}$

Improved Algorithms for Max Independent Set

- Max Independent Set, number of vertices as the complexity parameter — backtracking, measure and conquer Fomin, Grandoni, and Kratsch (2005), Robson (1986), Jian (1986), Tarjan and Trojanowski (1977)
  - $2^{0.287n}$ in polynomial space
  - $2^{0.276n}$ in exponential space
- 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)
  - **AC**^{0} Satisfiability for circuits of size cn and depth d — $2^{n(1-1/\Theta(c^d))}$
  - **ACC** Satisfiability — $2^{n-n^{\varepsilon}}$

Improved Algorithms for MAX INDEPENDENT SET

- MAX INDEPENDENT SET, number of vertices as the complexity parameter — backtracking, measure and conquer
  Fomin, Grandoni, and Kratsch (2005), Robson (1986), Jian (1986), Tarjan and Trojanowski (1977)
  - $2^{0.287n}$ in polynomial space
  - $2^{0.276n}$ in exponential space
- 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)
  - $\mathbf{AC}^0$ Satisfiability for circuits of size cn and depth d — $2^{n(1-1/\Theta(c^d))}$
  - **ACC** Satisfiability — $2^{n-n^{\varepsilon}}$
  - Formula Satisfiability for formulas of size cn — $2^{n(1-1/c^2)}$

Improved Algorithms for MAX INDEPENDENT SET

- MAX INDEPENDENT SET, number of vertices as the complexity parameter — backtracking, measure and conquer Fomin, Grandoni, and Kratsch (2005), Robson (1986), Jian (1986), Tarjan and Trojanowski (1977)
  - $2^{0.287n}$ in polynomial space
  - $2^{0.276n}$ in exponential space
- 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)
  - **AC**^{0} Satisfiability for circuits of size cn and depth d — $2^{n(1-1/\Theta(c^d))}$
  - **ACC** Satisfiability — $2^{n-n^{\varepsilon}}$
  - Formula Satisfiability for formulas of size cn — $2^{n(1-1/c^2)}$
  - Formula Satisfiability for formulas of size cn over the full binary basis — $2^{n(1-1/2^{-c^2})}$

Improved Algorithms for Max Independent Set

- Max Independent Set, number of vertices as the complexity parameter — backtracking, measure and conquer Fomin, Grandoni, and Kratsch (2005), Robson (1986), Jian (1986), Tarjan and Trojanowski (1977)
  - $2^{0.287n}$ in polynomial space
  - $2^{0.276n}$ in exponential space
- 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)
  - **AC^0** Satisfiability for circuits of size cn and depth d — $2^{n(1-1/\Theta(c^d))}$
  - **ACC** Satisfiability — $2^{n-n^{\varepsilon}}$
  - Formula Satisfiability for formulas of size cn — $2^{n(1-1/c^2)}$
  - Formula Satisfiability for formulas of size cn over the full binary basis — $2^{n(1-1/2^{-c^2})}$
  - Depth-2 Threshold Circuit Satisfiability for circuits with cn

Exact Complexity — Motivating Questions

- Which problems have improved algorithms?
  Is there a $2^{\mu n}$ algorithm for COLORABILITY or CNFSAT or CIRCUIT SAT for $\mu < 1$?

Exact Complexity — Motivating Questions

- Which problems have improved algorithms?
  Is there a $2^{\mu n}$ algorithm for COLORABILITY or CNFSAT or CIRCUIT SAT for $\mu < 1$?
- Can the improvements extend to arbitrarily small exponents?
  Is 3-SAT solvable in subexponential-time? How about 3-COLORABILITY?

Exact Complexity — Motivating Questions

- Which problems have improved algorithms?
  Is there a $2^{\mu n}$ algorithm for COLORABILITY or CNFSAT or CIRCUIT SAT for $\mu < 1$?
- Can the improvements extend to arbitrarily small exponents?
  Is 3-SAT solvable in subexponential-time? How about 3-COLORABILITY?
- Can we prove improvements beyond a certain point are not possible (at least under some complexity assumption)?
  Lower bounding the exponent for 3-SAT under suitable complexity assumptions?

