2
(Strong) Exponential Time Hypothesis

Version: 174

In this chapter, we will introduce one of the main hardness assumptions used in this course, the **Strong Exponential Time Hypothesis**. To this end, we first discuss current algorithms for Satisfiability. We then postulate **increasingly strong assumptions** about its time complexity: $\mathsf{P} \neq \mathsf{NP}$, the Exponential Time Hypothesis (ETH), and the Strong Exponential Time Hypothesis (SETH). To illustrate them, we give conditional lower bounds for the **dominating set problem** based on these assumptions. Finally, we show that **SETH implies the OV hypothesis**, making it a stronger assumption.

Let us first introduce the satisfiability problem. Recall that a Boolean formula ϕ is in conjunctive normal form (CNF) if it is a conjunction of *clauses*. A clause is a disjunction of *literals*, and each literal is either a Boolean variable x_i or its negation $\overline{x_i}$.

$(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_4})$
Figure 2.1: Example of a CNF formula

Problem 2.1. *Let $k \in \mathbb{N}$.*

k-SAT
Given: *CNF formula ϕ with:*

- *N variables*
- *clause width k*
- *M clauses*

Determine: *Is there a **satisfying assignment** for ϕ?*

A **satisfying assignment** for ϕ is an assignment of true/false values to the variables such that all the clauses of ϕ are satisfied, i.e, it contains x_i where x_i is set to true or a negation $\overline{x_i}$ where x_i is set to false.
Note: There are at most $\binom{N}{k}2^k$ distinct clauses of width k. Thus, for fixed $k \in \mathbb{N}$, we typically assume without loss of generality that $M \leq \binom{N}{k}2^k = \mathcal{O}(N^k)$; i.e., the input for k-SAT is of polynomial size in N.

By Cook's Theorem, we know that k-SAT is NP-hard for $k \geq 3$. Thus, $\mathsf{P} \neq \mathsf{NP}$ implies that there is no polynomial-time algorithm for k-SAT for $k \geq 3$. (For $k = 2$, we can solve the problem in polynomial time.)

Let us introduce our running example for this chapter: the dominating set problem.

Problem 2.2.

Dominating Set (DomSet)
Given: *Undirected graph $G = (V, E)$ with n vertices, integer $q \in \mathbb{N}$.*
Determine: *Is there a **dominating set** S of size q?*

A **dominating set** S is a subset of V such that each $v \in V$ is **dominated by** S, i.e., we either have $v \in S$ or there exists an edge $\{u, v\} \in E$ such that $u \in S$.

A baseline algorithm solves Dominating Set in time $\mathcal{O}\left(\binom{n}{q} \cdot qn\right)$: For each size-$q$ subset of V, we check whether each $v \in V$ satisfies $v \in S$ or has an edge to some $u \in S$. The latter test can be done in time $\mathcal{O}(q)$ per vertex v, and there are $\binom{n}{q}$ size-q subsets of V, thus the running time bound follows. Unfortunately, as q can be as large as $\Theta(n)$, this gives no polynomial-time algorithm in the worst case.

2.1 *Lower Bounds under* P $\neq$ NP

Could there be a polynomial-time algorithm for Dominating Set? We give a negative answer, assuming P $\neq$ NP. This gives a simple example for a huge number of lower bounds that were proven under the P $\neq$ NP conjecture throughout decades of research since Karps's list of 21 NP-complete problems.

Theorem 2.3. *Dominating Set cannot be solved in polynomial-time unless* P $=$ NP.

Proof. We give a standard polynomial-time reduction from the NP-hard problem 3SAT to Dominating Set:

3-SAT

- N variables
- M clauses

$\xrightarrow{\text{time } \mathcal{O}(N+M)}$

Dominating Set

- $n = 3N + M$ nodes
- $q = N$

Since the above reduction runs in polynomial time $\mathcal{O}(N + M)$, a polynomial-time algorithm for dominating set would allow us to decide any 3SAT instance in polynomial-time, which would contradict P $\neq$ NP.

Let us give a reduction as specified above. Let ϕ be a 3-CNF formula. We construct a graph G as follows: For each variable x_i, we introduce *literal vertices* x_i, $\overline{x_i}$ and a *dummy vertex* d_i. We connect these three vertices to form a triangle. Additionally, for each clause C_j, we introduce a *clause vertex* representing C_j. Finally, we connect a literal vertex ℓ to a clause vertex C_j if and only if C_j contains the literal ℓ, i.e., $C = (\ell \vee \ell' \vee \ell'')$ for some literals ℓ', ℓ''. Observe that G has $3N + M$ vertices.

