

# Computational Complexity

## Lecture 12: Subexponential-time complexity and the ETH

Ronald de Haan

me@ronalddehaan.eu

University of Amsterdam

May 17, 2023

- Approximation algorithms
- Limits of approximation algorithms
- PCP Theorem

## What will we do today?

- Consider exponential-time and subexponential-time algorithms
- A new assumption: ETH
- Use this assumption to derive exponential-time lower bounds

## Our favorite example: 3SAT

- Let's find some exponential-time algorithms for 3SAT
- Take some 3CNF formula  $\varphi = c_1 \wedge \dots \wedge c_m$  with  $\text{var}(\varphi) = \{x_1, \dots, x_n\}$ .
- Consider this naive algorithm:
  - Iterate over all truth assignments  $\alpha : \text{var}(\varphi) \rightarrow \{0, 1\}$
  - If  $\alpha$  satisfies  $\varphi$ , for some  $\alpha$ , return 1; otherwise, return 0
- This algorithm takes time  $2^n \cdot O(m^c)$ , for some  $c \in \mathbb{N}$
- Can we do better?

## Our favorite example: 3SAT (ct'd)

$A_{\text{recursive}}(\varphi)$ :

```
if  $\varphi$  contains only clauses of size at most 2 then
  | decide if  $\varphi$  is satisfiable in polynomial time, and return the answer;
else
  | take some clause  $c_j$  in  $\varphi$  of size 3;
  | for each of the 7 truth assignments  $\alpha$  to  $\text{var}(c_j)$  that satisfy  $c_j$  do
  |   | if  $A_{\text{recursive}}(\varphi[\alpha]) = 1$  then
  |   |   | return 1;
  |   |   end
  |   end
  | return 0;
end
```

- This algorithm  $A_{\text{recursive}}$  takes time  $1.92^n \cdot O(m^c)$ , for some  $c \in \mathbb{N}$ 
  - Recursion tree has branching factor 7 and depth  $n/3$ , so is of size  $O(7^{n/3}) = O(1.92^n)$
- Can we keep improving the base of the exponential? Is there some limit?

## Functions between polynomial and exponential

exponential-time

$$2^n \cdot m$$

-----  
"ETH"

subexponential-time

$$n^{\log n} \cdot m^2, 2^{\sqrt{n}} \cdot m, \text{ etc.}$$

-----  
"P  $\neq$  NP"

polynomial-time

$$2m, m^2, n \cdot m^2, \text{ etc.}$$

- The assumption  $P \neq NP$  is not enough to rule out subexponential-time algorithms for NP-complete problems
  - Typical strategy to rule out polynomial-time algorithms:
    - Take some NP-complete  $L$ .
    - Assume  $P \neq NP$ .
    - Suppose that  $L$  is solvable in polynomial time.
    - Then  $P = NP$ . ⚡
- only works for polynomial time

## The Exponential-Time Hypothesis (ETH)

### Definition ( $\delta_q$ )

For  $q \geq 3$ , let  $\delta_q$  be the infimum of the set of constants  $c$  for which there exists an algorithm solving  $q$ -SAT in time  $O(2^{cn}) \cdot m^{O(1)}$ , where  $n$  is the number of variables in the  $q$ -SAT input and  $m$  the number of clauses.

### Definition (Exponential-Time Hypothesis; ETH)

*Exponential-Time Hypothesis* (unproven conjecture):  $\delta_3 > 0$ .



- The ETH implies that there is no  $2^{o(n)}$ -time algorithm for 3SAT:
  - Suppose that some  $2^{o(n)}$ -time algorithm  $A$  for 3SAT exists.
  - Suppose also that the ETH is true:  $\delta_3 > 0$ .
  - Then there is some  $c$  such that no  $2^{cn} \cdot m^{O(1)}$ -time algorithm for 3SAT exists.
  - For large enough  $n$ ,  $A$  runs in time  $2^{cn} \cdot m^{O(1)}$ .  $\nexists$
- So we can solve 3SAT in time  $2^{O(n)}$ , but—assuming the ETH—not in time  $2^{o(n)}$ .
  - E.g., not in time  $2^{O(n/\log n)}$ ,  $2^{O(\sqrt{n})}$  or  $n^{O(\log n)}$ .
- The ETH implies  $P \neq NP$ —or in other words:  $P = NP$  implies that the ETH is false

