Hardness and The Exponential Time Hypothesis: Classical, Fine Grained and Parameterized Complexity

Alexander Temper

March 26, 2025

In general, it is hard to prove that algorithms of a certain time complexity do not exist. However, under some reasonable assumptions, it is possible to prove *conditional lower bounds*, that is, lower bounds of the form

"If hypothesis H is true, then problem P can not be solved in time $T(n)$."

The *Exponential Time Hypothesis (ETH)* is such a reasonable assumption, namely that 3-SAT can not be solved in subexponential time. In this seminar paper, I introduce the ETH, argue for why it is a reasonable assumption and prove that, unless the ETH fails, certain problems can not be solved in subexponential time. We consider the ETH's implications from three perspectives, that of classical, fine grained and parameterized complexity.

1 Classical Complexity

Classical complexity theory tries to assign problems to certain complexity classes, which, under some assumptions, tells us how much time and space are needed to solve them asymptotically. To begin with, let us introduce the formal notion of a problem.

Definition 1 (Problem)**.** Formally, a *(decision) problem* $A \subseteq \{0,1\}^*$ is a subset of the set of strings consisting only of ones and zeros. An instance $x \in \{0,1\}^*$ is called a YES-instance if $x \in A$, otherwise it is called a NO-instance. Less formally, a decision problem can be posed as a question about a set of instances/inputs, and a YES-instance is correspondingly an input for which the question can be answered with "yes", and conversely a NO-instance is an input for which the answer to the question is "no".

The classes P and NP are generally agreed on to represent tractability and intractability respectively and are the most prominent complexity classes to date.

Definition 2 (P and NP)**.** A problem $A \subseteq \{0,1\}^*$ is contained in P (respectively in NP) if there is a polynomial-time deterministic (respectively non-deterministic) Turing machine [1] (TM) which accepts an input $x \in \{0,1\}^*$ if and only if $x \in A$.

However, this is not yet a useful definition to separate the tractable from the intractable, because it is difficult to show that *no* deterministic polynomial time TM for a given problem exists. In fact, such a statement has not been shown for any *NP-complete* problem yet; doing so would famously be awarded with a million dollars. However, certain problems appear to be really difficult, to an extent that it seems unimaginable that they admit a deterministic polynomial time TM. Using the notion of a *reduction*, this hardness can be transferred to other problems.

Definition 3 (Many-one reduction)**.** Let $A, B \subseteq \{0,1\}^*$ be two problems. A *many-one* reduction R is an algorithm that transforms an instance x of A into an instance x' of B runs in $\mathrm{poly}(|x|)$ time, such that

[1]For an extensive introduction to Turing machines, see Arora and Barak [1, Chapter 1]

- $|x'| \leq \text{poly}(|x|)$ and
- $x' \in B \iff x \in A$.

Thus, if one can transform a any instance of problem A into a similarly sized instance of problem B in polynomial time, then A must be similarly difficult to solve as B. If B is solvable in polynomial time, then A is so too. The contrapositive, inspires the concept of NP-*hardness*: a seemingly hard A implies that B should also be difficult.

Definition 4 (NP-hardness)**.** A problem A is called *NP-hard* if every problem in NP reduces to it.

To offer a non-canonical definition of NP that is based firmly on the concepts of problems and reductions, we can think of NP as all problems that reduce to the following problem.

Turing Machine Acceptance
Input: A non-deterministic polynomial time Turing machine M, an input-string $x \in \{0,1\}^*$
Question: Does M have a computation path accepting x?

Observation 1. *The following equivalences hold:*

- *Problem A is contained in NP if and only if it reduces to* Turing Machine Acceptance.
- *A is NP-hard if and only if* Turing Machine Acceptance *reduces to A.*

I feel like this definition is useful to have in mind if one likes to think of NP as having an "original" problem, and it will allows one to see an illustrative analogue in a later chapter on *parameterized* complexity.

Irrespective of how one defines NP-hardness, it is not yet useful as a stand-alone concept. This is because its defining problem is difficult to handle. First, an easy to work with NP-*complete* problem, i.e., an NP-hard problem that is also contained in NP, needed to be found. It turned out to be SAT, which represents the heart of NP; a restriction of SAT, called 3-SAT, is central to the topic of this paper. I define both simulataneously:

(3-)SAT
Input: A (3-)CNF formula Φ over variables $X := \{x_1, \dots, x_n\}$.
Question: Does there exist a satisfying assignment $I : X \to \{0,1\}$?

Here, recall that a *3-CNF* formula is a conjunction of clauses of length 3, and a satisfying assignment is an assignment such that $I(\Phi) = 1$. An illustrative example should suffice to refresh the readers memory.

Example 1. The formula

$$\Phi := \underbrace{(a \vee b \vee c)}_{\text{3 literals per clause}} \wedge (a \vee \neg b \vee \neg c)$$

is a 3-CNF formula, thus an instance of 3-SAT. It is a YES-instance, as it has a satisfying assignment, namely $I(a) = 1, I(b) = 0, I(c) = 1$, because

$$\begin{aligned}
I(\Phi) &= \min(I(a \vee b \vee c), I(a \vee \neg b \vee \neg c)) \\
&= \min(\max(I(a), I(b), I(c))), \max(I(a), I(\neg b), I(\neg c)) \\
&= \min(\max(I(a), I(b), I(c))), \max(I(a), (1 - I(b)), (1 - I(\neg c))) \\
&= \min(\max(1, 0, 1), \max(1, (1 - 0), (1 - 1))) \\
&= \min(1, 1) \\
&= 1.
\end{aligned}$$

The paramount result obtained by Cook, Levin and Karp independently, known as the Cook-Levin Theorem, established 3-SAT as the first practical NP-complete problem.

Theorem 1 (Cook-Levin Theorem [15, 8])**.** TURING MACHINE ACCEPTANCE *reduces to* SAT.

In addition, SAT reduces to 3-SAT, which lends itself well to reductions, and showing its completeness was the necessary (and tedious) step needed to make the concept of NP-hardness practically relevant. Nevertheless, all claims deeming problems in NP intractable rely on the following hypothesis.

Hypothesis 1. *$P \neq NP$, that is,* 3-SAT *has no algorithms running in polynomial time.*

Even though this hypothesis is largely agreed upon and thus 3-SAT is believed to not admit any polynomial time algorithms, its importance and relative simplicity have been the motivation for decades of research on algorithms for solving 3-SAT. Nevertheless, there has not been any asymptotic improvement to the *exponent* of the brute force algorithm which runs in time $O(2^n)$. Below, we give an incomplete list of contributions and the worst-case time complexity of their proposed algorithm.

