

Exact Complexity and Satisfiability

What does complexity theory have to say about the difficulty of Satisfiability?

Ramamohan Paturi

University of California, San Diego
Invited Talk — SAT 2010

July 2010

Outline

- 1 Exact Algorithms
- 2 Exact Complexity
- 3 Exponential-Time Hypothesis
- 4 CIRCUIT SAT
- 5 Summary

Exact Algorithms for **NP**-complete Problems

- Exact solutions, worst-case complexity, deterministic or randomized

Exact Algorithms for **NP**-complete Problems

- Exact solutions, worst-case complexity, deterministic or randomized
- Improvements over **exhaustive search** or **standard** algorithms

Exact Algorithms for NP-complete Problems

- Exact solutions, worst-case complexity, deterministic or randomized
- Improvements over **exhaustive search** or **standard** algorithms
- What are the **obstructions** for improved exponential-time algorithms?

Exact Complexity — Parameterization of **NP** Problems

- Two parameters with each instance: m , the size of the input and n , the **complexity parameter**

Exact Complexity — Parameterization of NP Problems

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

Exact Complexity — Parameterization of NP Problems

- Two parameters with each instance: m , the size of the input and n , the **complexity parameter**
- Natural and robust complexity parameters
 - 1 k -SAT: formula $F \rightarrow$ (formula size m , **number of variables n**)
 - 2 Alternatively, formula $F \rightarrow$ (formula size m , **number of clauses**)

Exact Complexity — Parameterization of NP Problems

- Two parameters with each instance: m , the size of the input and n , the **complexity parameter**
- Natural and robust complexity parameters
 - 1 k -SAT: formula $F \rightarrow$ (formula size m , **number of variables** n)
 - 2 Alternatively, formula $F \rightarrow$ (formula size m , **number of clauses**)
 - 3 CIRCUIT SAT: circuit $F \rightarrow$ (circuit size m , **number of variables** n)

Exact Complexity — Parameterization of NP Problems

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

Exact Complexity — Parameterization of NP Problems

- Two parameters with each instance: m , the size of the input and n , the **complexity parameter**
- Natural and robust complexity parameters
 - 1 k -SAT: formula $F \longrightarrow$ (formula size m , **number of variables** n)
 - 2 Alternatively, formula $F \longrightarrow$ (formula size m , **number of clauses**)
 - 3 CIRCUIT SAT: circuit $F \longrightarrow$ (circuit size m , **number of variables** n)
 - 4 HAMILTONIAN PATH: graph $G = (V, E) \longrightarrow$ (size of the graph m , **number of vertices** n)
 - 5 Alternatively, graph $G = (V, E) \longrightarrow$ (size of the graph m , **$\log n!$**)

Improved Exact Algorithms

- Given an **NP** problem instance with size parameter m and complexity parameter n , we call
- **Standard** algorithms: (deterministic or random) worst-case time complexity — $2^n \text{poly}(m)$

Improved Exact Algorithms

- Given an **NP** problem instance with size parameter m and complexity parameter n , we call
- **Standard** algorithms: (deterministic or random) worst-case time complexity — $2^n \text{poly}(m)$
- **Improved exact algorithms**: worst-case time complexity — $O(\text{poly}(m)2^{\mu n})$, $\mu < 1$.

Improved Exact Algorithms

- Given an **NP** problem instance with size parameter m and complexity parameter n , we call
- **Standard** algorithms: (deterministic or random) worst-case time complexity — $2^n \text{poly}(m)$
- **Improved exact algorithms**: worst-case time complexity — $O(\text{poly}(m)2^{\mu n})$, $\mu < 1$.
- Also known as **moderately exponential-time** or **nontrivial exponential-time algorithms**

Improved Exact Algorithms

- Given an **NP** problem instance with size parameter m and complexity parameter n , we call
- **Standard** algorithms: (deterministic or random) worst-case time complexity — $2^n \text{poly}(m)$
- **Improved exact algorithms**: worst-case time complexity — $O(\text{poly}(m)2^{\mu n})$, $\mu < 1$.
- Also known as **moderately exponential-time** or **nontrivial exponential-time** algorithms
- What improvements can we expect over standard algorithms?
Our focus is on μ .

Improved Algorithms: Maximum Independent Set

- Given $G = (V, E)$, find a maximum size independent set. Number of vertices is the complexity parameter
- $2^{0.334n}$ algorithm in polynomial space — [Tarjan and Trojanowski 1977](#)
- $2^{0.304n}$ algorithm in polynomial space — [T. Jian 1986](#)
- $2^{0.296n}$ in polynomial space and $2^{0.276n}$ in exponential space — [Robson 1986](#)
- $2^{0.25n}$ — [Robson 2001](#), relatively long, partially computer-generated proof in a technical report
- $2^{0.287n}$ in polynomial space using [measure and conquer](#) analysis technique — [Fomin, Grandoni, and Kratsch 2006](#)
- Better bounds are known for sparse graphs.