Exact Complexity — Motivating Questions

- Which problems have improved algorithms?
  Is there a $2^{\mu n}$ algorithm for Colorability or CNFSAT or Circuit Sat for $\mu < 1$?
- Can the improvements extend to arbitrarily small exponents?
  Is 3-SAT solvable in subexponential-time? How about 3-Colorability?
- Can we prove improvements beyond a certain point are not possible (at least under some complexity assumption)?
  Lower bounding the exponent for 3-SAT under suitable complexity assumptions?
- Is progress on different problems connected?
  Do improved algorithms for 3-SAT imply improved algorithms for 3-Colorability or vice versa?
  If Colorability has a 2^{cn} algorithm, can we prove CNFSAT has a 2^{dn} algorithm for some $c, d < 1$?

A Zoo of Algorithms, Techniques, and Analyses

- A lot of effort has gone into improving the exponents.

A Zoo of Algorithms, Techniques, and Analyses

- A lot of effort has gone into improving the exponents.
- A disparate variety of algorithmic techniques and analyses have been used.
  backtracking, local search, split and list, random restrictions, dynamic programming, algebraization, matrix multiplication, Möbius inversion, measure and conquer, inclusion-exclusion, determinant summation formulas, algebraic sieving

A Zoo of Algorithms, Techniques, and Analyses

- A lot of effort has gone into improving the exponents.
- A disparate variety of algorithmic techniques and analyses have been used.
  backtracking, local search, split and list, random restrictions, dynamic programming, algebraization, matrix multiplication, Möbius inversion, measure and conquer, inclusion-exclusion, determinant summation formulas, algebraic sieving
- A priori, it is not clear that we can expect a common principle to govern the exact complexities.

Connections between Problems

- Is there a connection between the exponential complexities of problems?

Connections between Problems

- Is there a connection between the exponential complexities of problems?
- 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?

Connections between Problems

- Is there a connection between the exponential complexities of problems?
- 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?
- Problem: 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.

Connections between Problems

- Is there a connection between the exponential complexities of problems?
- 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?
- Problem: 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.

Sparsification Lemma

Lemma (Sparsification Lemma, IPZ 1997)

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

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

Sparsification Lemma

Lemma (Sparsification Lemma, IPZ 1997)

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

- To complete the reduction from 3-SAT to 3-COLORABILITY and preserve linearity in the parameter value,

Sparsification Lemma

Lemma (Sparsification Lemma, IPZ 1997)

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

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

- To complete the reduction from 3-SAT to 3-COLORABILITY and preserve linearity in the parameter value,
- Apply Sparsification Lemma to the given 3-CNF ϕ.

Sparsification Lemma

Lemma (Sparsification Lemma, IPZ 1997)

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

- To complete the reduction from 3-SAT to 3-COLORABILITY and preserve linearity in the parameter value,
- 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.

Sparsification Lemma

Lemma (Sparsification Lemma, IPZ 1997)

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

- To complete the reduction from 3-SAT to 3-COLORABILITY and preserve linearity in the parameter value,
- 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.

Sparsification Lemma

Lemma (Sparsification Lemma, IPZ 1997)

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

- To complete the reduction from 3-SAT to 3-COLORABILITY and preserve linearity in the parameter value,
- 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.
- Key ideas: subexponential time reductions, sparsification

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

- **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 $k \geq 3$.

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 $[n]$.

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**.

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 (Impagliazzo, P, and Zane (1977))

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

Completeness of 3-SAT in **SNP**

Theorem (Impagliazzo, P, and Zane (1977))

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 existential quantifiers.

Completeness of 3-SAT in **SNP**

Theorem (Impagliazzo, P, and Zane (1977))

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 existential quantifiers.
- Reduce k-SAT to the union of subexponentially many linear-size k-SAT using Sparsification Lemma.