Year	c	**Source**
	1	Brute force
1985	0.694	Monien and Speckenmeyer [16]
1999	0.415	Schoning [18]
2010	0.404	Iwama and Tamaki [14]
2011	0.400	Moser and Scheder [17]
2018	0.386	Hertli [11]

Table 1: An incomplete list of upper-bounds $O(2^{cn})$ on the worst-case time complexity for solving 3-SAT

Over the decades, improvements have even declined. It seems as if there is a bound to how fast 3-SAT can be solved. This observation is the motivation for the following hypothesis.

Hypothesis 2 (Exponential Time Hypothesis, [12])**.** *There exists a $\delta^* > 0$ such that* 3-SAT *can not be solved in time $O(2^{\delta^* n})$.*

This hyphothesis is frequently stated in two different ways — the one stated above is very useful for proofs, the other one, a little less technical and more intuitive, states that 3-SAT can not be solved in *subexponential time.* Next, we will see that the two definitions are equivalent, for which we recall the meaning of some notation and give meaning to the word "subexponential".

Definition 5 (Little-o and Big-O notation)**.** Let $f : \mathbb{N} \to N, g : \mathbb{N} \to N$ be two functions. One says that $f(n) = o(g(n))$ iff

$$\forall k > 0 \exists n_0 \forall n > n_0 : |f(n)| < kg(n)$$

and $f(n) = O(g(n))$ iff

$$\exists k > 0 \exists n_0 \forall n > n_0 : |f(n)| \leq kg(n).$$

Definition 6 (Subexponential function)**.** A function f is *subexponential* if $f(n) = 2^{o(n)}$.

Proposition 1.

Hypothesis 2 $\iff$ 3-SAT *can not be solved in subexponential time.*

Proof. " $\implies$ ": Assume there exists such a $\delta^* > 0$ such that 3-SAT can not be solved in time $O(2^{\delta^* n})$. That is that no algorithm with running time $f(n)$ can exist for which

$$\exists k > 0 \exists n_0 \forall n > n_0 : f(n) \leq k2^{\delta^* n} \tag{1}$$

Suppose by contradiction that 3-SAT can be solved in subexponential time, i.e., there is an algorithm $\mathcal{A}$ with running time h such that $h(n) = 2^{o(n)}$. This would imply that that

$$\forall k > 0 \exists n_0 \forall n > n_0 : \, h(n) < 2^{kn}$$

so also that

$$\exists n_0 \forall n > n_0 : \, h(n) < 2^{\delta^* n},$$

but then, taking $f := h$ and $k := 1$ contradicts assumption 1.

" $\Longleftarrow$ ": This direction works nearly identically to the other, so I omit the technical details. □

Let us remind ourselves that we are here to see useful lower bounds. Unfortunately, there is a trade-off between generality and specificity when it comes to how one can define reductions. Many-one reductions are relatively loose and thus general. This allows for a well-connected network of problems which all reduce to each other — at this point, this network spans probably around more than a thousand problems. However, this comes at a cost: no noteworthy general conditional lower bounds are obtainable. As we will see soon, to make use of the ETH, reductions from 3-SAT will be necessary, and the size of the instances obtained from the reductions will be critical. However, with many-one reductions, those sizes need to be bounded only polynomially by the size of the original instance, which means that no useful general conditional lower bounds can be given for NP-complete problems that are not already implied by P $\neq$ NP. This leads us to a branch of computational complexity theory called *fine grained complexity*.

2 Fine Grained Complexity

The ETH is one of the protagonists of a field called *fine grained complexity theory*, which is a subfield of computational complexity theory that aims to give sharper lower bounds than the ones classical complexity theory can offer. It does so by studying hardness under the paradigm of so called *fine-grained reductions*, which are reductions that are specific in the size of their resulting instances and their running times.

The goal of this chapter is to illustrate the how a fine-grained reduction can be coupled with a reasonable hypothesis to give such a conditional lower bound. We will show that, unless the ETH fails, Dominating Set does not admit any subexponential algorithms. To do so, we will give fine-grained reduction from 3-SAT to Dominating Set. We will then establish that a subexponential algorithm for Dominating Set would allow us to solve 3-SAT in subexponential time.

In the beginning, we will fail to do so, as we lack yet an important tool, called the *sparsification lemma*. Afterwards, we will indeed show that a subexponential algorithm for Dominating Set would allow us to solve 3-SAT in subexponential time, which would contradict the ETH. Thereafter, we will see how that result generalizes to a whole class of problems.

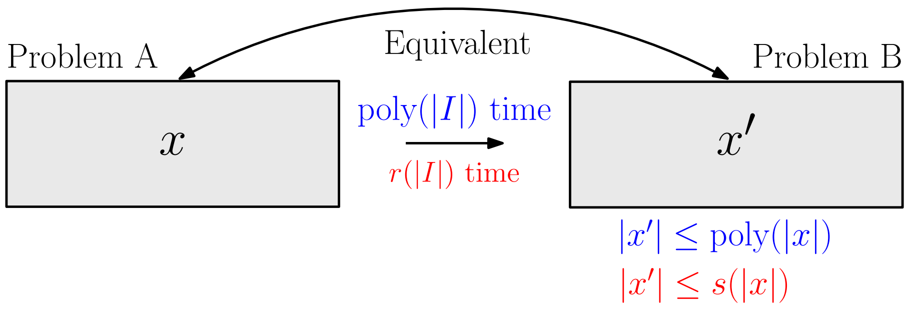

Figure 1: Illustration of many-one reductions (blue) vs. fine-grained (s,t)-reductions (red). The essence of fine-grained complexity is to be specific about reductions and their resulting instance sizes.

To start off our journey, let us introduce the problem formally.

Dominating Set

Input: A graph $G = (V, E)$, a number k

Question: Does there exist a subset $D \subset V$ of size k such that

$$\forall v \in V : v \in D \vee \exists u \in V : \{u, v\} \in E \wedge u \in D?$$

Example 2.