Improved Algorithms: k -SAT

- Decide if given a k -CNF Φ is satisfiable. n , the number of variables is the complexity parameter
- Best known bounds for small values of k : $2^{?n}$

k	unique- k -SAT	k -SAT	k -SAT	k -SAT	k -SAT
3	0.386...	0.521...	0.415...	0.409...	0.404...
4	0.554...	0.562...	0.584...		0.559...
5	0.650...		0.678...		
6	0.711...		0.736...		
	Paturi, Pudlák, Saks, Zane		Schöning	Rolf, ...	Iwama, Tamaki

- Best bound for $k \geq 5$: $2^{(1-\mu_k/(k-1))n}$ with $\mu_k \approx 1.6$ for large k .

Improved Algorithms: k -SAT

- Decide if given a k -CNF Φ is satisfiable. n , the number of variables is the complexity parameter
- Best known bounds for small values of k : $2^{?n}$

k	unique- k -SAT	k -SAT	k -SAT	k -SAT	k -SAT
3	0.386...	0.521...	0.415...	0.409...	0.404...
4	0.554...	0.562...	0.584...		0.559...
5	0.650...		0.678...		
6	0.711...		0.736...		
	Paturi, Pudlák, Saks, Zane		Schöning	Rolf, ...	Iwama, Tamaki

- Best bound for $k \geq 5$: $2^{(1-\mu_k/(k-1))n}$ with $\mu_k \approx 1.6$ for large k .
- 3-COLORABILITY with the number of vertices as the complexity parameter: $2^{0.41n}$ in polynomial space — [Beigel and Eppstein, 2005](#)
- 4-COLORABILITY: $2^{0.807n}$ in polynomial space — [Bykov, 2004](#)

Time/Success Probability Trade-off

- Most of the algorithms are of **backtracking** or **local search** variety.
- Fact (**Eppstein**): Many of the known deterministic exponential-time algorithms for k -SAT, INDEPENDENT SET, k -COLORABILITY and other problems based on backtracking or local search can be converted to run in **randomized (one-sided error)** polynomial or quasipolynomial time with **success probability inverse of the exponential-time**.

Time/Success Probability Trade-off

- Most of the algorithms are of **backtracking** or **local search** variety.
- Fact (**Eppstein**): Many of the known deterministic exponential-time algorithms for k -SAT, INDEPENDENT SET, k -COLORABILITY and other problems based on backtracking or local search can be converted to run in **randomized (one-sided error)** polynomial or quasipolynomial time with **success probability inverse of the exponential-time**.
- Can we optimally trade uncertainty for time for **SATISFIABILITY** problems? What about for other **NP**-complete problems?

Further Improvements for k -SAT?

- Further improvements for k -SAT, INDEPENDENT SET, and k -COLORABILITY (for fixed k)?

Further Improvements for k -SAT?

- Further improvements for k -SAT, INDEPENDENT SET, and k -COLORABILITY (for fixed k)?
- Do they admit subexponential-time algorithms?

Further Improvements for k -SAT?

- Further improvements for k -SAT, INDEPENDENT SET, and k -COLORABILITY (for fixed k)?
- Do they admit subexponential-time algorithms?
- What are the best possible exponents?

Further Improvements for k -SAT?

- Further improvements for k -SAT, INDEPENDENT SET, and k -COLORABILITY (for fixed k)?
- Do they admit subexponential-time algorithms?
- What are the best possible exponents?
- What are the hardest instances?

Further Improvements for k -SAT?

- Further improvements for k -SAT, INDEPENDENT SET, and k -COLORABILITY (for fixed k)?
- Do they admit subexponential-time algorithms?
- What are the best possible exponents?
- What are the hardest instances?
- Are there algorithms where the exponent does not increase with k ?

Further Improvements for k -SAT?

- Further improvements for k -SAT, INDEPENDENT SET, and k -COLORABILITY (for fixed k)?
- Do they admit subexponential-time algorithms?
- What are the best possible exponents?
- What are the hardest instances?
- Are there algorithms where the exponent does not increase with k ?
- What about improved algorithms for CNF-SAT, CIRCUIT SAT, COLORABILITY, and HAMILTONIAN PATH?

Improved Algorithms: COLORABILITY, HAMILTONIAN PATH, ...

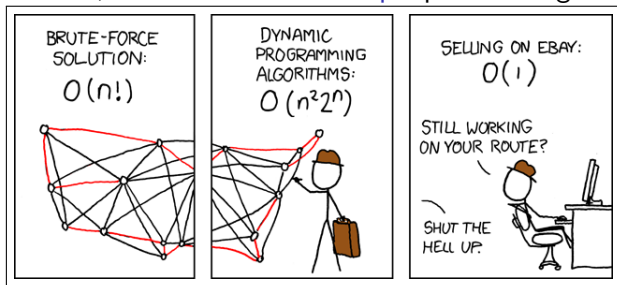
- COLORABILITY (where the number of colors k is given as part of the input): 2^n time and in 2^n space — Björklund, Husfeldt, Kaski, Koivisto 2006-2008

Improved Algorithms: COLORABILITY, HAMILTONIAN PATH, ...