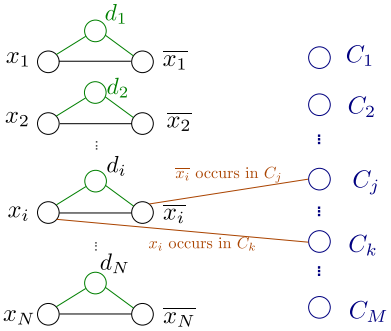

Figure 2.2: Illustration of constructed graph G.

Claim 2.4. *G has a dominating set of size N if and only if ϕ is satisfiable.*

Proof. If there is a satisfying assignment for ϕ, then let S be the set of literal vertices ℓ such that ℓ is true under this assignment. Note that S includes for any variable x_i precisely one of x_i and $\overline{x_i}$ and thus $|S| = N$. We show that S is a dominating set. For each variable x_i, the vertices $x_i, \overline{x_i}, d_i$ are dominated: one of x_i and $\overline{x_i}$ is in S, and the other two vertices are connected to this vertex. Furthermore, each clause vertex C_j is dominated: Our satisfying assignment sets some literal ℓ of C_j to true. Thus, the literal vertex ℓ, which is contained in S, has an edge to the clause vertex for C_j, which is thus dominated by S.

Conversely, let S be a dominating set in G of size N. To dominate all vertices $x_i, \overline{x_i}, d_i$ for $i \in [N]$, S must contain, for each $i \in [N]$, precisely one of $x_i, \overline{x_i}, d_i$. We define an assignment for ϕ, by setting each x_i to true if $x_i \in S$, to false if $\overline{x_i} \in S$ and to an arbitrary value, say true, if $d_i \in S$. We claim that this assignment satisfies ϕ: Each clause vertex C_j must be dominated by some literal vertex ℓ in S, which by definition requires that C_j contains ℓ. As our assignment is chosen such as to set every literal $\ell \in S$ to true, C_j must be satisfied. □

The above claim proves correctness of the reduction. Furthermore, observe that the graph G can be constructed in time $\mathcal{O}(N + M)$. Consequently, if Dominating Set is solvable in polynomial time, then 3-SAT is solvable in polynomial time.

□

2.2 *Lower Bounds under ETH*

Given that we do not expect polynomial-time algorithms for k-SAT and Dominating Set, what are the fastest (superpolynomial) algorithms for these problems?

Let us first consider this question for 3-SAT. It is hardly surprising that an extensive line of research has tried to obtain faster and faster algorithms for this problem:

Notation: $\mathcal{O}^*(\cdot)$ hides polynomial factors.

3-SAT Algorithms (partial list):

- $\mathcal{O}^*\left(2^N\right)$: trivial – check each of the 2^N assignments in $\text{poly}(N)$ time
- $\mathcal{O}^*\left(1.6181^N\right)$: Monien and Speckenmeyer[1]
- $\mathcal{O}^*\left(1.3334^N\right)$: Schöning[2], randomized
- $\mathcal{O}^*\left(1.3334^N\right)$: Moser, Scheder[3]
- ...
- $\mathcal{O}^*\left(1.3280^N\right)$: Liu[4]
- $\mathcal{O}^*\left(1.3071^N\right)$: PPSZ algorithm[5] with modifications by Hertli[6], randomized

A natural question is: When the research effort of the community

[1] Burkhard Monien and Ewald Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics*, 10 (3):287–295, 1985

[2] Uwe Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *FOCS'99*, pages 410–414, 1999

[3] Robin A. Moser and Dominik Scheder. A full derandomization of Schöning's k-SAT algorithm. In *STOC'11*, pages 245–252, 2011

[4] Sixue Liu. Chain, generalization of covering code, and deterministic algorithm for k-SAT. In *ICALP'18*, pages 88:1–88:13, 2018

[5] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. *J. ACM*, 52(3):337–364, 2005

[6] Timon Hertli. 3-SAT faster and simpler - Unique-SAT bounds for PPSZ hold in general. *SIAM J. Comput.*, 43(2):718–729, 2014

tends to infinity, what is the fastest algorithm that we will obtain? More formally, we define the following constant:

Definition 2.5. *For any $k \geq 3$, we define*

$$s_k := \inf\{\delta \mid k\text{-SAT has an } \mathcal{O}(2^{\delta N})\text{-time algorithm}\}.$$

Intuitively, we would hope that if we continue our collective search for faster algorithms, eventually, we will find $\mathcal{O}(2^{s_k \cdot n + \varepsilon})$-time k-SAT algorithms for some very small $\varepsilon > 0$.