Completeness of 3-SAT in **SNP**

Theorem (Impagliazzo, P, and Zane (1977))

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 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**)

- The previous theorem gives evidence that 3-SAT does not have a subexponential-time algorithm as it is unlikely that the whole class **SNP** has such algorithms.

Exponential-time Hypothesis (**ETH**)

- The previous theorem gives evidence that 3-SAT does not have a subexponential-time algorithm as it is unlikely that the whole class **SNP** has such algorithms.
- While it seems beyond our scope to prove this, our plan is to explore the state of affairs given the likelihood.

Exponential-time Hypothesis (**ETH**)

- The previous theorem gives evidence that 3-SAT does not have a subexponential-time algorithm as it is unlikely that the whole class **SNP** has such algorithms.
- While it seems beyond our scope to prove this, our plan is to explore the state of affairs given the likelihood.
- Let $s_k = \inf\{\delta | \exists \quad 2^{\delta n} \text{ algorithm for } k\text{-SAT}\}$;

Exponential-time Hypothesis (**ETH**)

- The previous theorem gives evidence that 3-SAT does not have a subexponential-time algorithm as it is unlikely that the whole class **SNP** has such algorithms.
- While it seems beyond our scope to prove this, our plan is to explore the state of affairs given the likelihood.
- Let $s_k = \inf\{\delta | \exists \;\; 2^{\delta n}$ algorithm for k-SAT$\}$;
- **ETH** — Exponential Time Hypothesis: $s_3 > 0$

Exponential-time Hypothesis (**ETH**)

- The previous theorem gives evidence that 3-SAT does not have a subexponential-time algorithm as it is unlikely that the whole class **SNP** has such algorithms.
- While it seems beyond our scope to prove this, our plan is to explore the state of affairs given the likelihood.
- Let $s_k = \inf\{\delta | \exists \quad 2^{\delta n}$ algorithm for k-SAT$\}$;
- **ETH** — Exponential Time Hypothesis: $s_3 > 0$
- Assuming **ETH**, we conclude none of the problems in (size-constrained) **SNP** have a subexponential time algorithms

Exponential-time Hypothesis (**ETH**)

- The previous theorem gives evidence that 3-SAT does not have a subexponential-time algorithm as it is unlikely that the whole class **SNP** has such algorithms.
- While it seems beyond our scope to prove this, our plan is to explore the state of affairs given the likelihood.
- Let $s_k = \inf\{\delta | \exists \ \ 2^{\delta n}$ algorithm for k-SAT$\}$;
- **ETH** — Exponential Time Hypothesis: $s_3 > 0$
- Assuming **ETH**, we conclude none of the problems in (size-constrained) **SNP** have a subexponential time algorithms
- Furthermore, **SNP**-hard problems such as HAMILTONIAN PATH cannot have a subexponential time algorithm.

Explanatory Value of **ETH**

- We have a very little understanding of exponential time algorithms.

Explanatory Value of **ETH**

- We have a very little understanding of exponential time algorithms.
- For **ETH** to be useful,

Explanatory Value of **ETH**

- We have a very little understanding of exponential time algorithms.
- For **ETH** to be useful,
  - it must be able to provide an explanation for the exact complexities of various other problems,

Explanatory Value of **ETH**

- We have a very little understanding of exponential time algorithms.
- For **ETH** to be useful,
    - it must be able to provide an explanation for the exact complexities of various other problems,
    - ideally, by providing lower bounds that match the upper bounds from the best known algorithms.

Explanatory Value of **ETH**

- We have a very little understanding of exponential time algorithms.
- For **ETH** to be useful,
  - it must be able to provide an explanation for the exact complexities of various other 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.

Lower Bounds based on **ETH** — I

- We follow the nice summary provided by Lokshtanov, Marx and Saurabh (2011).

Lower Bounds based on **ETH** — I

- We follow the nice summary provided by Lokshtanov, Marx and Saurabh (2011).
- All the following results assume **ETH**.