- COLORABILITY (where the number of colors k is given as part of the input): 2^n time and in 2^n space — Björklund, Husfeldt, Kaski, Koivisto 2006-2008
- HAMILTONIAN PATH problem: given graph $G = (V, E)$ on n vertices, does there exist a **simple** path of length $n - 1$?

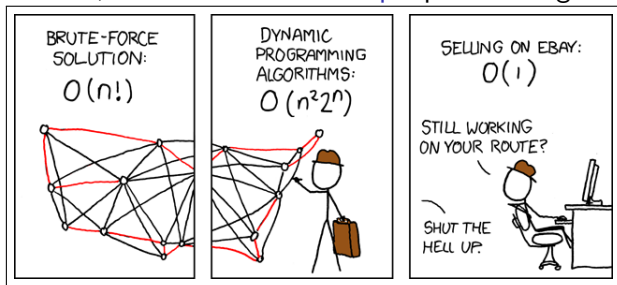
Improved Algorithms: COLORABILITY, HAMILTONIAN PATH, ...

- COLORABILITY (where the number of colors k is given as part of the input): 2^n time and in 2^n space — Björklund, Husfeldt, Kaski, Koivisto 2006-2008
- HAMILTONIAN PATH problem: given graph $G = (V, E)$ on n vertices, does there exist a **simple** path of length $n - 1$?



Improved Algorithms: COLORABILITY, HAMILTONIAN PATH, ...

- COLORABILITY (where the number of colors k is given as part of the input): 2^n time and in 2^n space — Björklund, Husfeldt, Kaski, Koivisto 2006-2008
- HAMILTONIAN PATH problem: given graph $G = (V, E)$ on n vertices, does there exist a **simple** path of length $n - 1$?



- Algorithmic techniques: inclusion-exclusion, dynamic programming

Improved Algorithms: COLORABILITY, HAMILTONIAN PATH, ...

- Improved algorithms if one takes $\log k^n$ or $\log n!$ as the complexity parameter.

Improved Algorithms: COLORABILITY, HAMILTONIAN PATH, ...

- Improved algorithms if one takes $\log k^n$ or $\log n!$ as the complexity parameter.
- We do not know of any improved algorithms for COLORABILITY or HAMILTONIAN PATH when the number of vertices is the complexity parameter.

Improved Algorithms: COLORABILITY, HAMILTONIAN PATH, ...

- Improved algorithms if one takes $\log k^n$ or $\log n!$ as the complexity parameter.
- We do not know of any improved algorithms for COLORABILITY or HAMILTONIAN PATH when the number of vertices is the complexity parameter.
- Similarly, we do not know of any algorithms that run in time less than $2^{n-o(n)}$ for CNF-SAT and CIRCUIT SAT.

Improved Algorithms: COLORABILITY, HAMILTONIAN PATH, ...

- Improved algorithms if one takes $\log k^n$ or $\log n!$ as the complexity parameter.
- We do not know of any improved algorithms for COLORABILITY or HAMILTONIAN PATH when the number of vertices is the complexity parameter.
- Similarly, we do not know of any algorithms that run in time less than $2^{n-o(n)}$ for CNF-SAT and CIRCUIT SAT.
- However, the best-known algorithms for CNF-SAT and CIRCUIT SAT can be converted to run in randomized polynomial time where the success probability is inverse of the exponential-time.

Improved Algorithms: COLORABILITY, HAMILTONIAN PATH, ...

- Improved algorithms if one takes $\log k^n$ or $\log n!$ as the complexity parameter.
- We do not know of any improved algorithms for COLORABILITY or HAMILTONIAN PATH when the number of vertices is the complexity parameter.
- Similarly, we do not know of any algorithms that run in time less than $2^{n-o(n)}$ for CNF-SAT and CIRCUIT SAT.
- However, the best-known algorithms for CNF-SAT and CIRCUIT SAT can be converted to run in randomized polynomial time where the success probability is inverse of the exponential-time.
- On the other hand, we do not know of any randomized polynomial-time algorithms that succeed with probability c^{-n} for COLORABILITY or HAMILTONIAN PATH.

Exact Complexity — Motivation Questions

- Which problems have improved algorithms?
Is there a 2^{cn} algorithm for HAMILTONIAN PATH problem for $c < 1$?