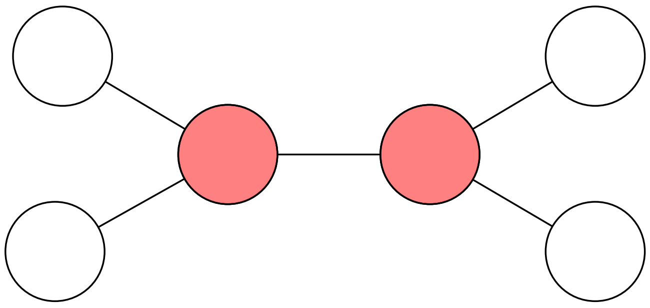

Figure 2: A graph with a dominating set of size 2 in red

Theorem 2. *There is a reduction R which turns an instance Φ 3-SAT with n variables and m clauses into an instance (G, n) of* Dominating Set *that runs in $O(n + m)$ time such that G has $3n + m$ vertices.*

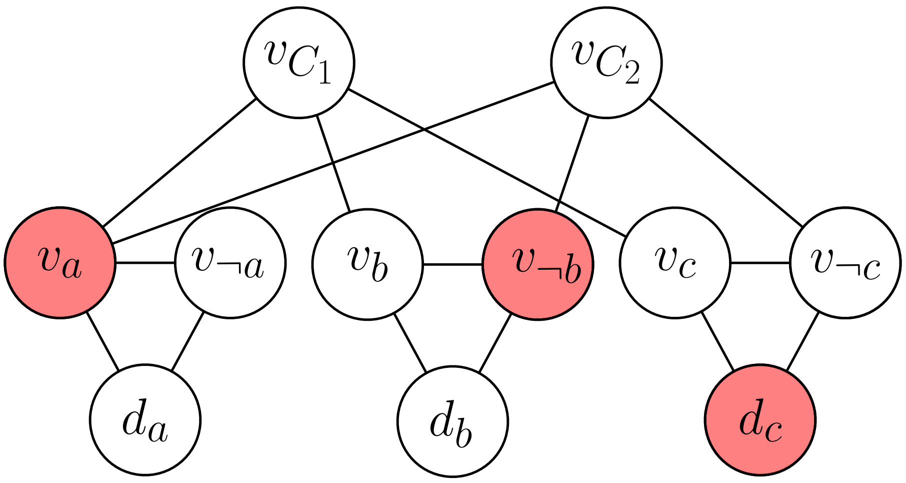

Figure 3: The graph $(a \vee b \vee c) \wedge (a \vee \neg b \vee c)$ reduces to

Proof. We first describe the reduction, then prove its correctness and argue for the running time and size of the resulting instance.

Let Φ be a 3-CNF formula over n variables $X := \{x_1, \ldots, x_n\}$ and clauses $\mathcal{C} := \{C_1, \ldots, C_m\}$, which are sets of literals of size 3. To obtain $G := (V, E)$ do the following:

- Let $k := n$.
- For each variable $x \in X$,
  - add a dummy vertex d_x, a vertex v_x for the positive literal and a vertex $v_{\neg x}$ for its dual to V and
  - add the edges $\{v_x, v_{\neg x}\}, \{v_x, d_x\}, \{\neg v_x, d_x\}$ to E.

- For each clause C,
  - add a vertex v_C to V and
  - for every $l \in C$, add an edge $\{v_c, v_l\}$ to E.

Next, I show that (G, n) is a YES-instance of Dominating Set if and only if Φ is a YES-instance of 3-SAT. First, *assume that Φ has a satisfying assignment I.* I claim that

$$D := \{v_{\neg x} | x \in X, I(x) = 0\} \cup \{v_x | x \in X, I(x) = 1\}$$

is a dominating set of size n. It has size n, as for each variable x, only either v_x or $v_\neg x$ is in D, which in turn means that d_x is dominated and v_x is dominated if $v_\neg x \in D$ and vice versa. As I is a satisfying assignment, C is satisfied, so only of its literals l must evaluate to true, so $v_l \in D$, so also v_C is dominated. Hence, all vertices are dominated or part of the dominating set.

To see why the other direction holds, please *assume that G has a dominating set D of size n.* Observe that there is a triangle consisting of $v_x, v_{\neg x}$ and d_x for each variable $x \in X$. D must include exactly one of $v_x, v_\neg x$ or d_x, as otherwise d_x would neither be dominated, nor in D. As $|D| = n$ and there are n such triangles, at most one vertex of each triangle is included in D. Thus exactly one vertex of each triangle is included in D. Because of that, the partial assignment I from D following the rules

- if $v_x \in D$ set $I(x) := 1$,
- if $v_\neg x \in D$ set $I(x) := 0$

for any $x \in X$ is well defined. We claim that any such I is a satisfying assignment to Φ. To see why, consider an arbitrary clause $C \in \mathcal{C}$ and call its literals l_1, l_2 and l_3. The clause's corresponding vertex v_C can not be included in D, as there are n triangles, each triangle requires at least one vertex to be picked, and $|D| = n$. Thus, either $v_{l_1} \in D$, $v_{l_2} \in D$ or $v_{l_3} \in D$. This means that any either $I(l_1) = 1$, $I(l_2) = 1$ or $I(l_3) = 1$, so C is satisfied. This concludes the proof of correctness.

Lastly, a constant time operation is done for each variable and clause, so the reduction runs in $O(n + m)$ time, and the resulting graph clearly has $3n + m$ vertices, namely one triangle per variable and one vertex per clause. $\square$

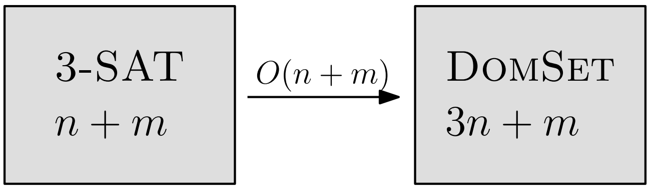

Figure 4: A graphical representation of the sparsification lemma

Theorem 3 ([3])**.** *Unless the ETH fails, there exists a $\delta > 0$ such that* Dominating Set *can not be solved in time $O(2^{\delta\left(|V|^{\frac{1}{3}}\right)})$.*

Proof. Aiming to contradict the ETH, assume that Dominating Set can be solved in time $O(2^{\delta\left(|V|^{\frac{1}{3}}\right)})$ for any $\delta > 0$. Let $\delta := \frac{\delta^*}{11}$, where δ^* is the δ^* from the ETH. By assumption, there exists an algorithm $\mathcal{A}$ that solves Dominating Set in time $O(2^{\delta\left(|V|^{\frac{1}{3}}\right)})$. We show that for any 3-CNF formula Φ, applying $\mathcal{A}$ on $G := (V, E)$ obtained by applying the previous reduction on Φ decides satisfiability in time $O(2^{\delta^* n})$. This would contradict the ETH, thus $\mathcal{A}$ can not exist, unless the ETH fails. To see why that is the case, recall that $|V| = 3n+m$. Φ can have at most all possible triples of variables and their respective variants with regards to negations as clauses, so $m < \binom{n}{3}8 \leq 8n^3$ and thus $|V| = 3n + m < 3n^3 + 8n^3 \leq 11n^3$. This means Φ could be solved in

$$O(n + m + 2^{\delta(11n)^{3^{\frac{1}{3}}}}) = O(2^{\frac{\delta^*}{11}11n}) = O(2^{\delta^* n})$$

time, which contradicts the ETH. □

This already constitutes a good blueprint for proving a conditional lower bound. However, it does not yet accomplish our goal of showing that no subexponential algorithms for Dominating Set exist, unless the ETH fails. The problem here is that we have not yet given a good upper bound for m. Because of that, the lower bound has a cubic root in its exponent. Luckily, there is very powerful lemma, which intuitively states that the hardness of deciding satisfiability for a 3-CNF Φ is only due to the number of its variables.

