Theory of Computing Systems
https://doi.org/10.1007/s00224-023-10119-x

Effective Guessing Has Unlikely Consequences

András Z. Salamon[1] · **Michael Wehar**[2]

Accepted: 21 January 2023

Abstract

A classic result of Paul, Pippenger, Szemerédi and Trotter states that $\mathsf{DTIME}(n) \subsetneq \mathsf{NTIME}(n)$. The natural question then arises: could the inclusion $\mathsf{DTIME}(t(n)) \subseteq \mathsf{NTIME}(n)$ hold for some superlinear time-constructible function $t(n)$? If such a function $t(n)$ does exist, then there also exist effective nondeterministic guessing strategies to speed up deterministic computations. In this work, we prove limitations on the effectiveness of nondeterministic guessing to speed up deterministic computations by showing that the existence of effective nondeterministic guessing strategies would have unlikely consequences. In particular, we show that if a subpolynomial amount of nondeterministic guessing could be used to speed up deterministic computation by a polynomial factor, then $\mathsf{P} \subsetneq \mathsf{NTIME}(n)$. Furthermore, even achieving a logarithmic speedup at the cost of making every step nondeterministic would show that $\mathrm{SAT} \in \mathsf{NTIME}(n)$ under appropriate encodings. Of possibly independent interest, under such encodings we also show that SAT can be decided in $O(n \log n)$ steps on a nondeterministic multitape Turing machine, improving on the well-known $O(n(\log n)^c)$ bound for some constant but undetermined exponent $c \geq 1$.

Keywords Computational complexity · Structural complexity · Limited nondeterminism · Effective guessing · Complexity class containments

This article belongs to the Topical Collection: *Commemorative Issue for Alan L. Selman*
Guest Editors: Mitsunori Ogihara, Elvira Mayordomo, Atri Rudra

✉ András Z. Salamon
Andras.Salamon@st-andrews.ac.uk

Michael Wehar
mwehar1@swarthmore.edu

[1] School of Computer Science, University of St Andrews, St Andrews, UK

[2] Computer Science Department, Swarthmore College, Swarthmore, PA, USA

Published online: 25 March 2023

1 Dedication for Alan L. Selman

This paper is dedicated to the memory of Alan L. Selman. The second author was a student in Professor Alan Selman's graduate course titled *Introduction to the Theory of Computation* at University at Buffalo during Fall 2013. Of the many fond memories of Professor Selman's instruction, his careful treatment of computational concepts and their technical details especially stood out. During the course, Professor Selman's discussion of one particular topic is most clearly remembered and directly relates to this work, namely the topic of Linear Speedup Theorems for both deterministic and nondeterministic Turing machines (which can be found in Section 5.1 of [1]). This discussion from Professor Selman is particularly relevant and motivating to this paper because the following investigates when additional nondeterminism could lead to improved efficiency, and the proof of the Linear Speedup Theorem for nondeterministic Turing machines (Theorem 5.3 of [1]) provides an example where the use of additional nondeterminism leads to a slightly faster simulation.

2 Introduction

How powerful is nondeterminism? To make progress investigating this general philosophical question we have to consider a more focused technical question: *when is it possible to replace some portion of a deterministic computation by nondeterministic guessing to reduce the total computation time?*

The Linear Speedup Theorem tells us that computations can be sped up by any constant factor by using larger tape alphabets [2, Theorem 2]. Conversely, the tight form of the Deterministic Time Hierarchy Theorem shows that it is not generally possible to achieve a speedup of more than a constant factor [3]. These classic results leave open the possibility that a computation could be further sped up by increasing some other resource such as nondeterminism.

Along with the example from Section 1, there are some cases where additional nondeterminism is known to speed up computation. SAT is an example of a language for which, unless $\mathsf{P} = \mathsf{NP}$, we can speed up a deterministic decision procedure superpolynomially by instead guessing an assignment and verifying that the guess satisfies every clause. Another example is deciding if an item occurs in a list. With a random access model of computation, search can be sped up exponentially by guessing the position of the item in the input list and verifying this guess. However, to the best of our knowledge, no general speed-up result has been proven for languages in P.

3 Main Contributions

We express our results in terms of complexity classes defined by joint bounds on time and nondeterminism. Let $\mathsf{NTIGU}(t(n), w(n))$ denote the class consisting of languages which can be decided by multitape nondeterministic Turing machines

operating with time bound $O(t(n))$ and using at most $w(n)$ nondeterministic bits, for n-bit inputs. (We follow prior usage with this definition [4].)

An *effective guessing hypothesis* is a statement of the form: it is possible to speed up a computation by using more nondeterminism. We present two main results conditional on two different effective guessing hypotheses, one somewhat stronger than the other.

Our first result is that if even a small polynomial speedup can be achieved by introducing a polylogarithmic amount of nondeterminism, then we could decide all of P using nondeterministic linear time.

Theorem 1 *If there is some constant $c > 1$ such that*

$$\mathit{DTIME}(n^c) \subseteq \mathit{NTIGU}(n, \text{polylog}(n)),$$

then $\mathsf{P} \subsetneq \mathit{NTIME}(n)$.