Exact Complexity — Motivation Questions

- Which problems have improved algorithms?
Is there a 2^{cn} algorithm for HAMILTONIAN PATH problem for $c < 1$?
- Can the improvements extend to arbitrarily small exponents?
Is 3-SAT solvable in subexponential-time? How about 3-COLORABILITY?

Exact Complexity — Motivation Questions

- Which problems have improved algorithms?
Is there a 2^{cn} algorithm for HAMILTONIAN PATH problem for $c < 1$?
- 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 — Motivation Questions

- Which problems have improved algorithms?
Is there a 2^{cn} algorithm for HAMILTONIAN PATH problem for $c < 1$?
- 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? If COLORABILITY has a 2^{cn} algorithm, can we prove CNF-SAT has a 2^{dn} algorithm for $c, d < 1$?

Exact Complexity — Motivation Questions

- Which problems have improved algorithms?
Is there a 2^{cn} algorithm for HAMILTONIAN PATH problem for $c < 1$?
- 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? If COLORABILITY has a 2^{cn} algorithm, can we prove CNF-SAT has a 2^{dn} algorithm for $c, d < 1$?

Possibility of Subexponential-time Algorithms

- Possibility of $2^{o(n)}$ algorithm for 3-SAT

Possibility of Subexponential-time Algorithms

- Possibility of $2^{o(n)}$ algorithm for 3-SAT
- **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

Possibility of Subexponential-time Algorithms

- Possibility of $2^{o(n)}$ algorithm for 3-SAT
- **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
- k -SAT and k -COLORABILITY for $k \geq 3$.
VERTEX COVER, CLIQUE, INDEPENDENT SET and k -SET COVER are in **size-constrained SNP**.

Possibility of Subexponential-time Algorithms

- Possibility of $2^{o(n)}$ algorithm for 3-SAT
- **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
- k -SAT and k -COLORABILITY for $k \geq 3$.
VERTEX COVER, CLIQUE, INDEPENDENT SET and k -SET COVER are in **size-constrained SNP**.

Theorem

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

Possibility of Subexponential-time Algorithms

- Possibility of $2^{o(n)}$ algorithm for 3-SAT
- **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
- k -SAT and k -COLORABILITY for $k \geq 3$.
VERTEX COVER, CLIQUE, INDEPENDENT SET and k -SET COVER are in **size-constrained SNP**.

Theorem

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

- Proof requires **reductions that preserve the complexity parameter** (up to a constant factor)

Possibility of Subexponential-time Algorithms

- Possibility of $2^{o(n)}$ algorithm for 3-SAT
- **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
- k -SAT and k -COLORABILITY for $k \geq 3$.
 VERTEX COVER, CLIQUE, INDEPENDENT SET and k -SET COVER are in **size-constrained SNP**.

Theorem

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

- Proof requires **reductions that preserve the complexity parameter** (up to a constant factor)
- Key tools: **Subexponential-time reductions** and **Sparsification Lemma** — Impagliazzo, Paturi, and Zane, 1997

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 $\phi_1, \dots, \phi_s \in k\text{-CNF}$ 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 $\leq O\left(\frac{k}{\epsilon}\right)^{3k}$ times in ϕ_i .

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 $\phi_1, \dots, \phi_s \in k\text{-CNF}$ 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 .

- Proof Sketch: Branch on frequently occurring subclauses rather than just on variables.

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 $\phi_1, \dots, \phi_s \in k\text{-CNF}$ 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 .

- Proof Sketch: Branch on frequently occurring subclauses rather than just on variables.
- Start with small clauses and look for longest subclauses with required frequency

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 $\phi_1, \dots, \phi_s \in k\text{-CNF}$ 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 $\leq O\left(\frac{k}{\epsilon}\right)^{3k}$ times in ϕ_i .

- Proof Sketch: Branch on frequently occurring subclauses rather than just on variables.
- Start with small clauses and look for longest subclauses with required frequency
- Required frequency is roughly inverse exponential in the subclause size.

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 $\phi_1, \dots, \phi_s \in k\text{-CNF}$ 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 .

- Proof Sketch: Branch on frequently occurring subclauses rather than just on variables.
- Start with small clauses and look for longest subclauses with required frequency
- Required frequency is roughly inverse exponential in the subclause size.
- Clause branching results in less information, but the tree does not grow too much.

Completeness of 3-SAT in SNP

Theorem

*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. Complexity parameter is the number of existential quantifiers.
- Reduce k -SAT to **linear-size 3-SAT** using **subexponential-time Turing reductions**. (Sparsification Lemma).

