12. Exponential Time Hypothesis
COMP6741: Parameterized and Exact Computation

Serge Gaspers[1,2]

[1]School of Computer Science and Engineering, UNSW Australia
[2]Data61, Decision Sciences Group, CSIRO

Semester 2, 2016

Outline

Outline

1. SAT and k-SAT
2. Subexponential time algorithms
3. ETH and SETH
4. Algorithmic lower bounds based on ETH
5. Algorithmic lower bounds based on SETH
6. Further Reading

SAT

SAT	
Input:	A propositional formula F in conjunctive normal form (CNF)
Parameter:	$n = \|\mathsf{var}(F)\|$, the number of variables in F
Question:	Is there an assignment to $\mathsf{var}(F)$ satisfying all clauses of F?

k-SAT	
Input:	A CNF formula F where each clause has length at most k
Parameter:	$n = \|\mathsf{var}(F)\|$, the number of variables in F
Question:	Is there an assignment to $\mathsf{var}(F)$ satisfying all clauses of F?

Example:

$$(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$$

Algorithms for SAT

- Brute-force: $O^*(2^n)$

Algorithms for SAT

- Brute-force: $O^*(2^n)$
- ... after > 50 years of SAT solving
  (SAT association, SAT conference, JSAT journal, annual SAT competitions, ...)

Algorithms for SAT

- Brute-force: $O^*(2^n)$
- ... after > 50 years of SAT solving
(SAT association, SAT conference, JSAT journal, annual SAT competitions, ...)
- fastest known algorithm for SAT: $O^*(2^{n\cdot(1-1/O(\log m/n))})$, where m is the number of clauses [Calabro, Impagliazzo, Paturi, 2006] [Dantsin, Hirsch, 2009]
- However: no $O^*(1.9999^n)$ time algorithm is known
- fastest known algorithms for 3-SAT: $O^*(1.3303^n)$ deterministic [Makino, Tamaki, Yamamoto, 2013] and $O^*(1.3071^n)$ randomized [Hertli, 2014]

Algorithms for SAT

- Brute-force: $O^*(2^n)$
- ... after > 50 years of SAT solving
(SAT association, SAT conference, JSAT journal, annual SAT competitions, ...)
- fastest known algorithm for SAT: $O^*(2^{n\cdot(1-1/O(\log m/n))})$, where m is the number of clauses [Calabro, Impagliazzo, Paturi, 2006] [Dantsin, Hirsch, 2009]
- However: no $O^*(1.9999^n)$ time algorithm is known
- fastest known algorithms for 3-SAT: $O^*(1.3303^n)$ deterministic [Makino, Tamaki, Yamamoto, 2013] and $O^*(1.3071^n)$ randomized [Hertli, 2014]
- Could it be that 3-SAT cannot be solved in $2^{o(n)}$ time?
- Could it be that SAT cannot be solved in $O^*((2-\epsilon)^n)$ time for any $\epsilon > 0$?

Outline

NP-hard problems in subexponential time?

- Are there any NP-hard problems that can be solved in $2^{o(n)}$ time?

NP-hard problems in subexponential time?

- Are there any NP-hard problems that can be solved in $2^{o(n)}$ time?
- **Yes**. For example, INDEPENDENT SET is NP-comlpete even when the input graph is planar (can be drawn in the plane without edge crossings). Planar graphs have treewidth $O(\sqrt{n})$ and tree decompositions of that width can be found in polynomial time ("Planar separator theorem" [Lipton, Tarjan, 1979]). Using a tree decomposition based algorithm, INDEPENDENT SET can be solved in $2^{O(\sqrt{n})}$ time on planar graphs.

Outline

ETH and SETH

Definition 1

For each $k \geq 3$, define δ_k to be the infinimum[1] of the set of constants c such that k-SAT can be solved in $O^*(2^{c \cdot n})$ time.

Conjecture 2 (Exponential Time Hyphothesis (ETH))

$\delta_3 > 0$.

Conjecture 3 (Strong Exponential Time Hyphothesis (SETH))

$\lim_{k \to \infty} \delta_k = 1$.

Notes: (1) ETH $\Rightarrow$ 3-SAT cannot be solved in $2^{o(n)}$ time.
SETH $\Rightarrow$ SAT cannot be solved in $O^*((2-\epsilon)^n)$ time for any $\epsilon > 0$.