Currently, it is far from clear what the true value of, e.g., s_3 should be. If $\mathsf{P} = \mathsf{NP}$, or if 3-SAT has a subexponential algorithm (e.g., running in time $\mathcal{O}(2^{\sqrt{N}})$), then $s_3 = 0$. The fastest algorithm in the list above gives the upper bound $s_3 < 0.3863$.

Given the importance of 3-SAT for theoretical computer science, and the extensive effort of algorithmic research on this problem, it is plausible to conjecture that in fact $s_3 > 0$ – this is in essence the **conjecture that 3-SAT has no subexponential-time algorithms**. This conjecture is called the Exponential Time Hypothesis postulated by Impagliazzo and Paturi[7].

[7] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2): 367–375, 2001

Hypothesis 2.6 (Exponential Time Hypothesis (ETH))**.** *The Exponential Time Hypothesis postulates that*

$$s_3 > 0.$$

An equivalent formulation is the following:

There is some $\delta > 0$ such that 3-SAT cannot be solved in time $\mathcal{O}(2^{\delta N})$.

A note on randomness: In the definition of s_k (and thus in the hypothesis above), we allow for both deterministic and randomized algorithms. This is sometimes called *randomized* ETH – however, we will not distinguish between deterministic and randomized ETH. In fact, throughout this course, we will conjecture lower bounds both for deterministic *and* randomized algorithms (unless specifically stated otherwise).

Note that this assumption is stronger than $\mathsf{P} \neq \mathsf{NP}$. As such, we can actually get stronger lower bounds based on this conjecture (than based on $\mathsf{P} \neq \mathsf{NP}$). In fact, the reduction above immediately gives the following lower bound for dominating set:

Theorem 2.7 (Simple ETH lower bound)**.** *There is some $\delta > 0$ such that Dominating Set cannot be solved in time $\mathcal{O}(2^{\delta n^{1/3}})$ unless ETH fails.*

Proof. Assume that the converse is true, i.e., that for all $\delta > 0$, Dominating Set has an $\mathcal{O}(2^{\delta n^{1/3}})$-time algorithm. Given an arbitrary 3SAT instance ϕ, we use the reduction of the proof of Theorem 2.3 to construct, in polynomial time, a graph G on $n = 3N + M$ nodes such that ϕ is satisfiable if and only if G has a dominating set of size N. Note that we may assume without loss of generality that ϕ has $M \leq \binom{N}{3}2^3 \leq 8N^3$ clauses. Thus, $n = 3N + M \leq 11N^3$. Consequently, constructing G takes time $\text{poly}(N)$ and the above Dominating Set algorithm decides whether G has a dominating set of size N in time $\mathcal{O}(2^{\delta n^{1/3}}) = \mathcal{O}(2^{\delta' N})$ with $\delta' = \sqrt[3]{11} \cdot \delta$. As this holds for all δ, we obtain a $\mathcal{O}(2^{\delta' N})$-time 3SAT algorithm for all $\delta' > 0$, which would refute the Exponential Time Hypothesis. □

Why do we only obtain a lower bound of $2^{\Omega(n^{1/3})}$ instead of $2^{\Omega(n)}$? The crucial point is that the number of nodes in the Dominating Set instance may get as large as $\Theta(M) = \Theta(N^3)$ instead of $\Theta(N)$. However, if we could, for some reason, assume that already 3SAT on a **linear number of clauses** $M = \mathcal{O}(N)$ has no subexponential time algorithms, then we would get a stronger lower bound. Surprisingly, this is possible because of the following result, the **Sparsification Lemma** due to Impagliazzo, Paturi and Zane[8]. Let us first state the basic intuition of this result:

[8] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001

Intuition: Sparsification Lemma
*The **hard case** for k-SAT consists of **sparse formulas**.*
That is, in reductions from k-SAT, we may assume that $M = \mathcal{O}(N)$.

More formally, the lemma describes an algorithm that reduces satisfiability of an N-variable k-CNF formula to a *subexponential* number of satisfiability instances consisting of *sparse* N-variable k-CNF formulas (i.e., formulas with $M = \mathcal{O}(N)$ clauses).