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 make progress by assuming this statement —
Exponential-time Hypothesis (ETH)
- Let $s_k = \inf\{\delta \mid \exists 2^{\delta n} \text{ algorithm for } k\text{-SAT}\}$;

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 make progress by assuming this statement —
Exponential-time Hypothesis (ETH)
- Let $s_k = \inf\{\delta \mid \exists 2^{\delta n} \text{ algorithm for } k\text{-SAT}\}$;
- Define $s_\infty = \lim_{k \rightarrow \infty} s_k$

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 make progress by assuming this statement —
Exponential-time Hypothesis (ETH)
- Let $s_k = \inf\{\delta \mid \exists 2^{\delta n} \text{ algorithm for } k\text{-SAT}\}$;
- Define $s_\infty = \lim_{k \rightarrow \infty} s_k$
- **ETH**: $s_3 > 0$

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 make progress by assuming this statement —
Exponential-time Hypothesis (ETH)
- Let $s_k = \inf\{\delta \mid \exists 2^{\delta n} \text{ algorithm for } k\text{-SAT}\}$;
- Define $s_\infty = \lim_{k \rightarrow \infty} s_k$
- **ETH**: $s_3 > 0$
- What are the consequences of **ETH**?

Complexity of k -SAT with Increasing k

Theorem (Impagliazzo and Paturi, 1999)

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

Complexity of k -SAT with Increasing k

Theorem (Impagliazzo and Paturi, 1999)

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

- More specifically, $s_\infty - s_k \geq d/k$ for some absolute constant $d > 0$.

Complexity of k -SAT with Increasing k

Theorem (Impagliazzo and Paturi, 1999)

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

- More specifically, $s_\infty - s_k \geq d/k$ 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 Paturi, 1999)

If ETH is true, s_k increases infinitely often

- More specifically, $s_\infty - s_k \geq d/k$ 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 k -SAT with Increasing k

Theorem (Impagliazzo and Paturi, 1999)

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

- More specifically, $s_\infty - s_k \geq d/k$ 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.
- Sparsify the k -CNF, **separate the forced variables**, and **express the forced variables as functions of non-forced variables**.

Complexity of k -SAT with Increasing k

Theorem (Impagliazzo and Paturi, 1999)

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

- More specifically, $s_\infty - s_k \geq d/k$ 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.
- Sparsify the k -CNF, **separate the forced variables**, and **express the forced variables as functions of non-forced variables**.
- **Locality** provided by sparsification helps control the width of the resulting formula.

Further Consequences of ETH

- **ETH** implies that $(d, 2)$ -CSP requires d^{cn} time where c is an absolute constant. Traxler 2008

Further Consequences of ETH

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

Further Consequences of ETH

- **ETH** implies that $(d, 2)$ -CSP requires d^{cn} time where c is an absolute constant. [Traxler 2008](#)
- The constant c depends on s_3 .
- $(d, 2)$ -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](#).

Further Consequences of ETH

- **ETH** implies that $(d, 2)$ -CSP requires d^{cn} time where c is an absolute constant. [Traxler 2008](#)
- The constant c depends on s_3 .
- $(d, 2)$ -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](#).
- [Greater expressiveness of \$k'\$ -CNF and \$\(d', 2\)\$ -CSP has been exploited](#).

Parameterized Complexity and Complexity of SATISFIABILITY

- k -DOMINATING SET; given a graph, does there exist a set of k vertices that dominates all vertices?

Parameterized Complexity and Complexity of SATISFIABILITY

- k -DOMINATING SET; given a graph, does there exist a set of k vertices that dominates all vertices?
- Canonical $W[2]$ -complete languages in the hierarchy of parameterized complexity classes.

Parameterized Complexity and Complexity of SATISFIABILITY

- k -DOMINATING SET; given a graph, does there exist a set of k vertices that dominates all vertices?
- Canonical $W[2]$ -complete languages in the hierarchy of parameterized complexity classes.
- Does k -DOMINATING SET have a $n^{k-\epsilon}$ time algorithm for $\epsilon > 0$?

Parameterized Complexity and Complexity of SATISFIABILITY

- k -DOMINATING SET; given a graph, does there exist a set of k vertices that dominates all vertices?
- Canonical $W[2]$ -complete languages in the hierarchy of parameterized complexity classes.
- Does k -DOMINATING SET have a $n^{k-\epsilon}$ time algorithm for $\epsilon > 0$?
- Positive answer implies an improved algorithm for CNF-SAT — Patrascu and Williams, 2009
- For $d < N^{0.99}$, if d -SUM problem over N numbers of $O(d \log N)$ bits can be solved in $N^{o(d)}$ time, then 3-SAT can be solved in $2^{o(n)}$ time. — Patrascu and Williams, 2009

Parameterized Complexity and Complexity of SATISFIABILITY

- k -DOMINATING SET; given a graph, does there exist a set of k vertices that dominates all vertices?
- Canonical $W[2]$ -complete languages in the hierarchy of parameterized complexity classes.
- Does k -DOMINATING SET have a $n^{k-\epsilon}$ time algorithm for $\epsilon > 0$?
- Positive answer implies an improved algorithm for CNF-SAT — Patrascu and Williams, 2009
- For $d < N^{0.99}$, if d -SUM problem over N numbers of $O(d \log N)$ bits can be solved in $N^{o(d)}$ time, then 3-SAT can be solved in $2^{o(n)}$ time. — Patrascu and Williams, 2009
- **ETH** implies that CSP for instances whose primal graphs have treewidth k requires time $n^{\Omega(k/\log k)}$ where n is the size of the instance. — Marx 2008

Hardest Instances of k -SAT

- Which instances of k -SAT are among the hardest?