Theorem 4 (Sparsification lemma, simplified, [13]). *For any $\epsilon > 0$, every* 3-SAT *instance Φ of size $n + m$ can be turned into $t < 2^{\epsilon n}$ sub-instances in time $O(2^{\epsilon n} \cdot poly(n))$, such that*

- *each sub-instance has size $n + K(\epsilon)n$ and*
- *Φ is a YES-instance iff any of the sub-instances is a YES-instance.*

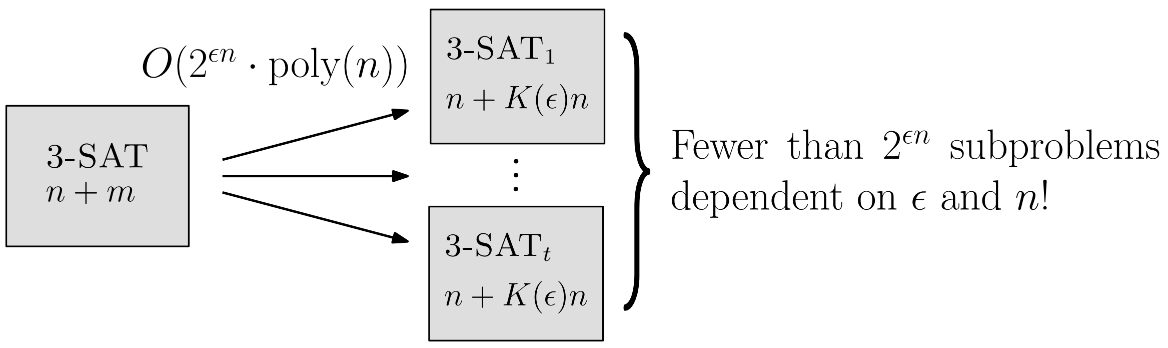
$O(2^{\epsilon n} \cdot \text{poly}(n))$
3-SAT
$n + m$
3-SAT$_1$
$n + K(\epsilon)n$
3-SAT$_t$
$n + K(\epsilon)n$
Fewer than $2^{\epsilon n}$ subproblems dependent on ϵ and n!

Figure 5: A graphical representation of the sparsification lemma

Using the sparsification lemma, we finally are ready to prove our statement.

Theorem 5 ([3]). *Unless the ETH fails, there exists a $\delta > 0$ such that* Dominating Set *can not be solved in time $O(2^{\delta|V|})$, that is,* Dominating Set *does not admit a subexponential algorithm.*

Proof. For a simpler proof with the same structure, consider the proof of Theorem 3. Let $\epsilon := \frac{\delta^*}{2}$ and $\delta := \frac{\delta^*}{2(3+K(\epsilon))}$. Again, aiming for a contradiction, assume that for any $\delta > 0$, so also for the δ defined, there exists an algorithm $\mathcal{A}$ solving Dominating Set in time $O(2^{\delta|V|})$. under this assumption, we claim that satisfiability can be decided for an arbitrary 3-SAT instance Φ in time $O(2^{\delta^* n})$ with the following algorithm $\mathcal{B}$:

1. Apply the sparsification lemma, that is, split Φ into $\Phi_1, \ldots, \Phi_t$ subinstances,
2. obtaining t Dominating Set instances $G_1, \ldots, G_t$ by applying reduction R from Theorem 2 and finally
3. apply $\mathcal{A}$ on each of Dominating Set instance $G_1, \ldots, G_t$ and return YES if $\mathcal{A}$ returns YES on any them, and NO otherwise.

First, I claim that the algorithm is correct. If Φ is a YES-instance, then one of $\Phi_1, \dots, \Phi_t$ is a YES-instance. R gives equivalent instances, so some of the subinstances $G_1, \dots, G_t$ is a YES-instance, so $\mathcal{A}$ returns YES, thus $\mathcal{B}$ returns YES . Likewise, if Φ is a NO-instance, none of $\Phi_1, \dots, \Phi_t$ and consequently none of $G_1, \dots, G_t$ is a YES-instance, so $\mathcal{B}$ returns NO .

Next, I show that algorithm $\mathcal{B}$ takes less than $O(2^{\delta^* n})$ time. First, note that $t < 2^{\epsilon n}$ by the sparsification lemma. The whole procedure can be conceptually broken down to *transforming* Φ *into* $G_1, \dots, G_t$ and then *applying* $\mathcal{A}$ *on* $G_1, \dots, G_t$.

1. The transformation, splits Φ into $\Phi_1, \dots, \Phi_t$, which takes $O(2^{\epsilon n} \cdot \text{poly}(n))$ time. Afterwards it computes $G_1, \dots, G_t$, which takes $O(t \cdot (n + m)) < O(2^{\epsilon n} \cdot (n + m))$ time. Overall, the transformation takes less than $O(2^{(\delta^*/2) \cdot n}(\text{poly(n)} + \text{m}))$ time.