[1] The infinimum of a set of numbers is the largest number that is smaller or equal to each number in the set. E.g., the infinimum of $\{\varepsilon \in \mathbb{R} : \varepsilon > 0\}$ is 0.

Outline

Algorithmic lower bounds based on ETH

- Suppose ETH is true
- Can we infer lower bounds on the running time needed to solve other problems?

Algorithmic lower bounds based on ETH

- Suppose ETH is true
- Can we infer lower bounds on the running time needed to solve other problems?
- Suppose there is a polynomial-time reduction from 3-SAT to a graph problem Π, which constructs an equivalent instance where the number of vertices of the output graph equals the number of variables of the input formula, $|V| = |\mathsf{var}(F)|$.
- Using the reduction, we can conclude that, if Π has an $O^*(2^{o(|V|)})$ time algorithm, then 3-SAT has an $O^*(2^{o(|\mathsf{var}(F)|)})$ time algorithm, contradicting ETH.
- Therefore, we conclude that Π has no $O^*(2^{o(|V|)})$ time algorithm unless ETH fails.

Sparsification Lemma

Issue: Many reductions from 3-SAT create a number of vertices / variables / elements that are related to the number of clauses of the 3-SAT instance.

Sparsification Lemma

Issue: Many reductions from 3-SAT create a number of vertices / variables / elements that are related to the number of clauses of the 3-SAT instance.

Theorem 4 (Sparsification Lemma, [Impagliazzo, Paturi, Zane, 2001])

For each $\varepsilon > 0$ and positive integer k, there is a $O^*(2^{\varepsilon \cdot n})$ time algorithm that takes as input a k-CNF formula F with n variables and outputs an equivalent formula $F' = \bigvee_{i=1}^{t} F_i$ that is a disjunction of $t \leq 2^{\varepsilon n}$ formulas F_i with $\mathrm{var}(F_i) = \mathrm{var}(F)$ and $|\mathrm{cla}(F_i)| = O(n)$.

3-SAT with a linear number of clauses

Corollary 5

ETH $\Rightarrow$ 3-SAT cannot be solved in $O^*(2^{o(n+m)})$ time where m denotes the number of clauses of F.

Observation: Let A, B be parameterized problems and f, g be non-decreasing functions.
Suppose there is a polynomial-parameter transformation from A to B such that if the parameter of an instance of A is k, then the parameter of the constructed instance of B is at most $g(k)$. Then an $O^*(2^{o(f(k))})$ time algorithm for B implies an $O^*(2^{o(f(g(k)))})$ time algorithm for A.

More general reductions are possible

Definition 6 (SERF-reduction)

A SubExponential Reduction Family from a parameterized problem A to a parameterized problem B is a family of Turing reductions from A to B (i.e., an algorithm for A, making queries to an oracle for B that solves any instance for B in constant time) for each $\varepsilon > 0$ such that

- for every instance I for A with parameter k, the running time is $O^*(2^{\varepsilon k})$, and
- for every query I' to B with parameter k', we have that $k' \in O(k)$ and $|I'| = |I|^{O(1)}$.

Note: If A is SERF-reducible to B and A has no $2^{o(k)}$ time algorithm, then B has no $2^{o(k')}$ time algorithm.

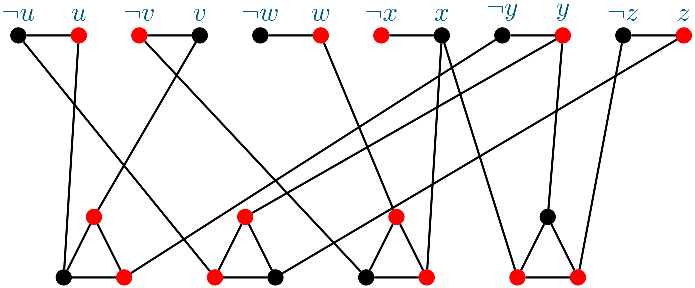
Vertex Cover has no subexponential algorithm

Vertex Cover has no subexponential algorithm

Polynomial-parameter transformation from 3-SAT.
For simplicity, assume all clauses have length 3.
3-CNF Formula $F = (u \vee v \vee \neg y) \wedge (\neg u \vee y \vee z) \wedge (\neg v \vee w \vee x) \wedge (x \vee y \vee \neg z)$

Vertex Cover has no subexponential algorithm

Polynomial-parameter transformation from 3-SAT.
For simplicity, assume all clauses have length 3.
3-CNF Formula $F = (u \vee v \vee \neg y) \wedge (\neg u \vee y \vee z) \wedge (\neg v \vee w \vee x) \wedge (x \vee y \vee \neg z)$



For a 3-CNF formula with n variables and m clauses, we create a VERTEX COVER instance with $|V| = 2n + 3m$, $|E| = n + 6m$, and $k = n + 2m$.