Hardest Instances of k -SAT

- Which instances of k -SAT are among the hardest?
 - ① Instances where each variable occurs with frequency $O(k^{3k})$.
(Sparsification Lemma)

Hardest Instances of k -SAT

- Which instances of k -SAT are among the hardest?
 - ① Instances where each variable occurs with frequency $O(k^{3k})$. (Sparsification Lemma)
 - ② Instances with at most one satisfying assignment. More specifically, $s_k \leq \sigma_k + O(\log^2 k/k)$. — Calbro, Impagliazzo, Kabanets, and Paturi, 2001

Hardest Instances of k -SAT

- Which instances of k -SAT are among the hardest?
 - ① Instances where each variable occurs with frequency $O(k^{3k})$. (Sparsification Lemma)
 - ② Instances with at most one satisfying assignment. More specifically, $s_k \leq \sigma_k + O(\log^2 k/k)$. — Calabro, Impagliazzo, Kabanets, and Paturi, 2001
 - ③ Conversely, more solutions can speed up the algorithm. $(2^n/S)^{1-1/k}$ algorithm for k -SAT where S is the number of solutions. — Paturi, Pudlák, Zane, 1997

Hardest Instances of k -SAT

- Which instances of k -SAT are among the hardest?
 - ① Instances where each variable occurs with frequency $O(k^{3k})$. (Sparsification Lemma)
 - ② Instances with at most one satisfying assignment. More specifically, $s_k \leq \sigma_k + O(\log^2 k/k)$. — Calabro, Impagliazzo, Kabanets, and Paturi, 2001
 - ③ Conversely, more solutions can speed up the algorithm. $(2^n/S)^{1-1/k}$ algorithm for k -SAT where S is the number of solutions. — Paturi, Pudlák, Zane, 1997
 - ④ As k increases, k -SAT gets harder.

Duality between Clause Width and Density

- Let Δ -SAT denote the problem of deciding the satisfiability of CNF where the number of clauses to variables ratio is at most Δ .

Duality between Clause Width and Density

- Let Δ -SAT denote the problem of deciding the satisfiability of CNF where the number of clauses to variables ratio is at most Δ .
- Let $s_{\Delta\text{-SAT}}$ denote the best exponent for solving Δ -SAT.

Duality between Clause Width and Density

- Let Δ -SAT denote the problem of deciding the satisfiability of CNF where the number of clauses to variables ratio is at most Δ .
- Let $s_{\Delta\text{-SAT}}$ denote the best exponent for solving Δ -SAT.

Duality between Clause Width and Density

- Let Δ -SAT denote the problem of deciding the satisfiability of CNF where the number of clauses to variables ratio is at most Δ .
- Let $s_{\Delta\text{-SAT}}$ denote the best exponent for solving Δ -SAT.

Theorem (Calabro, Impagliazzo, Paturi, 2006)

$\forall k, \epsilon > 0, \exists \Delta \in 2^{O(k \log k)}$ such that $s_k \leq s_{\Delta\text{-SAT}} + \epsilon$.

Duality between Clause Width and Density

- Let Δ -SAT denote the problem of deciding the satisfiability of CNF where the number of clauses to variables ratio is at most Δ .
- Let $s_{\Delta\text{-SAT}}$ denote the best exponent for solving Δ -SAT.

Theorem (Calabro, Impagliazzo, Paturi, 2006)

$\forall k, \epsilon > 0, \exists \Delta \in 2^{O(k \log k)}$ such that $s_k \leq s_{\Delta\text{-SAT}} + \epsilon$.

Theorem (Calabro, Impagliazzo, Paturi, 2006)

ETH implies $\forall \Delta, \exists l$ such that $s_{\Delta\text{-SAT}} \leq s_l - O\left(\frac{1}{\log \Delta}\right)$.

Duality between Clause Width and Density

- Let Δ -SAT denote the problem of deciding the satisfiability of CNF where the number of clauses to variables ratio is at most Δ .
- Let $s_{\Delta\text{-SAT}}$ denote the best exponent for solving Δ -SAT.

Theorem (Calabro, Impagliazzo, Paturi, 2006)

$\forall k, \epsilon > 0, \exists \Delta \in 2^{O(k \log k)}$ such that $s_k \leq s_{\Delta\text{-SAT}} + \epsilon$.