2. Each of $G_1, \dots, G_t$ has $3n + K(\epsilon)n$ many vertices, so applying $\mathcal{A}$ on all of them takes time

$$\begin{aligned}
&O(t \cdot 2^{\delta(3n+K(\epsilon)n)}) \\
&< O(2^{\epsilon n} \cdot 2^{\delta(3n+K(\epsilon)n)}) && (t < 2^{\epsilon n}.) \\
&= O(2^{\epsilon n + \delta(3n+K(\epsilon)n)}) \\
&= O(2^{\delta^* n}). && \left(\epsilon n + \delta(3n + K(\epsilon)n) = \frac{\delta^*}{2} n + \frac{\delta^*(3 + K(\epsilon))n}{2(3 + K(\epsilon))} = \delta^* n.\right)
\end{aligned}$$

Considering the exponents, $\delta^* > \frac{\delta^*}{2}$, so only the second step is of asymptotic significance. Hence, $\mathcal{B}$ takes less than $O(2^{\delta^* n})$ time. If we believe in the ETH, this is a contradiction, so $\mathcal{A}$ can not exist. $\square$

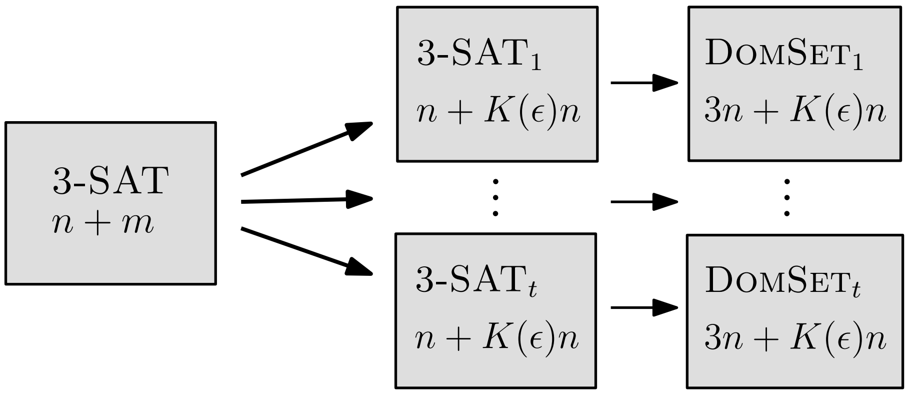

Figure 6: Graphical representation of of how to build a subexponential algorithm for 3-SAT with a subexponential algorithm for Dominating Set using the previous reduction and the sparsification lemma.

We have experienced the proof of a good conditional lower bound based on the ETH. Can we generalize the above? First, let us consider a more formal definition of a fine-grained reduction.

Definition 7 ((s, t)-reduction, simplified [19])**.** Let $A, B \subseteq \{0, 1\}^*$ be two problems and $s, t : \mathbb{N} \to \mathbb{N}$. An (s, r)-reduction from A to B is an algorithm that, given an instance x of A, outputs an instance x' such that

- x is a YES-instance of A if and only if x' is a YES-instance of B,
- $|x'| \leq s(|x|)$ and
- the running time of the algorithm is bounded by $t(|x|)$.

There are two big disadvantages of considering such fine-grained reductions. Notably, if for example A (n^2, n^2)-reduces to B, and B (n^2, n^2)-reduces to C, then A does *not* automatically (n^2, n^2) reduce to C, because composing the two reductions yields a $(n^4, 2n^2)$-reduction. Thus, (s,t)-reducibility is not transitive. Thus, the network of reductions thus needs to be considered carefully, because in some sense, (s,t)-reducibility is a local phenomenon. Further, one reason why many-one reductions are so useful is that the model of computation does not matter; according to the Church-Turing Thesis [1, section 1.6.1], all models of computation can simulate each other with at most *polynomial slowdown*, of which the slow-down is irrelevant for many-one reductions, but not so for fine-grained reductions. However, the specificity of an (s,t)-reduction allows us to give more general lower-bounds. The result shown for Dominating Set can be generalized, namely for *linear reductions*, which can be considered as a corollary of the sparsification lemma. This statement is even more powerful than it seems, as linear reductions are transitive.

Definition 8. A *linear reduction* is an (s,t)-reduction such that $s : \mathbb{N} \to \mathbb{N}$ is a linear function and $t : \mathbb{N} \to \mathbb{N}$ is a polynomial function.

Theorem 6 ([12])**.** *Unless the ETH fails, no problem which has a linear reduction from* 3-SAT *can be solved in subexponential time.*

Proof. Suppose 3-SAT has a linear reduction from problem A, transforming a 3-CNF formula Φ over n variables and m clauses to an instance x of A. The instance x' obtained from the reduction has size $|x'| \leq s(|\Phi|) = an + bm$ for some $a, b \in \mathbb{N}$. Thus, the very same proof as for Theorem 5 works, however with $\delta := \frac{\delta^*}{2(a+b\cdot K(\epsilon))}$. □

To conclude this section, we give a few problems which have a linear reduction from 3-SAT and thus no subexponential algorithms assuming ETH, without formally introducing them.

Corollary 1 ([9, Theorem 14.6])**.** *As the following problems have linear reductions from* 3-SAT*, none of them have subexponential algorithms, unless ETH fails:*

- Vertex Cover*,*
- Hamiltonian Cycle*, and*
- 3-Coloring*.*

The fact that 3-Coloring admits no subexponential algorithms under the ETH turned out to result in a fundamental fact about another branch of complexity theory, namely *parameterized complexity.*

3 Parameterized Complexity

Parameterized Complexity is a pragmatic take on computational complexity, of which the philosophy can be best explained by an example.

Example 3. Consider elections; typically *many voters* vote on *few alternatives.* Many problems about elections, such as manipulation [10], are NP-complete. However, as noted before, there are typically few alternatives to vote on. Thus, an algorithm $\mathcal{A}$ that increases exponentially *only when the number of alternatives increases* would be highly desirable. Such algorithms (and their respective problems) are called *fixed parameter tractable* (FPT) with respect to the number of alternatives, because if the number of alternatives, which is the *parameter* of those *parameterized problems*, is fixed, then the running time of the algorithm increases at most polynomially if the rest of the instance grows. In our case, this would allow millions of voters to join the elections without that leading to exponential slowdowns of $\mathcal{A}$. In fact, some forms of the example given, manipulation, are [2] FPT with respect to the number of alternatives.

The aim of this section is to illustrate the similarities of classical computational complexity and parameterized complexity, and finally show that the ETH acts as a unifying hypothesis for both of them. Let us begin with the central definition of the field.

Definition 9 ([9, Def 1.2])**.** A *parameterized problem* $A \subseteq \{0,1\}^* \times \mathbb{N}$ is called *fixed parameter tractable* with respect to its parameter if there exists an algorithm which decides if $(x,k) \in (\{0,1\}^* \times \mathbb{N})$ is an element of A in time

$$f(k) \cdot \text{poly}(|x|),$$

where $f : \mathbb{N} \to \mathbb{N}$ is a computable function. The class of all fixed parameter tractable problems is called *FPT*.

Let us see an example of a fixed parameter tractable problem; its parameterization is definitely *not useful*, yet the FPT-algorithm is straight-forward.