Lower Bounds based on **ETH** — I

- We follow the nice summary provided by Lokshtanov, Marx and Saurabh (2011).
- All the following results assume **ETH**.
- 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 based on **ETH** — I

- We follow the nice summary provided by Lokshtanov, Marx and Saurabh (2011).
- All the following results assume **ETH**.
- 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 based on **ETH** — I

- We follow the nice summary provided by Lokshtanov, Marx and Saurabh (2011).
- All the following results assume **ETH**.
- 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)

Lower Bounds based on **ETH** — II

- 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 based on **ETH** — II

- 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)

Lower Bounds based on **ETH** — II

- 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)

Lower Bounds based on **ETH** — II

- 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)

Lower Bounds based on **ETH** — II

- 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 $\cdots$

Complexity of k-SAT with Increasing k

Theorem (Impagliazzo and P, 1999)

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

Complexity of k-SAT with Increasing k

Theorem (Impagliazzo and P, 1999)

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

- Let $s_\infty = \lim_{k\to\infty} s_k$.

Complexity of k-SAT with Increasing k

Theorem (Impagliazzo and P, 1999)

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

- Let $s_\infty = \lim_{k \to \infty} s_k$.
- More specifically, we prove $s_\infty - s_k \geq d/k$ for some absolute constant $d > 0$.

Complexity of k-SAT with Increasing k

Theorem (Impagliazzo and P, 1999)

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

- Let $s_\infty = \lim_{k \to \infty} s_k$.
- More specifically, we prove $s_\infty - s_k \geq d/k$ for some absolute constant $d > 0$.
- Provides evidence to the observation that heuristics for k-SAT perform worse as k increases.

Complexity of k-SAT with Increasing k

Theorem (Impagliazzo and P, 1999)

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

- Let $s_\infty = \lim_{k\to\infty} s_k$.
- More specifically, we prove $s_\infty - s_k \geq d/k$ for some absolute constant $d > 0$.
- Provides evidence to the observation that heuristics for k-SAT perform worse as k increases.
- Proof Sketch: Trade clause width up to reduce the number of variables: reduce k-SAT to k'-CNF for $k' \gg k$ such that the resultant formula has fewer variables.

Complexity of CSP Problems

- **ETH** implies that $(d, 2)$-CSP requires d^{cn} time where c is an absolute constant. The constant c depends on s_3. Traxler 2008
- $(d, 2)$-CSP is the class of constraint satisfaction problems where variables take d values and each clause has two variables.

Complexity of CSP Problems

- **ETH** implies that $(d, 2)$-CSP requires d^{cn} time where c is an absolute constant. The constant c depends on s_3. Traxler 2008
- $(d, 2)$-CSP is the class of constraint satisfaction problems where variables take d values and each clause has two variables.
- Proof involves reducing a $(d, 2)$-CSP instance to a $(d', 2)$-CSP instance for $d' \gg d$, but with fewer variables.

Complexity of CSP Problems

- **ETH** implies that $(d, 2)$-CSP requires d^{cn} time where c is an absolute constant. The constant c depends on s_3. Traxler 2008
- $(d, 2)$-CSP is the class of constraint satisfaction problems where variables take d values and each clause has two variables.
- Proof involves reducing a $(d, 2)$-CSP instance to a $(d', 2)$-CSP instance for $d' \gg d$, but with fewer variables.
- A special case of $(k, 2)$-CSP, k-$\textsc{Colorability}$, has a 2^n algorithm (exponent is independent of k).

Complexity of CSP Problems

- **ETH** implies that $(d, 2)$-CSP requires d^{cn} time where c is an absolute constant. The constant c depends on s_3. Traxler 2008
- $(d, 2)$-CSP is the class of constraint satisfaction problems where variables take d values and each clause has two variables.
- Proof involves reducing a $(d, 2)$-CSP instance to a $(d', 2)$-CSP instance for $d' \gg d$, but with fewer variables.
- A special case of $(k, 2)$-CSP, k-COLORABILITY, has a 2^n algorithm (exponent is independent of k).
- Greater expressiveness of k'-CNF and $(d', 2)$-CSP has been exploited.