Theorem 2.8 (Sparsification Lemma)**.** *Let $k \in \mathbb{N}$ and $\varepsilon > 0$. There is a constant $C = C(k, \varepsilon)$ and an algorithm such that*

1. *given a k-CNF formula ϕ, the algorithm computes formulas $\phi_1, \ldots, \phi_t$,*
2. *ϕ is satisfiable if and only if there is some $i \in [t]$ such that ϕ_i is satisfiable,*
3. *we have $t \leq 2^{\varepsilon N}$ and the algorithm runs in time $\mathcal{O}(2^{\varepsilon N}\text{poly}(N))$, and*
4. *each ϕ_i is a k-CNF formula with N variables and $M_i \leq C \cdot N$ clauses.*

The proof of this result is beyond the scope of this course.

Let us use the Sparsification Lemma to obtain a stronger lower bound for Dominating Set under ETH.

Theorem 2.9. *There is some $\delta > 0$ such that Dominating Set cannot be solved in time $\mathcal{O}(2^{\delta n})$.*

Proof. Let $\delta > 0$, $\varepsilon := \delta/2$ and set

$$\delta' := \frac{\delta}{2(3 + C(3,\varepsilon))}.$$

Assume there is an algorithm $\mathcal{A}$ solving dominating set in time $\mathcal{O}(2^{\delta' n})$. Given a 3-CNF formula ϕ, we compute $\phi_1, \ldots, \phi_t$ with $t \leq 2^{\varepsilon N}$ as in the Sparsification Lemma. For each ϕ_i, we construct G_i using the reduction in the proof of Theorem 2.3. Note that G_i has $n_i = 3N + M_i$ many nodes where $M_i \leq C(3,\varepsilon) \cdot N$. Thus, running $\mathcal{A}$ on G_i takes time

$$\mathcal{O}\left(2^{\delta'(3N+M_i)}\right) = \mathcal{O}\left(2^{\delta'(3+C(3,\varepsilon))N}\right) = \mathcal{O}\left(2^{\delta N/2}\right).$$

Thus, the total running time to decide satisfiability of all ϕ_i, and thus the satisfiability of ϕ, is bounded by

$$t \cdot \mathcal{O}\left(2^{\delta N/2}\right) = \mathcal{O}\left(2^{(\varepsilon+\delta/2)N}\right) = \mathcal{O}\left(2^{\delta N}\right).$$

Thus, if there is an $\mathcal{O}(2^{\delta' N})$-time dominating set algorithm for all $\delta' > 0$, we obtain an $\mathcal{O}(2^{\delta N})$-time algorithm for 3-SAT for all $\delta > 0$, contradicting ETH. □

In summary, with the above conditional lower bound, we obtain the following state of the art:

1. An $\mathcal{O}(1.4969^n)$-time algorithm solves dominating set.[9]
2. Dominating Set requires time $\Omega(c^n)$ for *some* constant $c > 0$, assuming ETH.

[9] Johan M. M. van Rooij and Hans L. Bodlaender. Exact algorithms for dominating set. *Discrete Applied Mathematics*, 159(17):2147–2164, 2011

Let us turn to the question what ETH might say about the **polynomial-time regime**. In particular consider the problem of finding *small* (i.e., constant-sized) dominating sets.

Problem 2.10. *Let $q \in \mathbb{N}$.*

q-Dominating Set	
Given:	*Undirected graph $G = (V, E)$*
Determine:	*Does G have a dominating set of size q?*

Note that for any fixed q, the problem q-Dominating Set is solvable in polynomial time: The $\mathcal{O}\left(\binom{n}{q} \cdot qn\right)$-time algorithm runs in time $\mathcal{O}(n^{q+1})$ for any constant q. Interestingly, for every $q \geq 7$, there is even an $n^{q+o(1)}$-time algorithm for q-Dominating Set[10,11], shaving off an almost linear factor.

[10] Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.*, 326(1-3):57–67, 2004

[11] Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *Proc. of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*, pages 1065–1075, 2010

Let us investigate whether a running time of $n^{q+o(1)}$ should or should not be essentially best possible. Could there be a $\mathcal{O}(n^{\sqrt{q}})$-time algorithm? Maybe such algorithms are not possible, but a $\mathcal{O}(n^{q-2})$-time algorithm exists? We will see that ETH implies that no $\mathcal{O}(n^{\sqrt{q}})$-time

algorithms exists *for some sufficiently large* q. For the second question, however, we will not obtain an answer based on the ETH; instead this will be a task for a stronger variant of ETH.