Proposition 2. 3-SAT *parameterized by the number of variables n is fixed parameter tractable.*

Proof. Let Φ be a 3-CNF with variables $x_1, \ldots, x_n$. Checking for each I of the 2^n variable assignments if $I(\Phi) = 1$ needs $2^n \cdot \text{poly}(|x|)$ time and decides Φ's satisfiability correctly. This concludes the proof already. $\square$

For an extensive discussion on fixed parameter tractability, I refer the reader to the book by Cygan et al. [9]. The goal of this section is to introduce W[1], a class of parameterized problems, allude to how it resembles NP in FPT-land and show that unless the ETH fails, no problem which has a *parameterized reduction* from CLIQUE that is linear in the parameter admits an algorithm of worst-case complexity $f(k) \cdot |x|^{o(k)}$. The latter point is of specific interest, as it shows that the ETH translates very well to parameterized complexity.

Reductions are defined for parameterized problems, but we shall see that they is not as nicely behaved as many-one reductions.

Definition 10 ([9, Def 13.1])**.** Let $A, B \subseteq \{0,1\}^* \times \mathbb{N}$ be two parameterized problems. A *parameterized reduction* from A to B is an algorithm that, given an instance (x,k) of A, outputs an instance (x', k') of B such that

- (x,k) is a YES-instance of A if and only if (x', k') is a YES-instance of B,
- $k' \leq g(k)$ for some computable function g and
- the running time of the algorithm is $f(k) \cdot \text{poly}(|x|)$.

Under this notion, an FPT algorithm for A yields an FPT algorithm for B, whilst the disbelief for an FPT algorithm for A transfers that disbelief to B.

Theorem 7. *[9, Thm 13.2] If there is a parameterized reduction from A to B and B admits an FPT algorithm, then A also admits an FPT algorithm.*

Unfortunately, under this notion of a reduction, the FPT-analogue to NP is not one class, but a hierarchy of classes, called the W-hierarchy, consisting of classes W[1], W[2], ... such that all of W[i] reduces to W[j] if $j \geq i$, but not the other way around. However, this inconvenience is not relevant for our concerns: just as an FPT-algorithm propagates down the hierarchy, a lower bound for W[1] conversely propagates upward the W-hierarchy. Thus, I do not give the canonical definition of W[1], which is based on problems that reduce and can be reduced to and from a specific problem on circuits of *weft* 1. For an extensive treatment thereof, I recommend [9, Chapter 13]. For the topic of this paper, it is useful to think of W[1] as the problems that reduce to SHORT TURING MACHINE ACCEPTANCE, which is W[1]-complete (see [9, Thm 13.32]) also under the canonical definition.

Short Turing Machine Acceptance

Input: A nondeterministic Turing Machine M, an input string $x \in \{0,1\}^*$ and a parameter $k \in \mathbb{N}$

Question: Does M have a computation path of length at most k accepting x?

Having seen this, we are ready to see our definition of W[1].

Definition 11. A parameterized problem A is contained in W[1] if it reduces to Short Turing Machine Acceptance. A is W[1]-*hard* if all problems in W[1] reduce to A. A is W[1]-*complete* if it is both W[1]-hard and contained in W[1].

This following theorem shows that our given definition gives FPT $\subseteq$ W[1] and may also serve as a very gentle introduction to parameterized reductions.

Theorem 8. *$FPT \subseteq W[1]$.*

Proof. Assume A is a parameterized problem with an FPT algorithm $\mathcal{A}$. To decide if some instance $(x,k) \in A$, run $\mathcal{A}$ on it. If x is a YES-instance according to $\mathcal{A}$, let M be the NDTM that instantly accepts. Otherwise, let M be the NDTM that instantly rejects. Thus, $x' := (M, k')$ with $k' := 1$ is the obtained Short Turing Machine Acceptance instance. It takes $f(k) \cdot \text{poly}(|x|)$ time to compute x', $k' \leq k$ and the x' is clearly equivalent to x. $\square$

The working horse of NP is 3-SAT; a problem with "nice" structure that is useful for reductions and for which there exists a (tedious) reduction to the defining problem Turing Machine Acceptance. What is the analogue for W[1]? The answer is Clique.

Clique

Input: A graph $G = (V, E)$, a parameter k

Question: Does there exist a subset $C \subseteq V$ of size k such that $\forall a, b \in C : \{a, b\} \in E$?

Cai et al. [5] have shown that Short Turing Machine Acceptance reduces to Clique and vice versa. Thus, the following can be treated as an analog to the Cook-Levin Theorem for parameterized complexity, even though it is not as paramount, as the "real" underlying problem (or collection thereof) of the W-hierarchy is a different one.