Our second result greatly weakens the effective guessing hypothesis while still yielding a surprising conclusion. In particular, the premise is weakened from polynomial to logarithmic speedup and from polylogarithmic to linear size witnesses. Being able to speed up computation by a logarithmic factor, even at the cost of making essentially every step nondeterministic, would imply a breakthrough nondeterministic algorithm for SAT on multitape Turing machines. This kind of effective guessing would allow us to overcome a barrier to progress that has stood for more than four decades.

Theorem 2 *If $\mathit{DTIME}(n \log n) \subseteq \mathit{NTIME}(n)$, then* $\text{SAT} \in \mathit{NTIME}(n)$.

Along the way to proving Theorem 2 we also derive an improved upper bound for SAT.

Theorem 3 $\text{SAT} \in \mathit{NTIGU}(n \log n, O(n/\log n))$.

This improves a well-known upper bound $\text{SAT} \in \mathsf{NTIME}(n(\log n)^c)$ for some constant $c \geq 1$ that is not explicitly stated in the literature, obtained either by a direct argument [5] or by using a random-access machine to perform the obvious guess-and-check algorithm in linear time, and using a standard simulation of RAMs by Turing machines [6]. Theorem 3 allows us to take $c = 1$.

4 Relationship to Prior Work

Some speedups are known to be impossible. This is the case for nondeterministic computations, which cannot be sped up polynomially by using a moderate amount of advice. In particular, Fortnow, Santhanam, and Trevisan showed that $\mathsf{NP} \not\subseteq \mathsf{NTIME}(n^c)/(\log n)^{1/2c}$ for all c [7], and this result was extended by Fortnow and Santhanam to polynomial speedup and advice [4].

Our work fits into the long tradition of conditional separations and containments of complexity classes. Previous work typically exploits classes that make nonuniform use of circuits. This includes Impagliazzo and Wigderson's result that if E requires circuits of exponential size for infinitely many input sizes, then BPP = P [8], Fortnow and Santhanam's strengthening of the nondeterministic time hierarchy theorem in the presence of advice [4], and the unconditional separation of Williams of ACC circuits of polynomial size and NEXP which was achieved by proving two conditional containments and then combining them to yield a contradiction [9]. In contrast, we focus here on subclasses of classical nondeterministic time classes which are defined by bounding the amount of nondeterminism. Our Theorem 1 is also a significant extension of a result sketched by Bloch, Buss, and Goldsmith, weakening the effective guessing hypothesis used in their work from logarithmic to polylogarithmic nondeterminism [10].

Our work further relates to questions raised by the classical result that $\mathsf{DTIME}(n) \subsetneq \mathsf{NTIME}(n)$ of Paul et al. [11]. This was obtained by assuming that $\mathsf{DTIME}(n)$ and $\mathsf{NTIME}(n)$ coincide, and then trading off an increase in alternations to obtain a speedup which contradicts a hierarchy theorem. Beginning with this strict containment, it is then natural to consider whether $\mathsf{DTIME}(t(n)) \subseteq \mathsf{NTIME}(n)$ for any superlinear time-constructible function $t(n)$. Our Theorem 2 demonstrates that a positive answer to this question for even a mildly superlinear function such as $t(n) = n \log n$ would lead to a breakthrough for SAT. Furthermore, from a form of the nondeterministic time hierarchy theorem [12, Corollary 2.3], the strict complexity class containment $\mathsf{DTIME}(n \log n) \subsetneq \mathsf{NTIME}(n \log n)$ would straightforwardly follow. Although Paul et al. [11] showed that the strict containment $\mathsf{DTIME}(t(n)) \subsetneq \mathsf{NTIME}(t(n))$ holds for $t(n) = n$, and Santhanam extended this to any $t(n) = o(n \lg^* n)$ [13, Theorem 2.5], such a result is not known for functions $t(n)$ that grow at least as fast as $n \lg^* n$. (The iterated logarithm $\lg^* x$ is the minimal height of a tower of 2s representing a number at least as large as x.) Note that we do not use the alternation-trading technique from [11] in our work.

5 Overview of Paper: Intuitions Behind Our Arguments

In Section 6, we first define time-witness classes as a technically convenient method of dealing with computations that limit the amount of nondeterminism. These classes have similarities with advice classes, and essentially treat the guess as part of the input. Our use of the existential projection allows for straightforward accounting of the nondeterministic bits when composing simulations.

In Section 7, we prove Theorem 1. This requires some machinery to precisely relate the speedup in each simulation step to the increase in witness size. Our arguments in this section rely on subpolynomial functions being closed under composition, and a Strong Speedup Lemma (Lemma 11) for exact witness size bounds.

In Section 8, we prove Theorem 2. The key is the Weak Speedup Lemma (Lemma 15) which allows witness size bounds up to arbitrary constant factors. This lemma allows us to transfer an effective guessing hypothesis from deterministic to

nondeterministic computations. We provide an upper bound on the number of distinct variables that can be contained in an n-bit SAT instance when using a reasonable encoding. This leads to a more precise upper bound for the time taken to decide SAT on a nondeterministic Turing machine (Theorem 3). We will also need to make more precise the classical time upper bounds for sorting on a deterministic Turing machine.

Theorem 1 relies on the Strong Speedup Lemma (Lemma 11) and Theorem 2 relies on the Weak Speedup Lemma (Lemma 15). These lemmas are closely related but use incomparable hypotheses so require separate proofs. Both speedup lemmas use the same intuition: if we assume some form of effective guessing hypothesis, then we can apply that hypothesis to speed up the deterministic verification step of a guess-and-check computation. Our proofs make this intuition precise by using the time-witness class definitions to ensure that the increase in witness size is appropriately bounded.