Theorem 2.11. *Assuming the ETH, there is some* $\delta > 0$ *such that* q*-Dominating Set has no* $\mathcal{O}(n^{\delta q})$*-time algorithms for all sufficiently large* q*.*

Proof. We will generalize our reduction from Theorem 2.3 to the following reduction:

k-SAT

- N variables
- M clauses

$\xrightarrow{\text{time } \mathcal{O}(2^{2N/q})}$

q-Dominating Set

- $n = q2^{N/q} + q + M$ nodes

Note: This reduction has an *exponential blow-up*: the number of nodes constructed is $\Theta(2^{N/q})$. This is indeed necessary to connect the exponential time regime of ETH to the polynomial-time regime of our desired lower bound.

[12] Note that we have given a general reduction from k-SAT, although we only need a reduction from 3-SAT at this point – we will make use of the more general reduction in the next section.

Given this reduction, it suffices to show that for any $\delta > 0$, an $\mathcal{O}(n^{\delta q})$-time algorithm for q-Dominating set for some $q \geq 2/\delta$ would give a $\mathcal{O}(2^{\delta N})$-time algorithm for k-SAT, and thus in particular, 3-SAT[12]. To see this, note that the resulting instance consists of $n = q2^{N/q} + q + M = \mathcal{O}(2^{N/q})$ nodes – here, we use that $M = \mathcal{O}(N^K) = \mathcal{O}(2^{N/q})$, as q is a fixed constant. Thus, running the $\mathcal{O}(n^{\delta q})$-time algorithm on this instance takes time

$$\mathcal{O}\left(\left(2^{N/q}\right)^{\delta q}\right) = \mathcal{O}\left(2^{\delta N}\right).$$

The reduction time is bounded by $\mathcal{O}(2^{2N/q}) = \mathcal{O}(2^{\delta N})$ using $q \geq 2/\delta$. Thus, we obtain an $\mathcal{O}(2^{\delta N})$-time algorithm for 3-SAT, and the claim follows.

It remains to design the above reduction. Consider the following basic idea: In the reduction for Theorem 2.3, each vertex of the dominating set "chose" an assignment of true or false for some variable. In our setting, the dominating set has a much smaller size of q – thus, each vertex in the dominating set needs to choose an assignment for a larger number of variables, specifically, for N/q variables.

Formally, let ϕ be a given k-CNF formula and construct a graph G as follows. We partition the variables into q groups $G^{(1)}, \ldots, G^{(q)}$, where $G^{(i)} = \{x_{(i-1)N/q+1}, \ldots, x_{iN/q}\}$. For each $i \in [q]$ and every possible assignment of true/false values to the variables in group $G^{(i)}$, we define a corresponding *partial assignment vertex* $a_p^{(i)}$, where $p \in [2^{N/q}]$ (as there are precisely N/q variables in each group). Additionally, for each group $G^{(i)}$, we define a *dummy vertex* $d^{(i)}$. For each $i \in [q]$, we connect all nodes $a_1^{(i)}, \ldots, a_{2^{N/q}}^{(i)}, d^{(i)}$ to each other; i.e., the vertices corresponding to a group $G^{(i)}$ form a *clique*.

Furthermore, we introduce a *clause vertex* for each clause C_j. We connect a partial assignment vertex $a_p^{(i)}$ to a clause vertex C_j if and

only if C_j contains a literal $\ell \in \{x_k, \overline{x_k}\}$ for some variable $x_k \in G^{(i)}$ *and the assignment represented by* $a_p^{(i)}$ *sets this literal to true.*