Vertex Cover has no subexponential algorithm II

Theorem 7

ETH $\Rightarrow$ Vertex Cover *has no* $2^{o(|V|)}$ *time algorithm.*

Theorem 8

ETH $\Rightarrow$ Vertex Cover *has no* $2^{o(|E|)}$ *time algorithm.*

Theorem 9

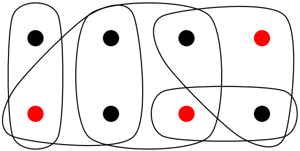
ETH $\Rightarrow$ Vertex Cover *has no* $2^{o(k)}$ *time algorithm.*

Outline

Hitting Set

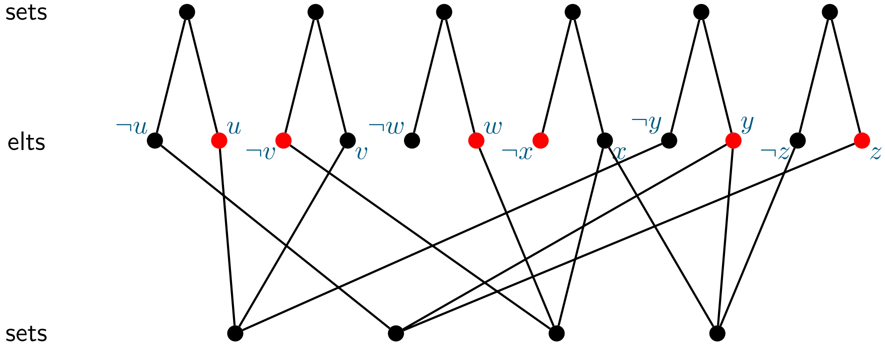
Recall: A hitting set of a set system $\mathcal{S} = (V, H)$ is a subset X of V such that X contains at least one element of each set in H, i.e., $X \cap Y \neq \emptyset$ for each $Y \in H$.

elts-HITTING SET	
Input:	A set system $\mathcal{S} = (V, H)$ and an integer k
Parameter:	$n = \|V\|$
Question:	Does $\mathcal{S}$ have a hitting set of size at most k?



SETH-lower bound for Hitting Set

CNF Formula $F = (u \vee v \vee \neg y) \wedge (\neg u \vee y \vee z) \wedge (\neg v \vee w \vee x) \wedge (x \vee y \vee \neg z)$
Inidence graph of equivalent Hitting Set instance:



For a CNF formula with n variables and m clauses, we create a Hitting Set instance with $|V| = 2n$ and $k = n$.

SETH-lower bound for Hitting Set

Theorem 10

SETH $\Rightarrow$ Hitting Set *has no* $O^*((2-\varepsilon)^{|V|/2})$ *time algorithm for any* $\varepsilon > 0$.

Note: With a more ingenious reduction, one can show that Hitting Set has no $O^*((2-\varepsilon)^{|V|})$ time algorithm for any $\varepsilon > 0$ under SETH.

Exercise

A dominating set of a graph $G = (V, E)$ is a set of vertices $S \subseteq V$ such that $N_G[S] = V$.

vertex-DOMINATING SET	
Input:	A graph $G = (V, E)$ and an integer k
Parameter:	$n = \|V\|$
Question:	Does G have a dominating set of size at most k?

- Prove that ETH $\Rightarrow$ vertex-DOMINATING SET has no $2^{o(n)}$ time algorithm.

Outline

Further Reading

- Chapter 14, *Lower bounds based on the Exponential-Time Hypothesis* in Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, MichałPilipczuk, and Saket Saurabh. Parameterized Algorithms. Springer, 2015.
- Section 11.3, *Subexponential Algorithms and ETH* in Fedor V. Fomin and Dieter Kratsch. Exact Exponential Algorithms. Springer, 2010.
- Section 29.5, *The Sparsification Lemma* in Rodney G. Downey and Michael R. Fellows. Fundamentals of Parameterized Complexity. Springer, 2013.