In Section 9, we discuss some final thoughts and outline directions for further work.

6 Preliminaries

With $\mathbb{N}$ we mean the set $\{0, 1, 2, \dots\}$. We assume a fixed alphabet $\Sigma = \{0, 1\}$ throughout. We also use the notation $\lg x = log_2 x$ throughout. For $x, y \in \Sigma^*$, the expression $\langle x, y\rangle$ simply denotes the bits of x followed by those of y, also known as concatenation. This guarantees associativity: $\langle x, \langle y, z\rangle\rangle = \langle\langle x, y\rangle, z\rangle$. For a word $x \in \Sigma^*$, we denote by $|x|$ the number of symbols in x, which may be 0 if x is the empty word. Hence $|\langle x, y\rangle| = |x| + |y|$. In a slight abuse of notation, when $f : \mathbb{N} \to \mathbb{N}$ is a function we will often write $f(n)$ to emphasize this fact, rather than just f, and when c is a constant, we will sometimes use c to denote the constant function $c(n) = c$. With SAT we mean the Boolean Satisfiability problem for Boolean formulas in conjunctive normal form (CNF).

Our exposition is based on the concept of existential projection. We will use existential projections to define classes of languages with combined time and witness size bounds. This approach simplifies the bookkeeping required to track witness sizes in composed simulations, yet these classes are closely related to more familiar complexity classes. We first define the notion and then discuss some consequences and an example.

Definition 4 Given a language $L \in \Sigma^*$ and a function $w : \mathbb{N} \to \mathbb{N}$, the *existential projection* of L by w is the language

$$L[w(n)] = \{x \mid x \in \Sigma^*, \exists y \in \Sigma^{w(|x|)} \langle x, y\rangle \in L\}.$$

Further, for functions $f(n)$ and $g(n)$ let $L[f(n), g(n)] = (L[f(n)])[g(n)]$ as a convenient notation for composition of existential projections.

The witness size function w represents some portion of each word in the language which is set aside to record choices made by a nondeterministic computation. The existential projection then removes this portion of each word.

We now make precise how Definition 4 affects witness size changes in composition of existential projections.

Lemma 5 *For any language $L \in \Sigma^*$, the identity*

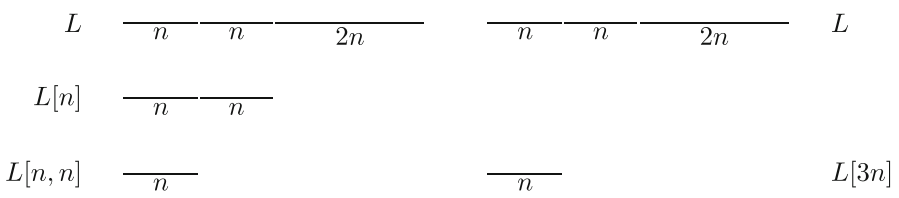
$$L[f(n), g(n)] = L[g(n) + f(n + g(n))]$$

holds for all functions $f, g : \mathbb{N} \to \mathbb{N}$.

Proof Suppose $x \in L[f(n), g(n)] = (L[f(n)])[g(n)]$. Then there is some $y \in \Sigma^*$ such that $\langle x, y\rangle \in L[f(n)]$ and $|y| = g(|x|)$. Further, there is some $z \in \Sigma^*$ such that $\langle x, \langle y, z\rangle\rangle = \langle\langle x, y\rangle, z\rangle \in L$ such that $|z| = f(|\langle x, y\rangle|) = f(|x| + |y|) = f(|x| + g(|x|))$. Hence $|\langle y, z\rangle| = |y| + |z| = g(|x|) + f(|x| + g(|x|))$ and so $x \in L[g(n) + f(n + g(n))]$.

For the converse, suppose $x \in L[g(n) + f(n + g(n))]$. Hence there is some $\alpha \in \Sigma^*$ such that $\langle x, \alpha\rangle \in L$ and $|\alpha| = g(|x|) + f(|x| + g(|x|))$. Let y be the prefix of α consisting of the first $g(|x|)$ symbols, and z be the remaining $f(|x| + g(|x|))$ symbols. Then $\langle x, y\rangle \in L[f(n)]$ and hence $x \in (L[f(n)])[g(n)] = L[f(n), g(n)]$. This completes our proof. □

Example 1 From Lemma 5 it follows that $L[n, n] = L[3n]$. This is most easily illustrated via a figure, showing how a word in L changes as we project out the witness.



We now define time-witness classes in terms of existential projections.

Definition 6 $\mathsf{TIWI}(t(n), w(n)) = \{L[w(n)] \mid L \in \mathsf{DTIME}(t(n))\}$.

Note 1 Our definition of TIWI has similarities with the classical definition of advice classes [14]. In the advice setting, the witness values are uniquely determined by a possibly uncomputable oracle function. In contrast the witness values are not determined with our notion of existential projection, and we therefore avoid undecidable languages in our TIWI classes.