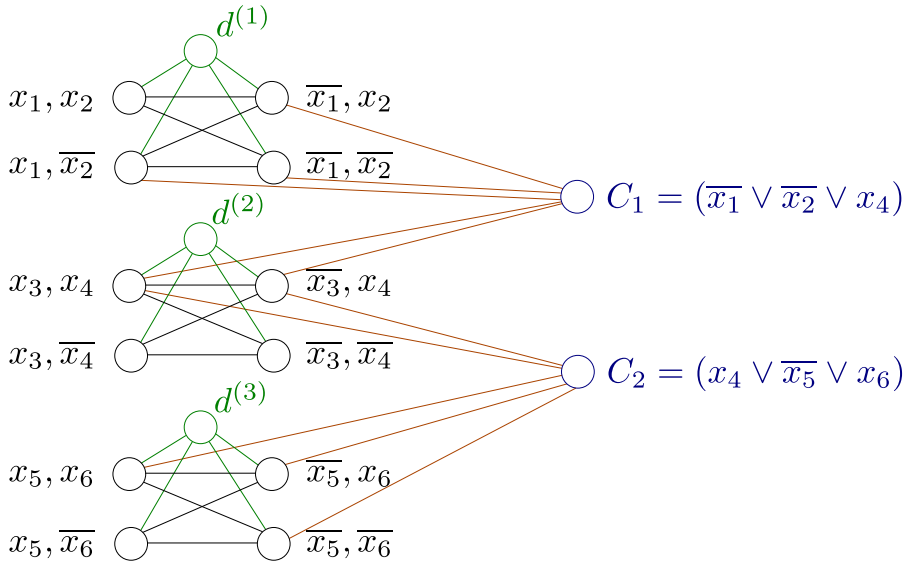

Figure 2.3: Example for the construction for a 6-variable formula $(\overline{x_1} \vee \overline{x_2} \vee x_4) \wedge (x_4 \vee \overline{x_5} \vee x_6)$ and $q = 3$. We label each partial assignment vertex by the set of literals that are true under this assignment.

The following claim is analogous to the corresponding claim in the proof of Theorem 2.3:

Claim 2.12. *ϕ is satisfiable if and only if G has a dominating set of size q.*

This claim establishes correctness of the reduction. Finally, observe that we can construct this graph in linear time in the number of its edges. Since this number is bounded by $n^2 = \mathcal{O}(2^{2N/q})$, the claimed reduction follows. □

The above lower bound shows, e.g., that there cannot be an algorithm that solves q-Dominating Set for arbitrary q in time $\mathcal{O}(n^{\sqrt{q}})$, unless the ETH fails. However, if we fix some q, the above result gives *no specific lower bound*, since any lower bound only applies for sufficiently large q. In this sense, this statement might best be described as a statement about the *family* of problems q-Dominating Set for $q \in \mathbb{N}$. In the next section, we will introduce a stronger variant of ETH that allows us to derive conditional lower bounds for specific polynomial-time problems.

2.3 Lower Bounds under SETH

So far, we have discussed specifically the fastest algorithms for 3-SAT. What are the fastest algorithms for k-SAT with $k > 3$? Currently, we have the following upper bounds[13,14]:

- 3-SAT: $\mathcal{O}(1.3071^N)$,

[13] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. *J. ACM*, 52(3):337–364, 2005

[14] Timon Hertli. 3-SAT faster and simpler - Unique-SAT bounds for PPSZ hold in general. *SIAM J. Comput.*, 43(2):718–729, 2014

- 4-SAT: $\mathcal{O}(1.4690^N)$,
- ...
- k-SAT: $\mathcal{O}(2^{(1-c/k)N})$ for some constant $c > 0$.

Note that for the current state of the art, the larger k, the more time it takes to solve the problem. In particular, if we let k tend to infinity, the running time of the fastest known k-SAT algorithm approaches $\mathcal{O}(2^N)$. The **Strong Exponential Time Hypothesis** postulated by Impagliazzo and Paturi[15] states that this is true for the best possible k-SAT algorithms. Intuitively, one may think of it as follows.

[15] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2): 367–375, 2001

Intuition: Strong Exponetial Time Hypothesis
For sufficently large k, k-SAT has no $\mathcal{O}(1.999^N)$-time algorithms.

Here, we could replace 1.999 by an arbitrary value $c < 2$. Formally, the hypothesis is stated as follows.

Hypothesis 2.13 (Strong Exponential Time Hypothesis (SETH))**.** *The Strong Exponential Time Hypothesis postulates that*

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

An equivalent formulation is the following:

For all $\varepsilon > 0$, there is some $k \geq 3$ such that k-SAT cannot be solved in time $\mathcal{O}(2^{(1-\varepsilon)N})$.

We often state SETH slightly weaker as follows: For no $\varepsilon > 0$, there is an algorithm that solves any k-SAT problem in time $\mathcal{O}(2^{(1-\varepsilon)N})$.

What evidence is there for SETH? Impagliazzo and Paturi proved that if ETH is true, then s_k increases infinitely often. The hypothesized value $\lim_{k\to\infty} s_k = 1$ would be consistent with the current state of the art of k-SAT algorithms. Cygan et al.[16] proved SETH is in fact equivalent to analogous conjectures for other problems such as Hitting Set, Set Splitting and k-NotAllEqual SAT.