SETH — Strong Exponential Time Hypothesis

- Earlier result regarding the increasing complexity of k-SAT tempts one to hypothesize
  SETH — Strong Exponential Time Hypothesis: $s_\infty = 1$

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 *formuals, cannot be computed in time* $O(2^{\varepsilon n})$ *time.*

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 *formuals, 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.*

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

Lower Bounds based on **SETH** - I

- If **SETH** holds, k-DOMINATING SET does not have a $f(k)n^{k-\varepsilon}$ time algorithm. — Patrascu and Williams, 2009

Lower Bounds based on **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

Lower Bounds based on **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 $\cdots$

Lower Bounds based on **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*

Lower Bounds based on **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*

Lower Bounds based on **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*

Probabilistic Polynomial Time Algorithms

- Consider natural, though restricted, models of computation for exponential time algorithms.

Probabilistic Polynomial Time Algorithms

- Consider natural, though restricted, models of computation for exponential time algorithms.
- **OP**$(T(n,m))$: one-sided error probabilistic algorithms that run in time $T(n,m)$

Probabilistic Polynomial Time Algorithms

- Consider natural, though restricted, models of computation for exponential time algorithms.
- $\mathbf{OP}(T(n,m))$: one-sided error probabilistic algorithms that run in time $T(n,m)$
- **OPP**: $\mathbf{OP}(T(n,m))$ where $T(n,m)$ is polynomially bounded.

Probabilistic Polynomial Time Algorithms

- Consider natural, though restricted, models of computation for exponential time algorithms.
- **OP**($T(n, m)$): one-sided error probabilistic algorithms that run in time $T(n, m)$
- **OPP**: **OP**($T(n, m)$) where $T(n, m)$ is polynomially bounded.
- Includes several Davis-Putnam style backtracking algorithms, local search algorithms

Probabilistic Polynomial Time Algorithms

- Consider natural, though restricted, models of computation for exponential time algorithms.
- **OP**$(T(n,m))$: one-sided error probabilistic algorithms that run in time $T(n,m)$
- **OPP**: **OP**$(T(n,m))$ where $T(n,m)$ is polynomially bounded.
- Includes several Davis-Putnam style backtracking algorithms, local search algorithms

Probabilistic Polynomial Time Algorithms

- Consider natural, though restricted, models of computation for exponential time algorithms.
- **OP**($T(n, m)$): one-sided error probabilistic algorithms that run in time $T(n, m)$
- **OPP**: **OP**($T(n, m)$) where $T(n, m)$ is polynomially bounded.
- Includes several Davis-Putnam style backtracking algorithms, local search algorithms
- OPP: space efficiency, parallelization, speed-up by quantum computation

Probabilistic Polynomial Time Algorithms

- Consider natural, though restricted, models of computation for exponential time algorithms.
- **OP**$(T(n, m))$: one-sided error probabilistic algorithms that run in time $T(n, m)$
- **OPP**: **OP**$(T(n, m))$ where $T(n, m)$ is polynomially bounded.
- Includes several Davis-Putnam style backtracking algorithms, local search algorithms
- OPP: space efficiency, parallelization, speed-up by quantum computation
- What is the best success probability achievable in OPP or OP$(T(n, m))$?

Probabilistic Polynomial Time Algorithms

- CIRCUIT SAT problem can be solved with probability $2^{-n+O(\lg T(n,m))}$ using $\mathrm{OP}(T(n,m))$ algorithms.

Probabilistic Polynomial Time Algorithms

- Circuit Sat problem can be solved with probability $2^{-n+O(\lg T(n,m))}$ using $\mathrm{OP}(T(n,m))$ algorithms.
- Best-known deterministic algorithm takes time $2^n\texttt{poly}(m)$.

Probabilistic Polynomial Time Algorithms

- CIRCUIT SAT problem can be solved with probability $2^{-n+O(\lg T(n,m))}$ using $\mathrm{OP}(T(n,m))$ algorithms.
- Best-known deterministic algorithm takes time $2^n \mathtt{poly}(m)$.
- Hamiltonian path problem can be solved with probability $1/n!$ in OPP.