We need to be careful to account for the total nondeterminism when composing two or more nondeterministic simulations; such compositions are crucial for the proofs of our results. We have chosen to define time-witness classes via the existential projection because this notation assists in explicitly keeping track of witness size scaling in compositions. Existential projections also allow us to control the

overhead of encoding. Such detailed bookkeeping becomes necessary when the number of compositions in an argument is allowed to grow with the instance size.

For a function $w\colon \mathbb{N} \to \mathbb{N}$, let

$$\mathsf{TIWI}(t(n), O(w(n))) = \bigcup_{c>0} \mathsf{TIWI}(t(n), cw(n)).$$

As usual, polylog(n) denotes the class of functions

$$\bigcup_{c>0} \{f(n)\colon \mathbb{N} \to \mathbb{N} \mid f(n) = O((\lg n)^c)\}.$$

As a notational convenience, if $\phi(n)$ is a logical expression in which the variable n occurs free and there are no other free variables, then we say that $\phi(n)$ holds *eventually* if there exists some $n_0 \in \mathbb{N}$ such that $\phi(n)$ is a true sentence for all $n \in \mathbb{N}$ such that $n \geq n_0$.

The following lemma will simplify several proofs. This result allows us to ignore minor differences in the witness size functions when comparing two time-witness classes, and instead to focus on how they behave asymptotically.

Lemma 7 *Suppose $t\colon \mathbb{N} \to \mathbb{N}$ is a function such that $t(n) = \Omega(n)$. If $v(n) \leq w(n)$ eventually, then $\mathit{TIWI}(t(n), v(n)) \subseteq \mathit{TIWI}(t(n), w(n))$.*

Proof Suppose $K \in \mathsf{TIWI}(t(n), v(n))$. Hence there is some $L \in \mathsf{DTIME}(t(n))$ such that $K = L[v(n)]$. Let $L' = \{\langle\langle x, y\rangle, z\rangle \mid \langle x, y\rangle \in L, |z| + |y| = w(|x|)\}$. Note that L' does not include any word $\langle\langle x, y\rangle, z\rangle$ for which $v(|x|) > w(|x|)$. Since $v(n) \leq w(n)$ eventually, the language L' does include such words for all sufficiently large $|x|$, and therefore K and $L'[w(n)]$ only differ in at most finitely many words up to some threshold size n_0, and will be the same for all inputs of size n_0 and greater. Now we can decide L' by using a decider for L while ignoring any additional part of the input. The overhead of setting up the decider for L is linear, and because $t(n) = \Omega(n)$ we have that $L' \in \mathsf{DTIME}(t(n))$. Hence $L'[w(n)] \in \mathsf{TIWI}(t(n), w(n))$.

Now we can further augment the decider for $L'[w(n)]$ with a brute force simulation which deterministically checks all possible witnesses for an input $x \in K$ if $|x|$ is below the threshold size n_0 where $v(n) \leq w(n)$ for all $n \geq n_0$. This introduces a large constant factor into the simulation, but this is taken care of by the time bound being $O(t(n))$. Therefore $K \in \mathsf{TIWI}(t(n), w(n))$. □

For functions $w, t\colon \mathbb{N} \to \mathbb{N}$, we say that $w(n)$ is computable in $t(n)$ time if there is a deterministic Turing machine that when given $x \in \Sigma^*$, in at most $t(|x|)$ steps writes a word on its output tape with precisely $w(|x|)$ symbols. For simplicity, we assume that all witness size functions in the following are computable within the provided time bounds. By first computing the witness size function for the given input, a Turing machine can determine where a word finishes and the witness bits begin, thereby avoiding overhead for self-terminating encodings or separator characters in the alphabet.

$\mathsf{NTIGU}(t(n), w(n))$ is the class of languages decidable by a multitape Turing machine that takes at most $O(t(n))$ steps and uses at most $w(n)$ nondeterministic

bits on any input of length n (for instance, see [4]). The following results explain our choice of Definition 6, by relating the time-witness TIWI classes to the more familiar NTIME (Lemma 8) and time-guess NTIGU (Lemma 9) classes.

First we consider the case where witness size is quite large, even possibly dominating the input size.

Lemma 8 *If $w(n) = \Omega(n)$, then $\mathit{TIWI}(n, O(w(n))) = \mathit{NTIME}(w(n))$.*

Proof First suppose $c > 0$ and let $K \in \mathsf{TIWI}(n, c \cdot w(n))$. Then there is some $L \in \mathsf{DTIME}(n)$ such that $K = L[c \cdot w(n)]$. Let M be a deterministic Turing machine which decides L in $O(n)$ steps. We define a nondeterministic Turing machine M' that given an n-bit input x, writes a copy of x followed by a string y consisting of $c \cdot w(n)$ bits chosen nondeterministically, and simulates M with input $\langle x, y\rangle$. Then M' takes $O(n + c \cdot w(n)) + O(c \cdot w(n)) = O(w(n))$ steps to decide whether $\langle x, y\rangle \in L$, and therefore $K \in \mathsf{NTIME}(w(n))$.