Theorem (Calabro, Impagliazzo, Paturi, 2006)

ETH implies $\forall \Delta, \exists l$ such that $s_{\Delta\text{-SAT}} \leq s_l - O(\frac{1}{\log \Delta})$.

- k -CNF correspond roughly to exponential density (in k) CNFs.

Duality between Clause Width and Density

- Let Δ -SAT denote the problem of deciding the satisfiability of CNF where the number of clauses to variables ratio is at most Δ .
- Let $s_{\Delta\text{-SAT}}$ denote the best exponent for solving Δ -SAT.

Theorem (Calabro, Impagliazzo, Paturi, 2006)

$\forall k, \epsilon > 0, \exists \Delta \in 2^{O(k \log k)}$ such that $s_k \leq s_{\Delta\text{-SAT}} + \epsilon$.

Theorem (Calabro, Impagliazzo, Paturi, 2006)

ETH implies $\forall \Delta, \exists l$ such that $s_{\Delta\text{-SAT}} \leq s_l - O(\frac{1}{\log \Delta})$.

- k -CNF correspond roughly to exponential density (in k) CNFs.
- **ETH** implies $s_{\Delta\text{-SAT}}$ increases with Δ

Duality between Clause Width and Density

- Let Δ -SAT denote the problem of deciding the satisfiability of CNF where the number of clauses to variables ratio is at most Δ .
- Let $s_{\Delta\text{-SAT}}$ denote the best exponent for solving Δ -SAT.

Theorem (Calabro, Impagliazzo, Paturi, 2006)

$\forall k, \epsilon > 0, \exists \Delta \in 2^{O(k \log k)}$ such that $s_k \leq s_{\Delta\text{-SAT}} + \epsilon$.

Theorem (Calabro, Impagliazzo, Paturi, 2006)

ETH implies $\forall \Delta, \exists l$ such that $s_{\Delta\text{-SAT}} \leq s_l - O(\frac{1}{\log \Delta})$.

- k -CNF correspond roughly to exponential density (in k) CNFs.
- **ETH** implies $s_{\Delta\text{-SAT}}$ increases with Δ
- $s_{\infty} = s_{\infty\text{-SAT}}$

Complexity of CIRCUIT SAT

- Consider **natural, though restricted**, models of computation
- **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.

Complexity of CIRCUIT SAT

- Consider **natural, though restricted**, models of computation
- **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

Complexity of CIRCUIT SAT

- Consider **natural, though restricted**, models of computation
- **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

Complexity of CIRCUIT SAT

- Consider **natural, though restricted**, models of computation
- **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

Complexity of CIRCUIT SAT

- Consider **natural, though restricted**, models of computation
- **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**($T(n, m)$)?

Complexity of CIRCUIT SAT

- Consider **natural, though restricted**, models of computation
- **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**($T(n, m)$)?
- CIRCUIT SAT problem can be solved with probability $2^{-n+O(\lg T(n, m))}$ **OP**($T(n, m)$). Best-known deterministic algorithm is of the form $2^n \text{poly}(m)$.

Complexity of CIRCUIT SAT

- Consider **natural, though restricted**, models of computation
- **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($T(n, m)$)?
- CIRCUIT SAT problem can be solved with probability $2^{-n+O(\lg T(n, m))}$ OP($T(n, m)$). Best-known deterministic algorithm is of the form $2^n \text{poly}(m)$.
- Hamiltonian path problem can be solved with probability $1/n!$ in OPP.

Time and Success Probability

- Consider $\lg t + \lg 1/p$ where p is the best success probability for time t .

Time and Success Probability

- Consider $\lg t + \lg 1/p$ where p is the best success probability for time t .
- How does this quantity behave with as a function of t ?
- For what problems, does this quantity decrease/stay the same over a certain range of time?

Time and Success Probability

- Consider $\lg t + \lg 1/p$ where p is the best success probability for time t .
- How does this quantity behave with as a function of t ?
- For what problems, does this quantity decrease/stay the same over a certain range of time?
- CIRCUIT SAT versus HAMILTONIAN PATH

Time and Success Probability

- Consider $\lg t + \lg 1/p$ where p is the best success probability for time t .
- How does this quantity behave with as a function of t ?
- For what problems, does this quantity decrease/stay the same over a certain range of time?
- CIRCUIT SAT versus HAMILTONIAN PATH
- We present partial evidence that for CIRCUIT SAT, $(\log t + \log 1/p)/n$ may not decrease with increasing time.

Results: Polynomial Size Circuits

Theorem

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 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 $O(\text{poly}(m)n^{O(\lg \lg m)})$.

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 $O(\text{poly}(m)n^{O(\lg \lg m)})$.