## Showing ETH-based lower bounds for other problems

- Take VC as example—solvable in time  $2^{O(v)}$ , where  $v$  is the number of vertices.
- Can we show a matching lower bound—i.e., VC not solvable in time  $2^{o(v)}$ ?
- Idea:
  - Use reduction from 3SAT to VC
  - $v$  of VC needs to increase at most linearly in  $n$  of 3SAT
  - In the reduction that we have,  $v$  is linear in  $n + m$
- ▶ Suppose VC is solvable in time  $2^{o(v)}$  using some algorithm  $A$
- ▶ Idea to construct a  $2^{o(n)}$ -time algorithm for 3SAT:
  - ▶ use reduction from 3SAT to VC
  - ▶ then run  $A$  to solve the resulting VC instance
- ▶ Only works in time  $2^{o(n)}$  if  $v$  is linear in  $n$ .

## Sparsification Lemma

For each  $\epsilon > 0$ , there is a constant  $\kappa(\epsilon)$  such that every 3CNF formula  $\varphi$  with  $n$  variables and  $m$  clauses can be expressed as:

$$\varphi \equiv \bigvee_{i=1}^t \psi_i,$$

where  $t \leq 2^{\epsilon n}$  and each  $\psi_i$  is a 3CNF formula on the same variables as  $\varphi$  and with  $\kappa(\epsilon) \cdot n$  clauses.

Moreover, this disjunction  $\bigvee_{i=1}^t \psi_i$  can be computed in time  $2^{\epsilon n} \cdot m^{O(1)}$ .

## Assuming the ETH, 3SAT cannot be solved in time $2^{o(n+m)}$

- Assume the ETH, i.e.,  $\delta_3 > 0$ .
- Suppose that 3SAT can be solved in time  $2^{o(n+m)}$  with some algorithm  $A$ .
- Take some  $c$  with  $0 < c < \delta_3$ .
- We will show that 3SAT is solvable in time  $2^{cn} \cdot m^{O(1)}$ :
  - Take some 3CNF formula  $\varphi$  with  $n$  variables and  $m$  clauses.
  - Let  $\epsilon = c/2$ .
  - Construct the  $\psi_i$ 's from the Sparsification Lemma (using the value  $\epsilon = c/2$ )
  - Run the algorithm  $A$  on these  $\psi_i$ 's.
  - Return 1 if some  $\psi_i$  is satisfiable; return 0 otherwise.
  - This runs in time  $2^{cn} \cdot m^{O(1)}$ . ⚡
    - For large enough  $n$ , running  $A$  on  $\psi_i$  takes time  $2^{\epsilon n} m^{O(1)}$  – since  $|\psi_i|$  is linear in  $n$ .

## Lower bound for VC using the ETH

- Suppose VC is solvable in time  $2^{o(v)}$  using some algorithm  $A$ , where  $v$  is the number of vertices.
- Idea to construct a  $2^{o(n+m)}$ -time algorithm for 3SAT:
  - Take some 3CNF formula  $\varphi$
  - Use polynomial-time reduction  $R$  from 3SAT to VC:  
 $R(\varphi) = (G, k)$  with  $G = (V, E)$ , where  $v = |V| = O(n + m)$
  - Then run  $A$  to decide if  $G$  has a vertex cover of size  $k$   
(which is the case if and only if  $\varphi$  is satisfiable)
  - This runs in time  $|\varphi|^{O(1)} + 2^{o(v)} = 2^{o(n+m)}$ .  $\nexists$
- So, assuming the ETH, there is no  $2^{o(v)}$ -time algorithm for VC.

## Strong Exponential-Time Hypothesis (SETH)

### Definition ( $\delta_q$ ; repeated)

For  $q \geq 3$ , let  $\delta_q$  be the infimum of the set of constants  $c$  for which there exists an algorithm solving  $q$ -SAT in time  $O(2^{cn}) \cdot m^{O(1)}$ , where  $n$  is the number of variables in the  $q$ -SAT input and  $m$  the number of clauses.

### Definition (Strong Exponential-Time Hypothesis; SETH)

*Strong Exponential-Time Hypothesis* (unproven conjecture):

$$\lim_{q \rightarrow \infty} \delta_q = 1.$$

- The SETH is a stronger assumption than the ETH
- SETH implies that CNF-SAT cannot be solved in time  $O(2^{cn})$  for any  $c < 1$

- Considered exponential-time and subexponential-time algorithms
- Assumption about (impossibility of) subexponential-time algorithms: ETH
- How to use the ETH to derive exponential-time lower bounds

- Average-case complexity
- Impagliazzo's Five Worlds