[16] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3): 41:1–41:24, 2016
Note: This list of evidence for SETH is not exhaustive!

What consequences does SETH imply for polynomial-time algorithms? Using the reduction of Theorem 2.11, we obtain the following result.

Theorem 2.14. *Let $q \geq 3$ and $\varepsilon > 0$. There is no q-Dominating Set algorithm running in time $\mathcal{O}(n^{q-\varepsilon})$ unless SETH fails.*

Proof. Let $0 < \varepsilon \leq 1$ and assume that there exists an $\mathcal{O}(n^{q-\varepsilon})$-time q-Dominating Set algorithm.

Given a k-SAT instance ϕ, we use the reduction of the proof of Theorem 2.11 to produce a q-Dominating set instance on $n = \mathcal{O}(2^{N/q})$

nodes in time $\mathcal{O}(2^{2N/q})$. Running the $\mathcal{O}(n^{q-\varepsilon})$-time q-Dominating Set algorithm on this instance takes time $\mathcal{O}((2^{N/q})^{q-\varepsilon}) = \mathcal{O}(2^{(1-\varepsilon/q)N})$. Thus, in total, we can decide satisfiability of ϕ in time

$$\mathcal{O}(2^{2N/q}) + \mathcal{O}(2^{(1-\varepsilon/q)N}) = \mathcal{O}(2^{(1-\varepsilon')N}),$$

where we set $\varepsilon' := \varepsilon/q \leq 1/3$.

Since k can be chosen arbitrarily, this would show that k-SAT has a $\mathcal{O}(2^{(1-\varepsilon')N})$-time algorithm for all k, proving $\lim_{k\to\infty} s_k \leq 1-\varepsilon'$, which would contradict SETH. □

The above lower bound yields a very strong conditional lower bound for polynomial-time problems. In particular, already a $\mathcal{O}(n^{2.9999})$-time algorithm for 3-Dominating Set would refute SETH!

Finally, we turn to the connection between SETH and OVH. The following conjecture establishes that OVH is even more plausible than SETH. This greatly helped to popularize OVH as a fundamental hardness assumption for polynomial-time problem. The following reduction is due to Williams[17].

[17] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005

Theorem 2.15 (SETH implies OVH)**.** *If SETH holds, then for no $\varepsilon > 0$ there is an $\mathcal{O}(n^{2-\varepsilon}\text{poly}(d))$-time OV algorithm.*

Proof. We show the following reduction.

k-SAT

- N variables
- M clauses

$\xrightarrow{\text{time } \mathcal{O}(2^{N/2}M)}$

OV

- $n = 2^{N/2}$ vectors
- $d = M$ dimensions

Assume that there is a $\mathcal{O}(n^{2-\varepsilon}\text{poly}(d))$-time OV algorithm for some $\varepsilon < 1/2$. Using this algorithm on the output of the above reduction, we obtain a k-SAT algorithm running in time

$$\mathcal{O}(2^{N/2}M + 2^{N/2(2-\varepsilon)}\text{poly}(M)) = \mathcal{O}(2^{N(1-\varepsilon/2)}\text{poly}(M)) = \mathcal{O}(2^{(1-\varepsilon')N}),$$

for an arbitrary $0 < \varepsilon' < \varepsilon/2$. Here, we used that $M \leq N^k$ is polynomially small.

Thus, as k was chosen arbitrarily, we obtain a contradiction to SETH, as this would give some $\varepsilon' > 0$ such that k-SAT is solvable in time $\mathcal{O}(2^{(1-\varepsilon')}N)$ for all $k \geq 3$.

It remains to design the above reduction, which is very similar to the reduction from Theorem 2.11: We partition the variables into two halves, i.e., $x_1, \ldots, x_{N/2}$ and $x_{N/2+1,\ldots,N}$ (here we assume for convenience that N is even). Let U denote the set of assignments of Boolean values to $x_1, \ldots, x_{N/2}$, and similarly, let V contain all assignments to the variables $x_{N/2+1}, \ldots, x_N$. These assignments are called *partial assignments*. Note that $|U| = |V| = 2^{N/2}$.

For a partial assignment $u \in U$ and a clause C, we define $\text{sat}(u, C) \coloneqq 1$ if and only if u satisfies the clause C, i.e., C contains a literal x_i such that x_i is set to true in u or C contains a literal $\overline{x_i}$ such that x_i is set to false in u. Otherwise, we set $\text{sat}(u, C) \coloneqq 0$. We define $\text{sat}(v, C)$ analogously for $v \in V$.