Now suppose $K \in \mathsf{NTIME}(w(n))$. Then there is a nondeterministic Turing machine M' which decides K in $O(w(n))$ steps. Given an n-bit input x, we then can construct a Turing machine M which records $O(w(n))$ nondeterministic bits for the steps taken by M', and verifies in linear time in the size of the guessed sequence of actions whether M' accepts. Consider the language L consisting of strings $\langle x, y\rangle$ such that $x \in K$ and $|y| = O(w(n))$, where y records the moves made by an accepting computation of M' on input x. Then $L \in \mathsf{DTIME}(n)$ and $K = L[O(w(n))]$. Hence $K \in \mathsf{TIWI}(n, O(w(n)))$. □

If $k \geq 3$ then any k-tape Turing machine can be simulated by a two-tape machine with a logarithmic increase in time [15]. This slowdown does not affect Lemma 8 as we are not trying to reduce the number of tapes: in the proof each inclusion increases the number of tapes. This increase does not matter for the purpose of establishing the result, or for the applications where we use it, although it might be important for other applications of the technique where the number of tapes has to be more carefully controlled.

Lemma 8 showed that TIWI classes for superlinear witness bounds lose the discrimination power of the NTIGU classes. However, our next result shows that TIWI and NTIGU classes coincide for at most linear witness size and many common time bounds.

Lemma 9 *Suppose that $w(n) \leq n$ for all n, and that there exist constants $c \geq 1$ and $d \geq 0$ such that $t(n) = \Theta(n^c (\lg n)^d)$. Then $\mathit{TIWI}(t(n), w(n)) = \mathit{NTIGU}(t(n), w(n))$.*

Proof First let $K \in \mathsf{TIWI}(t(n), w(n))$. Then there exists some language $L \in \mathsf{DTIME}(t(n))$ such that $K = L[w(n)]$. Suppose that M is a deterministic Turing machine which decides L in $O(t(n))$ steps. We need to decide whether $x \in K$ using a nondeterministic machine M'. M' first copies x to a tape (which will be the input tape of M), appends $w(|x|)$ bits to this tape the values of which are determined

nondeterministically, and then simulates M. The simulation can be performed using a constant number of deterministic steps per step of M, so the total number of steps to decide whether $x \in K$ is at most $a \cdot t(|x| + w(|x|)) + b \cdot w|x|$ for some constants a, b. Moreover, M' uses $w(|x|)$ nondeterministic bits and accepts x if, and only if, there is some $y \in \Sigma^{w(n)}$ such that $\langle x, y\rangle \in L$. This is equivalent to saying that M' accepts x iff $x \in K$, so M' correctly decides K.

We now claim that if $T(n) \leq a \cdot t(n + w(n)) + b \cdot w(n)$ eventually for constants a, b, then the conditions guarantee that $T(n) = O(t(n))$. Therefore $K \in \mathsf{NTIGU}(t(n), w(n))$ by putting $T(n)$ as the largest number of steps taken by M' to decide an input of n bits. To prove the claim, suppose there are constants a, b such that $T(n) \leq a \cdot t(n + w(n)) + b \cdot w(n)$ eventually. Since $t(n) = O(n^c(\lg n)^d)$, we have some constant e so that $t(n) \leq e \cdot n^c(\lg n)^d$ eventually. Then $T(n) \leq a \cdot t(n + w(n)) + b \cdot w(n) \leq a \cdot e(n + w(n))^c(\lg(n + w(n)))^d + b \cdot w(n) \leq a \cdot e(2n)^c(1 + \lg n)^d + b \cdot n \leq a \cdot e \cdot (2n)^c(2 \lg n)^d + b \cdot n = (a2^c2^d e)n^c(\lg n)^d + b \cdot n$ eventually. As $c \geq 1$ and $d \geq 0$ it follows that $T(n) = O(n^c(\lg n)^d)$. Now since $t(n) = \Omega(n^c(\lg n)^d)$, we have that $T(n) = O(t(n))$, and we are done.

For the other direction, suppose $K \in \mathsf{NTIGU}(t(n), w(n))$. There is then some nondeterministic Turing machine M' which decides K using $O(t(n))$ steps and with $w(n)$ bits of nondeterminism. M' accepts x precisely when there is some $y \in \Sigma^{w(n)}$ representing the nondeterministic moves made by M' in reaching an accepting state (within $O(t(n))$ steps). We build a deterministic Turing machine M, which when given x and the nondeterministic moves y as input $\langle x, y\rangle$, simulates M' using a number of deterministic steps per step of M' that is bounded by some constant a. To achieve this we first compute $w(|x|)$ in at most $O(t(|x|)$ steps and store this on a unary tape, which we can then use to determine where x ends and y begins. Hence M on input $\langle x, y\rangle$ takes at most $a \cdot t(|x|) + b|y|$ steps, so $O(t(|x| + |z|))$ steps. We can now let language L be the language of words $\langle x, y\rangle$ accepted by M. There is some $y \in \Sigma^{w(n)}$ such that machine M accepts $\langle x, y\rangle$ iff M' accepts x. Then $K = L[w(n)]$ and $L \in \mathsf{DTIME}(t(n))$, so $K \in \mathsf{TIWI}(t(n), w(n))$. □

In the proof of Lemma 9 each inclusion increases the number of tapes by one. This does not affect our results but might restrict some applications.