Theorem 9 (Cai et al. [5])**.** Clique *is W[1]-complete.*

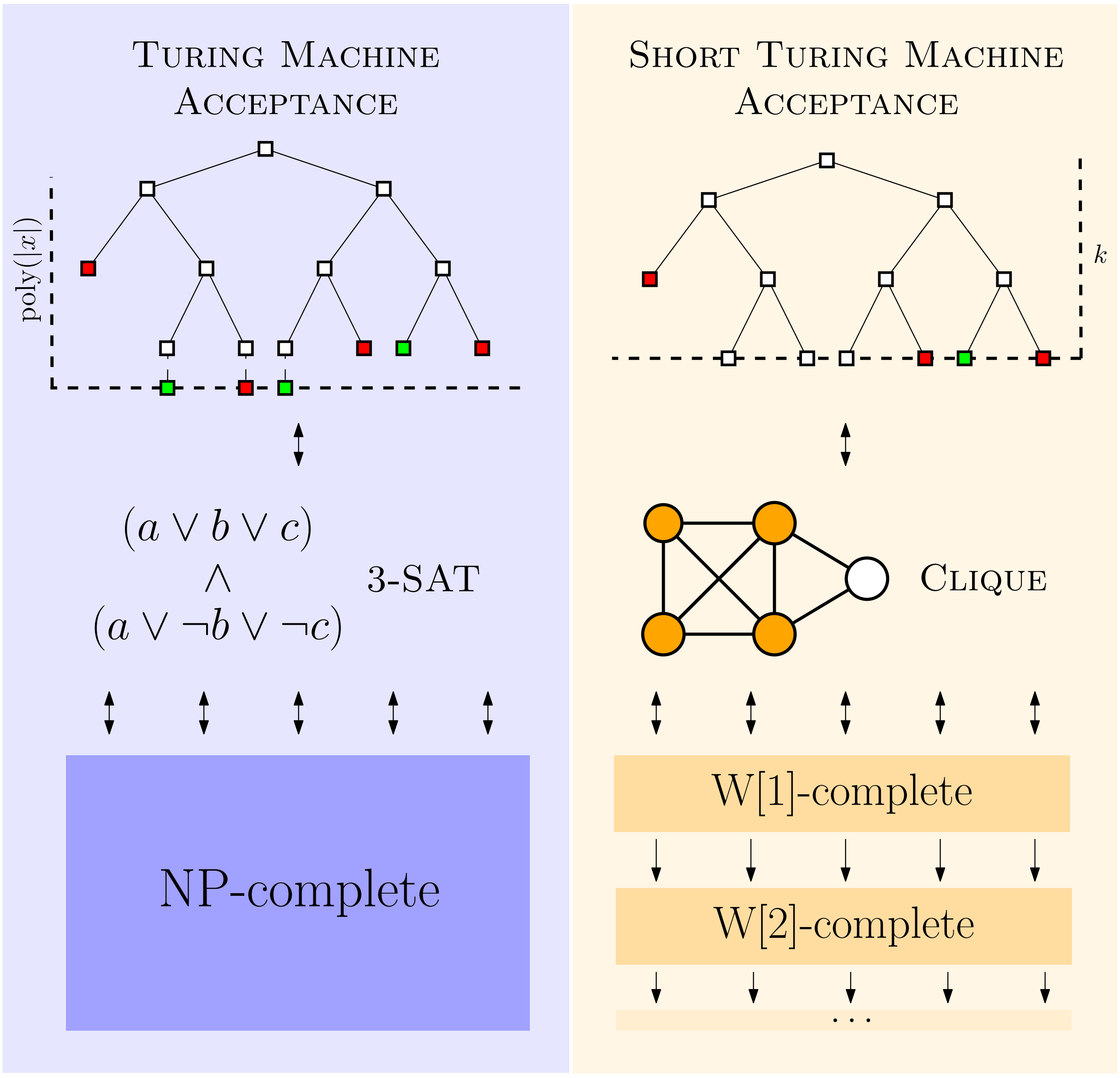

Figure 7: An illustrative comparison of classical computational complexity and parameterized complexity; a one-sided arrow indicates that the problem (or the class of problems) at the tail reduces to the other, a two-sided arrow means there is a reduction for both directions.

Clique is a well-studied problem. In fact, it is one of the original 21 NP-complete problems by Karp [15]. Nevertheless, no FPT-algorithm has been found for it yet. We thus have that

- an FPT-algorithm for a relatively well-structured and well-studied problem has been not been found yet and
- such an algorithm would give an FPT-algorithm for something as opaque as deciding if a polynomial-time NDTM has accepting computation paths of a given length.

This is evidence for another widely believed conjecture, namely that FPT $\neq$ W[1].

Hypothesis 3. *FPT $\neq$ W[1], that is, no problem in W[1] is fixed parameter tractable.*

However, here the symmetry breaks down, because FPT $\neq$ W[1] is a stronger conjecture than P $\neq$ NP. This is when we finally return to the main topic of this paper, the ETH; in fact, the ETH implies both FPT $\neq$ W[1] (see Corollary 2) and P $\neq$ NP (trivially), and in some sense is a conjecture over both classical complexity theory and parameterized complexity theory. Further, the ETH carries over through a series of proofs to the working-horse of parameterized complexity, because a $f(k) \cdot |V|^{o(k)}$ algorithm for Clique would imply a subexponential algorithm for 3-Colorability, which would contradict the ETH.

Theorem 10 ([7], [9, Thm 14.21])**.** *Unless the ETH fails,* Clique *does not admit an $f(k) \cdot |V|^{o(k)}$ algorithm.*

Corollary 2. *ETH $\implies$ FPT $\neq$ W[1].*

Proof. Suppose the ETH is true and let A be a parameterized problem in W[1]. By definition, A can be reduced to Clique. An FPT algorithm for A would thus allow for an FPT algorithm for Clique, contradicting the ETH. □

Corollary 3. *If a parameterized problem A has a parameterized reduction from* Clique*, transforming an instance (x, k) from* Clique *to an instance (x', k'), that is linear in the parameter, i.e., $k' \leq ak$ for some $a \in \mathbb{N}$, then A has no algorithms of complexity $g(k') \cdot |x|^{o(k')}$ for no computable function g, unless the ETH fails.*

Proof. The reduction can be used to obtain an instance (x', k') of A that is equivalent to Clique in $f(k') \cdot \text{poly}(|x|)$. If A had an algorithm of complexity $f(k') \cdot |x|^{o(k')}$, then this would yield an algorithm of complexity $g(ak') \cdot |x|^{ak} = f(k) \cdot |x|^{o(k)}$, with $f(k) := g(ak)$, which is impossible by Theorem 10, unless the ETH fails. □

4 Closing Remarks

We have very briefly taken a look at three different flavors of complexity theory under the guise of the Exponential Time Hypothesis. In some sense, classical complexity asks "*Is* a problem hard?", fine grained complexity asks the question "*How* hard is a problem?" by restricting the notion of a reduction, whereas parameterized complexity asks "*What* about a problem is hard?" by restricting the set of possible instances. At their intersection lie results like that of Theorem 3, which consider both more specific questions simultaneously. In some sense, the two subfields represent two different dimensions of hardness, with the ETH acting as a bridge connecting the two dimensions.

This quick excursion was not exhaustive. There is a stronger version of the ETH, the *Strong Exponential Time Hypothesis* (SETH) [13], which, states that SAT, so 3-SAT without a restriction on the clause length, has no algorithm that has a better time complexity than brute force. Whilst less commonly agreed upon, SETH gives stronger lower bounds and can be used to prove sharper lower bounds, even for problems that are not in P [19]. The ETH and SETH can both be expanded to the world of randomization [6] and also to quantum computing [4], so it seems as if research around it is ongoing. Despite being relatively new, parameterized complexity has left its infancy and is a mature field, yet still with open questions, also with respect to hardness. Fine grained complexity is not as matured, however therefore still offers much to be explored.

References

[1] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* 1st ed. USA: Cambridge University Press, Mar. 2009. 608 pp. isbn: 978-0-521-42426-4.

[0]This seminar paper was inspired in part by lecture notes about Fine-Grained Complexity Theory by Bringmann, Künnemann, and Wellnitz [3].

[2] Robert Bredereck et al. “Parameterized Algorithmics for Computational Social Choice: Nine Research Challenges”. In: *Tsinghua Science and Technology* 19.4 (Aug. 2014), pp. 358–373. ISSN: 1007-0214. DOI: `10.1109/TST.2014.6867518`. URL: `https://ieeexplore.ieee.org/abstract/document/6867518` (visited on 06/28/2024).