Probabilistic Polynomial Time Algorithms

- CIRCUIT SAT problem can be solved with probability $2^{-n+O(\lg T(n,m))}$ using $\mathrm{OP}(T(n,m))$ algorithms.
- Best-known deterministic algorithm takes time $2^n \texttt{poly}(m)$.
- Hamiltonian path problem can be solved with probability $1/n!$ in OPP.
- The best known deterministic exponential time algorithm takes time $2^{O(n)}\texttt{poly}(m)$.

Time and Success Probability Trade-off

- Let $X(n) = (\lg t + \lg 1/p)/n$ where p is the best success probability for time t.

Time and Success Probability Trade-off

- Let $X(n) = (\lg t + \lg 1/p)/n$ where p is the best success probability for time t.
- Let $X = \lim_{n\to\infty} X(n)$.

Time and Success Probability Trade-off

- Let $X(n) = (\lg t + \lg 1/p)/n$ where p is the best success probability for time t.
- Let $X = \lim_{n\to\infty} X(n)$.
- How does X behave as a function of t?

Time and Success Probability Trade-off

- Let $X(n) = (\lg t + \lg 1/p)/n$ where p is the best success probability for time t.
- Let $X = \lim_{n\to\infty} X(n)$.
- How does X behave as a function of t?
- For the CIRCUIT SAT problem, based on the best known algorithms, $X = 1$ whether t is polynomial in n or exponential in n.

Time and Success Probability Trade-off

- Let $X(n) = (\lg t + \lg 1/p)/n$ where p is the best success probability for time t.
- Let $X = \lim_{n\to\infty} X(n)$.
- How does X behave as a function of t?
- For the CIRCUIT SAT problem, based on the best known algorithms, $X = 1$ whether t is polynomial in n or exponential in n.
- On the other hand, for HAMILTONIAN PATH, based on the best known algorithms, $X = \infty$ when t is polynomial in n and $X \leq 1$ when t is exponential is n.

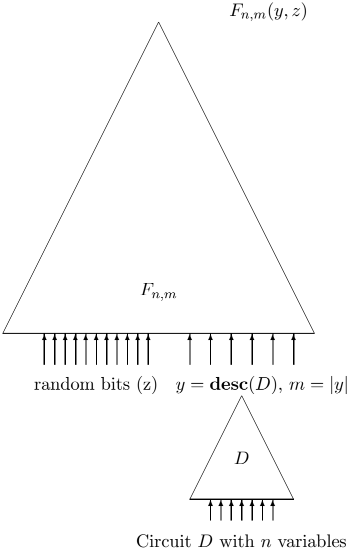
Time and Success Probability Trade-off

- Let $X(n) = (\lg t + \lg 1/p)/n$ where p is the best success probability for time t.
- Let $X = \lim_{n\to\infty} X(n)$.
- How does X behave as a function of t?
- For the CIRCUIT SAT problem, based on the best known algorithms, $X = 1$ whether t is polynomial in n or exponential in n.
- On the other hand, for HAMILTONIAN PATH, based on the best known algorithms, $X = \infty$ when t is polynomial in n and $X \leq 1$ when t is exponential is n.
- For what problems, does this quantity decrease/stay the same over a certain range of time?

Time and Success Probability Trade-off

- Let $X(n) = (\lg t + \lg 1/p)/n$ where p is the best success probability for time t.
- Let $X = \lim_{n\to\infty} X(n)$.
- How does X behave as a function of t?
- For the CIRCUIT SAT problem, based on the best known algorithms, $X = 1$ whether t is polynomial in n or exponential in n.
- On the other hand, for HAMILTONIAN PATH, based on the best known algorithms, $X = \infty$ when t is polynomial in n and $X \leq 1$ when t is exponential is n.
- For what problems, does this quantity decrease/stay the same over a certain range of time?
- We present (weak) evidence that for CIRCUIT SAT, $(\log t + \log 1/p)/n$ may not significantly decrease with increasing time.