While classes such as $\mathsf{NTIGU}(t(n), w(n))$ measure the time bound in terms of the input, $\mathsf{TIWI}(t(n), w(n))$ measures the time bound as a function of both the input and the witness. For witnesses growing strictly faster than the size of the input, the two definitions can diverge where $\mathsf{TIWI}(t(n), O(w(n)))$ is not equal to $\mathsf{NTIGU}(t(n), O(w(n)))$. To see this, take $t(n) = n$ and $w(n) = n^2$. We have that $\mathsf{NTIGU}(n, O(n^2)) = \mathsf{NTIGU}(n, O(n)) = \mathsf{NTIME}(n)$ because additional guess bits beyond $O(t(n))$ do not help us. However, $\mathsf{NTIME}(n^2) = \mathsf{TIWI}(n, O(n^2))$ by Lemma 8. Therefore, $\mathsf{NTIGU}(n, O(n^2)) = \mathsf{NTIME}(n) \subsetneq \mathsf{NTIME}(n^2) = \mathsf{TIWI}(n, O(n^2))$ by the nondeterministic time hierarchy theorem.

Notice from the preceding discussion that as the witness size grows beyond the input size, the NTIGU classes no longer capure new languages while the TIWI classes become equivalent to the coarser classical nondeterministic time classes. Even though these two definitions can diverge when the witness size is larger than the input size, Lemma 9 allows us to use TIWI rather than NTIGU in the subsequent

discussion, because in this work we are interested in witnesses of moderate size, at most as large as the input size and computable within the given time bounds. The choice of TIWI avoids technical difficulties arising from instance size blowup when composing simulations, and essentially amounts to preallocating the nondeterministic bits which are used in a computation and including them in the instance size.

7 Strong Effective Guessing Would Imply Linear-Time Simulation of P

In this section we show that a strong form of guessing would imply that all of P is contained in NTIME(n). It appears to us that this is unlikely to be true. Even though such an inclusion in turn would imply that P $\neq$ NP , many other rather less likely consequences would also follow. These include improving the current best $O(n^{2.37286})$ algorithm for multiplication of n by n matrices (see [16]) to $\tilde{O}(n^2)$ time, reducing the time for general graph maximum matching from $\tilde{O}(n^{2.5})$ (see [17]) to $\tilde{O}(n^2)$, and reducing the $n^{k/c}$ time (for some c such that $1 \leq c < k$) to decide if an input graph contains a k-clique to $O(n \lg n)$ time, all achieved through the use of nondeterminism. Yet it is not at all clear that allowing guessing could significantly speed up so many well-studied and disparate algorithms. (Here we use the common convention that $\tilde{O}(t(n))$ denotes the class of functions $\bigcup_{c>0} O(t(n)(\lg t(n))^c)$.)

Informally, our argument for Theorem 1 works as follows. We have defined the class TIWI$(n, \lg n)$, which by Lemma 5 can be regarded as the class of languages decided by nondeterministic machines that use linear time and $\lg n$ bits of nondeterminism. We suppose that DTIME$(n^2) \subseteq$ TIWI$(n, \lg n)$. By a padding argument it follows that DTIME$(n^4) \subseteq$ TIWI$(n^2, \lg n)$. Now suppose $L \in$ TIWI$(n^2, \lg n)$; this means that there is a language $L' \in$ DTIME(n^2) such that $x \in L$ if there is a y of length $\lg n$ and $xy \in L'$. Again applying our hypothesis, this time to L', we conclude via Lemmas 7 and 8 that $L \in$ TIWI$(n, \lg n)$. We therefore conclude that DTIME$(n^4) \subseteq$ TIWI$(n, \lg n)$. We can then use this step in an induction argument. We now proceed with a formal version of this argument.

In preparation for our result, we need to demonstrate that time-witness classes are structurally well-behaved. We first establish conditions which ensure that increases in witness size are kept reasonable when applying effective guessing.

Lemma 10 *If* DTIME$(t(n)) \subseteq$ TIWI$(t'(n), w(n))$ *then*

$$\text{TIWI}(t(n), w'(n)) \subseteq \text{TIWI}(t'(n), w(n + w'(n)) + w'(n)).$$

Proof Suppose that DTIME$(t(n)) \subseteq$ TIWI$(t'(n), w(n))$, and let J be an arbitrary language in TIWI$(t(n), w'(n))$. By definition then there exists some language K in DTIME$(t(n))$ such that $J = K[w'(n)]$. By our assumption there must exist some $L \in$ DTIME$(t'(n))$ such that $K = L[w(n)]$. By Lemma 5, we have that $J = K[w'(n)] = L[w(n), w'(n)] = L[w(n + w'(n)) + w'(n)]$. Finally, we can conclude that $J \in$ TIWI$(t'(n), w(n + w'(n)) + w'(n))$. □

Lemma 11 (Strong Speedup) *Suppose* $\mathsf{DTIME}(t(n)) \subseteq \mathsf{TIWI}(t'(n), w(n))$. *For all functions* $w' : \mathbb{N} \to \mathbb{N}$ *for which there exists a constant* C *such that* $w(n + w'(n)) \leq C \cdot w(n)$ *eventually, we have that*

$$\mathsf{TIWI}(t(n), w'(n)) \subseteq \mathsf{TIWI}(t'(n), C \cdot w(n) + w'(n)).$$

Proof Suppose first that the inclusion $\mathsf{DTIME}(t(n)) \subseteq \mathsf{TIWI}(t'(n), w(n))$ holds, and let K be a language in $\mathsf{TIWI}(t(n), w'(n))$. Via Lemma 10 we can now conclude that $K \in \mathsf{TIWI}(t'(n), w(n + w'(n)) + w'(n))$. From the properties of w' and Lemma 7 it then follows that $K \in \mathsf{TIWI}(t'(n), C \cdot w(n) + w'(n))$. □

We continue with a useful amplification property of time-witness classes in the presence of effective guessing. By analogy with superadditive functions (see [18]), we say that a function $f : \mathbb{N} \to \mathbb{N}$ is *weakly superadditive* if $f(n + d) \geq f(n) + d$ for all $d, n \in \mathbb{N}$. Note that any function $f(n) = n^c$, where $c \geq 1$, is weakly superadditive.