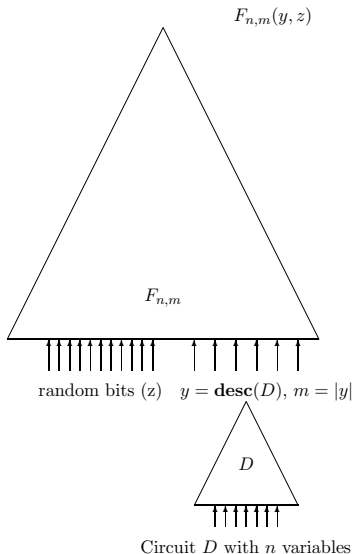
- The consequence is very close to the statement $\mathbf{NP} \subseteq \mathbf{P}/\text{poly}$.

Results: Subexponential Size Circuits

Theorem

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 $2^{o(n)} \text{poly}(m)$.

Circuit Family for deciding CIRCUIT SAT



Exponential Amplification Lemma

Lemma (Paturi and Pudlák 2010)

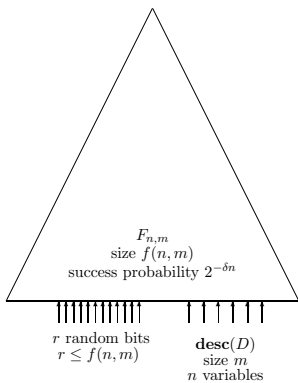
Exponential Amplification Lemma: *Let \mathcal{F} be an f -bounded family for some $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R}$ such that the success probability is $2^{-\delta n}$ for $0 < \delta < 1$. Then there exists a g -bounded circuit family \mathcal{G} such that $E_{\text{CIRCUIT SAT}}(\mathcal{G}) < \delta^2$ where $g(n, m) = O(f(\lceil \delta n \rceil + 5, \tilde{O}(f(n, m))))$.*

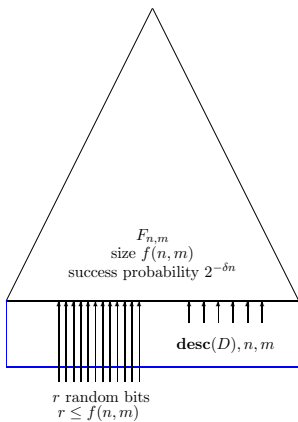
Exponential Amplification Lemma

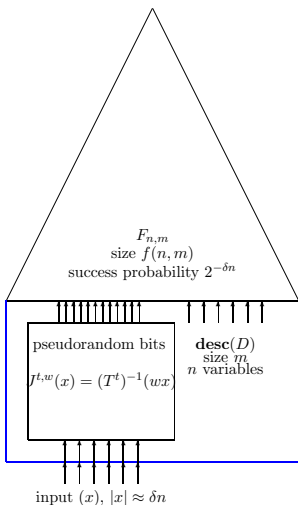
Lemma (Paturi and Pudlák 2010)

Exponential Amplification Lemma: *Let \mathcal{F} be an f -bounded family for some $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R}$ such that the success probability is $2^{-\delta n}$ for $0 < \delta < 1$. Then there exists a g -bounded circuit family \mathcal{G} such that $E_{\text{CIRCUIT SAT}}(\mathcal{G}) < \delta^2$ where $g(n, m) = O(f(\lceil \delta n \rceil + 5, \tilde{O}(f(n, m))))$.*

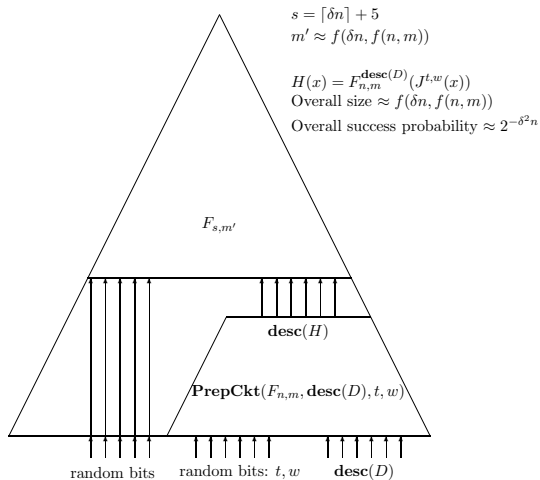
$\mathcal{F} : (f(n, m), \delta n) \rightarrow \mathcal{G} : (g(n, m), \delta^2 n)$

Picture 1: Probabilistic Circuit $F_{n,m}$

Picture 1: Specialization of $F_{n,m}$



Picture 1: $H(x) = F_{n,m}^{\text{desc}(D)}(J^{t,w}(x))$

Picture 1: Circuit $G_{n,m}$

Useful Messages

- Reductions among instances with careful attention to parameters. Width, density, frequency, number of variables, size, ...
- Branching that produces less information reduces the size of the tree. Sparsification Lemma
- Limitations of algorithmic techniques. Conditional lower bounds for CIRCUIT SAT

Open Problems

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

Thank You