[3] Karl Bringmann, Marvin Künnemann, and Phillip Wellnitz. *Lecture notes in Fine-Grained Complexity Theory.* 2019.

[4] Harry Buhrman, Subhasree Patro, and Florian Speelman. *The Quantum Strong Exponential-Time Hypothesis.* Nov. 14, 2019. arXiv: `1911.05686 [quant-ph]`. URL: `http://arxiv.org/abs/1911.05686` (visited on 04/11/2024). Pre-published.

[5] Liming Cai et al. “On the Parameterized Complexity of Short Computation and Factorization”. In: *Archive for Mathematical Logic* 36.4-5 (Aug. 1, 1997), pp. 321–337. ISSN: 0933-5846, 1432-0665. DOI: `10.1007/s001530050069`. URL: `http://link.springer.com/10.1007/s001530050069` (visited on 06/28/2024).

[6] Marco L. Carmosino et al. “Nondeterministic Extensions of the Strong Exponential Time Hypothesis and Consequences for Non-reducibility”. In: *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science.* ITCS’16: Innovations in Theoretical Computer Science. Cambridge Massachusetts USA: ACM, Jan. 14, 2016, pp. 261–270. ISBN: 978-1-4503-4057-1. DOI: `10.1145/2840728.2840746`. URL: `https://dl.acm.org/doi/10.1145/2840728.2840746` (visited on 04/03/2024).

[7] Jianer Chen et al. “Strong Computational Lower Bounds via Parameterized Complexity”. In: *Journal of Computer and System Sciences* 72.8 (Dec. 1, 2006), pp. 1346–1367. ISSN: 0022-0000. DOI: `10.1016/j.jcss.2006.04.007`. URL: `https://www.sciencedirect.com/science/article/pii/S0022000006000675` (visited on 06/28/2024).

[8] Stephen A. Cook. “The Complexity of Theorem-Proving Procedures”. In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing.* STOC ’71. New York, NY, USA: Association for Computing Machinery, May 3, 1971, pp. 151–158. ISBN: 978-1-4503-7464-4. DOI: `10.1145/800157.805047`. URL: `https://doi.org/10.1145/800157.805047` (visited on 06/27/2024).

[9] Marek Cygan et al. *Parameterized Algorithms.* Cham: Springer International Publishing, 2015. ISBN: 978-3-319-21274-6 978-3-319-21275-3. DOI: `10.1007/978-3-319-21275-3`. URL: `https://link.springer.com/10.1007/978-3-319-21275-3` (visited on 04/02/2024).

[10] Jessica Davies et al. “Complexity of and Algorithms for Borda Manipulation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 25.1 (1 Aug. 4, 2011), pp. 657–662. ISSN: 2374-3468. DOI: `10.1609/aaai.v25i1.7873`. URL: `https://ojs.aaai.org/index.php/AAAI/article/view/7873` (visited on 06/28/2024).

[11] Timon Hertli. “3-SAT Faster and Simpler—Unique-SAT Bounds for PPSZ Hold in General”. In: *SIAM Journal on Computing* 43.2 (Jan. 2014), pp. 718–729. ISSN: 0097-5397. DOI: `10.1137/120868177`. URL: `https://epubs.siam.org/doi/abs/10.1137/120868177` (visited on 05/13/2024).

[12] Russell Impagliazzo and Ramamohan Paturi. “On the Complexity of K-SAT”. In: *Journal of Computer and System Sciences* 62.2 (Mar. 1, 2001), pp. 367–375. ISSN: 0022-0000. DOI: `10.1006/jcss.2000.1727`. URL: `https://www.sciencedirect.com/science/article/pii/S0022000000917276` (visited on 04/03/2024).

[13] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. “Which Problems Have Strongly Exponential Complexity?” In: *Journal of Computer and System Sciences* 63.4 (Dec. 1, 2001), pp. 512–530. ISSN: 0022-0000. DOI: `10.1006/jcss.2001.1774`. URL: `https://www.sciencedirect.com/science/article/pii/S002200000191774X` (visited on 04/03/2024).

[14] Kazuo Iwama and Suguru Tamaki. “Improved Upper Bounds for 3-SAT”. In: *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’04. USA: Society for Industrial and Applied Mathematics, Jan. 11, 2004, p. 328. ISBN: 978-0-89871-558-3.

[15] Richard M. Karp. “Reducibility among Combinatorial Problems”. In: *Complexity of Computer Computations: Proceedings of a Symposium on the Complexity of Computer Computations, Held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and Sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*. Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2001-2. DOI: `10.1007/978-1-4684-2001-2_9`. URL: `https://doi.org/10.1007/978-1-4684-2001-2_9` (visited on 06/28/2024).

[16] B. Monien and E. Speckenmeyer. “Solving Satisfiability in Less than 2n Steps”. In: *Discrete Applied Mathematics* 10.3 (Mar. 1, 1985), pp. 287–295. ISSN: 0166-218X. DOI: `10.1016/0166-218X(85)90050-2`. URL: `https://www.sciencedirect.com/science/article/pii/0166218X85900502` (visited on 05/13/2024).

[17] Robin A. Moser and Dominik Scheder. “A Full Derandomization of Schöning’s k-SAT Algorithm”. In: *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*. STOC ’11. New York, NY, USA: Association for Computing Machinery, June 6, 2011, pp. 245–252. ISBN: 978-1-4503-0691-1. DOI: `10.1145/1993636.1993670`. URL: `https://dl.acm.org/doi/10.1145/1993636.1993670` (visited on 05/12/2024).

[18] T. Schoning. “A Probabilistic Algorithm for K-SAT and Constraint Satisfaction Problems”. In: *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*. 40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039). Oct. 1999, pp. 410–414. DOI: `10.1109/SFFCS.1999.814612`. URL: `https://ieeexplore.ieee.org/abstract/document/814612?casa_token=5UyB_EZzEUEAAAAA:JwuhLtu1cbAYRw8qStPLUWoUx2QbRwmZc3HEF-hDisi9UM7c-jAyf3aHiYP-Fwwe_kLNDC9svYQ` (visited on 05/13/2024).

[19] Virginia Vassilevska Williams. “Hardness of Easy Problems: Basing Hardness on Popular Conjectures Such as the Strong Exponential Time Hypothesis (Invited Talk)”. In: *DROPS-IDN/v2/Document/10.423* 10th International Symposium on Parameterized and Exact Computation (IPEC 2015). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. DOI: `10.4230/LIPIcs.IPEC.2015.17`. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.IPEC.2015.17` (visited on 04/03/2024).