Circuit Family for deciding CIRCUIT SAT



Probabilistic Circuit for **CircuitSat**

Success Probability for CIRCUIT SAT with OPP algorithms

Theorem (P, Pudlák 2010)

If CIRCUIT SAT *can be decided with probabilistic circuits of size m^k for some k with success probability $2^{-\delta n}$ for $\delta < 1$, then there exists a $\mu < 1$ depending on k and δ such that* CIRCUIT SAT(n, m) *can be decided by deterministic circuits of size $2^{O(n^{\mu} \lg^{1-\mu} m)}$.*

Results: Quasilinear Size Circuits

Theorem

If $\textsc{Circuit Sat}$ *can be decided with probabilistic circuits of size* $\tilde{O}(m)$ *with success probability* $2^{-\delta n}$ *for* $\delta < 1$, *then* $\textsc{Circuit Sat}(n, m)$ *can be decided by deterministic circuits of size* $O(\texttt{poly}(m) n^{O(\lg \lg m)})$.

Results: Quasilinear Size Circuits

Theorem

If CIRCUIT SAT *can be decided with probabilistic circuits of size* $\tilde{O}(m)$ *with success probability* $2^{-\delta n}$ *for* $\delta < 1$, *then* CIRCUIT SAT(n, m) *can be decided by deterministic circuits of size* $O(\mathtt{poly}(m) n^{O(\lg \lg m)})$.

- The consequence is very close to the statement $\mathbf{NP} \subseteq \mathbf{P}/\mathtt{poly}$.

Success Probability for Circuit Sat with Subexponential Size Circuits

Theorem (P, Pudlák 2010)

If Circuit Sat *can be decided with probabilistic circuits of size* $2^{o(n)}\tilde{O}(m)$ *with success probability* $2^{-\delta n}$ *for* $\delta < 1$*, then* Circuit Sat(n, m) *can be decided by deterministic circuits of size* $2^{o(n)}\mathtt{poly}(m)$.

Exponential Amplification Lemma

Lemma (P and Pudlák 2010)

Exponential Amplification Lemma: *Let $\mathcal{F}$ be an f-bounded family for some $f : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ that decides* Circuit Sat *with success probability $2^{-\delta n}$ for $0 < \delta < 1$. Then there exists a g-bounded circuit family $\mathcal{G}$ that decides* Circuit Sat *with success probability at least $2^{-\delta^2 n}$ where $g(n, m) = O(f(\lceil \delta n \rceil + 5, \tilde{O}(f(n, m))))$.*

Exponential Amplification Lemma

Lemma (P and Pudlák 2010)

Exponential Amplification Lemma: *Let $\mathcal{F}$ be an f-bounded family for some $f : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ that decides* Circuit Sat *with success probability $2^{-\delta n}$ for $0 < \delta < 1$. Then there exists a g-bounded circuit family $\mathcal{G}$ that decides* Circuit Sat *with success probability at least $2^{-\delta^2 n}$ where $g(n, m) = O(f(\lceil \delta n \rceil + 5, \tilde{O}(f(n, m))))$.*

- $\mathcal{F} : (f(n, m), \delta n) \to \mathcal{G} : (g(n, m), \delta^2 n)$

Drucker's Recent Result

Theorem (Drucker, 2013)

For any $\mu < 1$, if there is an OPP algorithm which takes the description of a 3-$\textsc{SAT}$ *formula of length m as input and decides its satisfiability with success probability at least $2^{-m^{\mu}}$, then* $\mathbf{NP} \subseteq \mathbf{coNP}/\mathtt{poly}$

Other Connections

- Hardest instances
- Satisfiability and circuit lower bounds
- $\cdots$

Open Problems

- Assuming **ETH** or other suitable assumption, prove

Open Problems

- Assuming **ETH** or other suitable assumption, prove

Open Problems

- Assuming **ETH** or other suitable assumption, prove
  - a specific lower bound on s_3
  - $s_\infty = 1$ (**SETH**)

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?

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?

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