12. Exponential Time Hypothesis
COMP6741: Parameterized and Exact Computation
Serge Gaspers
Semester 2, 2017

Contents

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

1 SAT and k-SAT

SAT

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

k-SAT

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

Example:

$$(x_1 \lor x_2) \land (\neg x_2 \lor x_3 \lor \neg x_4) \land (x_1 \lor x_4) \land (\neg x_1 \lor \neg x_3 \lor \neg x_4)$$

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: $O^*(2^{n^{(1-1/O(\log m/n))}})$, 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 $\epsilon > 0$?

2  Subexponential time algorithms

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-complete 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.

3  ETH and SETH

Definition 1. For each $k \geq 3$, define $\delta_k$ 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 Hypothesis (ETH)). $\delta_3 > 0$.

Conjecture 3 (Strong Exponential Time Hypothesis (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 $\epsilon > 0$.

4  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 $\Pi$, which constructs an equivalent instance where the number of vertices of the output graph equals the number of variables of the input formula, $|V| = |\text{var}(F)|$.

• Using the reduction, we can conclude that, if $\Pi$ has an $O^*(2^{o(|V|)})$ time algorithm, then 3-SAT has an $O^*(2^{o(|\text{var}(F)|)})$ time algorithm, contradicting ETH.

• Therefore, we conclude that $\Pi$ 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.

Theorem 4 (Sparsification Lemma, [Impagliazzo, Paturi, Zane, 2001]). For each $\epsilon > 0$ and positive integer $k$, there is a $O^*(2^{\epsilon n})$ 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 $t \leq 2^{\epsilon n}$ formulas $F_i$ with $\text{var}(F_i) = \text{var}(F)$ and $|\text{cla}(F_i)| = O(n)$.

\(^1\)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 $\{\epsilon \in \mathbb{R} : \epsilon > 0\}$ is 0.
Corollary 5. \( \text{ETH} \Rightarrow 3 \text{-SAT cannot be solved in } O^*(2^{o(n+m)}) \text{ time where } m \text{ denotes the number of clauses of } F. \)

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 \( O^*(2^{o(g(f(k)))}) \) 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

Polynomial-parameter transformation from 3-SAT.

For simplicity, assume all clauses have length 3.

3-CNF Formula \( F = (u \lor v \lor \neg y) \land (\neg u \lor y \lor z) \land (\neg y \lor w \lor x) \land (x \lor y \lor \neg z) \)

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 \( k = n + 2m \).

Theorem 7. \( \text{ETH} \Rightarrow \text{VERTEX COVER has no } 2^{o(|V|)} \text{ time algorithm.} \)

Theorem 8. \( \text{ETH} \Rightarrow \text{VERTEX COVER has no } 2^{o(|E|)} \text{ time algorithm.} \)

Theorem 9. \( \text{ETH} \Rightarrow \text{VERTEX COVER has no } 2^{o(k)} \text{ time algorithm.} \)

5 Algorithmic lower bounds based on SETH

Hitting Set

Recall: A hitting set of a set system \( S = (V, H) \) 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 \( Y \in H \).

<table>
<thead>
<tr>
<th>\text{elts-Hitting Set}</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A set system ( S = (V, H) ) and an integer ( k )</td>
</tr>
<tr>
<td><strong>Parameter:</strong> ( n =</td>
</tr>
<tr>
<td><strong>Question:</strong> Does ( S ) have a hitting set of size at most ( k )?</td>
</tr>
</tbody>
</table>
SETH-lower bound for Hitting Set

CNF Formula $F = (u \lor v \lor \neg y) \land (\neg u \lor y \lor z) \land (\neg v \lor w \lor x) \land (x \lor y \lor \neg z)$

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 $k = n$.

**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**

<table>
<thead>
<tr>
<th>Input</th>
<th>A graph $G = (V,E)$ and an integer $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>$n =</td>
</tr>
<tr>
<td>Question</td>
<td>Does $G$ have a dominating set of size at most $k$?</td>
</tr>
</tbody>
</table>

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

**Solution idea**

Reduce from 3-SAT. For each $x \in \text{var}(F)$, create a triangle with vertices $x$, $\neg x$ and $d_x$. For each $c \in \text{cla}(F)$, create a vertex $c$ adjacent to all the vertices whose corresponding literals are contained in the clause $c$.

6 Further Reading