Given partial assignments $u \in U, v \in V$, we define the M-dimensional vector

$$\text{vec}(u) \coloneqq (1 - \text{sat}(u, C_1), \ldots, 1 - \text{sat}(u, C_M))$$
$$\text{vec}(v) \coloneqq (1 - \text{sat}(v, C_1), \ldots, 1 - \text{sat}(v, C_M))$$

It is straightforward to verify following claim.

Claim 2.16. *For any $u \in U, v \in V$, we have that $\text{vec}(u)$ and $\text{vec}(v)$ are orthogonal if and only if (u, v) gives a satisfying assignment for ϕ.*

Proof. Note that $\text{vec}(u), \text{vec}(b)$ have an inner product of 0 if and only if for all $k = 1, \ldots, M$, we have $\text{sat}(u, C_k) = 1$, $\text{sat}(v, C_k) = 1$, or both. By definition, this occurs if and only if each clause C_k is satisfied by at least one literal given by the assignment (u, v). □

Thus, we can conclude the reduction by defining the vector sets $A \coloneqq \{\text{vec}(u) \mid u \in U\}$ and $B \coloneqq \{\text{vec}(v) \mid v \in V\}$, which are sets of $n = 2^{N/2}$ vectors in $\{0,1\}^d$ with $d = M$, as desired. □

The above reduction easily generalizes to an analogous reduction for the so called k-OV problem:

Problem 2.17.

k-Orthogonal Vectors (k-OV)

Given:	*vectors sets $A_1, \ldots, A_k \subseteq \{0,1\}^d$,* $\lvert A_1\rvert = \lvert A_2\rvert = \cdots = \lvert A_k\rvert = n$
Determine:	*Is there a tuple $(a_1, \ldots, a_k)$ such that for all $j \in [d]$ we have $a_1[j] \cdot a_2[j] \cdots a_k[j] = 0$?*

Analogously to OV, we can solve k-OV in time $\mathcal{O}(n^k d)$, but for no k, we know of a $\mathcal{O}(n^{k-\varepsilon}\text{poly}(d))$-time algorithm. Indeed, using a similar reduction as above, we can show that such algorithms do not exist under SETH. (→ **exc.**)

In conclusion, we have discussed the conjectures $P \neq \mathsf{NP}$, ETH and SETH. We have seen that while ETH rules out $n^{o(q)}$-algorithms for q-Dominating Set, we need the stronger assumption SETH to prove tight lower bounds for specific polynomial-time problems such as 3-Dominating Set or OV (for these, we have proven matching $n^{3-o(1)}$ and $n^{2-o(1)}$-time lower bounds, respectively).

$$\begin{array}{l} \mathsf{P} \neq \mathsf{NP} \\ \Uparrow \\ \text{ETH} \\ \Uparrow \\ \text{SETH} \end{array} \quad \Longrightarrow \text{OVH}$$

Exercises

2.1) Show that Subset Sum has no $2^{o(n)}$ time algorithm under ETH.

2.2) Show that SETH implies ETH.

2.3) Show that SETH implies that for no $k \geq 2$ and $\varepsilon > 0$, there is a $\mathcal{O}(n^{k-\varepsilon}\text{poly}(d))$-time algorithm for k-OV.

Bibliography

Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016.

Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.*, 326 (1-3):57–67, 2004.

Timon Hertli. 3-SAT faster and simpler - Unique-SAT bounds for PPSZ hold in general. *SIAM J. Comput.*, 43(2):718–729, 2014.

Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

Sixue Liu. Chain, generalization of covering code, and deterministic algorithm for k-SAT. In *ICALP'18*, pages 88:1–88:13, 2018.

Burkhard Monien and Ewald Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics*, 10(3):287–295, 1985.

Robin A. Moser and Dominik Scheder. A full derandomization of Schöning's k-SAT algorithm. In *STOC'11*, pages 245–252, 2011.

Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. In *Proc. of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*, pages 1065–1075, 2010.

Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-SAT. *J. ACM*, 52(3): 337–364, 2005.

Uwe Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *FOCS'99*, pages 410–414, 1999.

Johan M. M. van Rooij and Hans L. Bodlaender. Exact algorithms for dominating set. *Discrete Applied Mathematics*, 159(17):2147–2164, 2011.

Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005.