Lemma 12 *Let* f *be a weakly superadditive function. If*

$$\mathsf{DTIME}(t(n)) \subseteq \mathsf{TIWI}(t'(n), w(n))$$

then

$$\mathsf{DTIME}(t(f(n))) \subseteq \mathsf{TIWI}(t'(f(n)), w(f(n))).$$

Proof Suppose that $\mathsf{DTIME}(t(n)) \subseteq \mathsf{TIWI}(t'(n), w(n))$. Further, let K be an arbitrary language in $\mathsf{DTIME}(t(f(n)))$. Consider the function $B(n) = f(n) - n$. Let

$$\mathrm{pad}_B\text{-}K = \{\ \langle 1^k, x\rangle \mid k = B(|x|) \wedge x \in K\ \}.$$

As $K \in \mathsf{DTIME}(t(f(n)))$, we have $\mathrm{pad}_B\text{-}K \in \mathsf{DTIME}(t(n))$. By the assumption, $\mathrm{pad}_B\text{-}K \in \mathsf{TIWI}(t'(n), w(n))$. Hence there is some $L \in \mathsf{DTIME}(t'(n))$ such that $\mathrm{pad}_B - K = L[w(n)]$. Let

$$L' = \{\ \langle x, y\rangle \mid y \in \Sigma^{w(f(|x|))} \wedge \langle 1^{B(|x|)}, \langle x, y\rangle\rangle \in L\ \}.$$

Since $\langle 1^{B(|x|)}, \langle x, y\rangle\rangle = \langle\langle 1^{B(|x|)}, x\rangle, y\rangle$, it follows that $K = L'[w(f(n))]$. To show that $K \in \mathsf{TIWI}(t'(f(n)), w(f(n)))$, it then suffices to show that $L' \in \mathsf{DTIME}(t'(f(n)))$.

Because of our choice of the function B and since $L \in \mathsf{DTIME}(t'(n))$, we can determine if $\langle x, y\rangle \in L'$, where $|y| = w(f(|x|))$, in time

$$\begin{aligned} O(t'(|\langle 1^{B(|x|)}, \langle x, y\rangle\rangle|)) &= O(t'(B(|x|) + |x| + |y|)) \\ &= O(t'(f(|x|) + |y|)) \\ &\leq O(t'(f(|x| + |y|))) \\ &= O(t'(f(|\langle x, y\rangle|))). \end{aligned}$$

Therefore, $L' \in \mathsf{DTIME}(t'(f(n)))$. □

We say that a function $f(n)$ is *subpolynomial* if for every $c > 0$ we have that $f(n) = o(n^c)$, and *semihomogeneous* (see [18]) if for any $d > 1$ there is a constant $C = C(d)$ such that eventually $f(dn) \leq C \cdot f(n)$. Note that any polylogarithmic function (such as $f(n) = (\lg n)^3$) is subpolynomial and also semihomogeneous.

This leads up to our first amplification argument, showing that a form of effective guessing with small witnesses can be amplified to yield a larger speedup at the cost of only a moderate amount of additional guessing.

Lemma 13 *Let $c \geq 1$ be a constant, and let $v(n)$ be a non-decreasing function that is subpolynomial, semihomogeneous, and increases infinitely often. If $\mathsf{DTIME}(n^c) \subseteq \mathsf{TIWI}(n, v(n))$ then for all $k \in \mathbb{N}$, $\mathsf{DTIME}(n^{c^{k+1}}) \subseteq \mathsf{TIWI}(n, C^k \cdot v(n^{c^k}))$ for some constant $C \geq 1$.*

Proof Suppose that $\mathsf{DTIME}(n^c) \subseteq \mathsf{TIWI}(n, v(n))$. We will show by induction that

$$\mathsf{DTIME}(n^{c^{k+1}}) \subseteq \mathsf{TIWI}(n, C^k \cdot v(n^{c^k}))$$

for all $k \in \mathbb{N}$. The base case holds for $k = 0$ since $\mathsf{DTIME}(n^c) \subseteq \mathsf{TIWI}(n, v(n))$ is true by assumption. For the inductive step, suppose that for some $k \geq 1$ we have

$$\mathsf{DTIME}(n^{c^k}) \subseteq \mathsf{TIWI}(n, C^{k-1} \cdot v(n^{c^{k-1}})).$$

Now apply Lemma 11 to this inclusion with $w(n) = C^{k-1} \cdot v(n^{c^{k-1}})$, $w'(n) = C^{k-1} \cdot v(n^{c^k})$, $t(n) = n^{c^k}$, and $t'(n) = n$, to obtain

$$\mathsf{TIWI}(n^{c^k}, C^{k-1} \cdot v(n^{c^k})) \subseteq \mathsf{TIWI}(n, C^k \cdot v(n^{c^{k-1}}) + C^{k-1} \cdot v(n^{c^k})).$$

We can do this because the properties of v ensure that for any $\varepsilon > 0$, eventually

$$\begin{aligned} w(n + w'(n)) &= C^{k-1} \cdot v((n + w'(n))^{c^{k-1}}) \\ &\leq C^{k-1} \cdot v(((1 + \varepsilon)n)^{c^{k-1}}) \\ &\leq C^{k-1} \cdot C \cdot v(n^{c^{k-1}}) \\ &= C \cdot w(n). \end{aligned}$$

As a second step, now apply Lemma 12 to the same assumption, with $f(n) = n^{c^k}$, $w(n) = C^{k-1} \cdot v(n)$, $t(n) = n^c$, and $t'(n) = n$, to obtain

$$\mathsf{DTIME}(n^{c^{k+1}}) \subseteq \mathsf{TIWI}(n^{c^k}, C^{k-1} \cdot v(n^{c^k})).$$

This allows us to conclude that

$$\mathsf{DTIME}(n^{c^{k+1}}) \subseteq \mathsf{TIWI}(n, C^k \cdot v(n^{c^{k-1}}) + C^{k-1} \cdot v(n^{c^k})),$$

and hence

$$\mathsf{DTIME}(n^{c^{k+1}}) \subseteq \mathsf{TIWI}(n, C^k \cdot v(n^{c^k})(v(n^{c^{k-1}})/v(n^{c^k}) + 1/C)).$$

As v increases infinitely often, by Lemma 7 we then have that

$$\mathsf{DTIME}(n^{c^{k+1}}) \subseteq \mathsf{TIWI}(n, C^k \cdot v(n^{c^k})),$$

which completes our proof. □

We now wrap up our second amplification argument into a theorem.

Theorem 14 *If there exists a constant $c > 1$ and a subpolynomial function $v(n)$ such that $\mathsf{DTIME}(n^c) \subseteq \mathsf{TIWI}(n, v(n))$, then $\mathsf{P} \subsetneq \mathsf{NTIME}(n) \subsetneq \mathsf{NP}$.*

Proof Suppose that there exists $c > 1$ and a subpolynomial function $v(n)$ such that $\mathsf{DTIME}(n^c) \subseteq \mathsf{TIWI}(n, v(n))$. Since $v(n)$ is subpolynomial, so is $v(n^c)$ for any $c > 0$. By applying Lemma 13, we then have that for all $k \in \mathbb{N}$,

$$\mathsf{DTIME}(n^{c^k}) \subseteq \mathsf{TIWI}(n, w(n))$$

for some subpolynomial function $w(n)$. Since $c > 1$, $\lim_{k\to\infty} c^k = \infty$. By Lemmas 7 and 8 we then have that for all $k \in \mathbb{N}$, $\mathsf{TIWI}(n, w(n)) \subseteq \mathsf{TIWI}(n, O(n)) = \mathsf{NTIME}(n)$. It follows that $\mathsf{P} \subseteq \mathsf{NTIME}(n)$. Further, $\mathsf{P} \neq \mathsf{NTIME}(n)$ can be shown by a standard padding argument applied to the nondeterministic time hierarchy theorem [12]. □

Via Lemma 9, Theorem 1 is a corollary of Theorem 14 for the special case that $v(n)$ is a subpolynomial function that grows faster than any polylogarithmic function; $v(n) = (\lg n)^{\lg \lg n}$ is an example of such a function. A result similar to Theorem 14 was sketched in [10], with a logarithmic witness bound $v(n)$ rather than our stronger subpolynomial bound. To extract the most out of the iterated guessing technique, we have found that it is crucial (as we have done) to carefully take into account how the witness size grows as simulations are composed.

8 Effective Guessing Would Imply a SAT Breakthrough

We now show that if general computations can be significantly sped up by using nondeterministic guessing to replace part of the computation, then this would imply a breakthrough for solving SAT. More precisely, we show that using guessing to obtain an at least logarithmic factor reduction in time would imply that SAT can be decided in linear time on a nondeterministic multitape Turing machine.[1] Simple nondeterministic Turing machine algorithms for SAT use $O(n(\lg n)^c)$ time, for some constant $c \geq 1$, but this bound has resisted improvement for several decades and the at least logarithmic factor has stubbornly remained [5, 6].

A high level sketch of the argument for proving Theorem 2 is as follows. First, we show that any n-bit CNF formulas (in a reasonable encoding) can have at most $4n/\lg n$ variables. Then we establish a fairly precise time bound for sorting on deterministic multi-tape Turing machines: a list of m integers, each of size $\lg n$, can be sorted in at most $O(m(\lg m)(\lg n))$ steps. Combining these results we can show that SAT is in $\mathsf{TIWI}(n \lg n, 4n/\lg n)$. Now if $\mathsf{DTIME}(n \lg n)$ were contained in

[1] The question of whether SAT can be solved in linear time on a nondeterministic multitape Turing machine has previously been discussed within the cstheory stackexchange community [19].

$\mathsf{NTIME}(n)$, then $\mathsf{TIWI}(n \lg n, 4n/\lg n)$ would be contained in $\mathsf{NTIME}(n)$, and therefore $\mathrm{SAT} \in \mathsf{NTIME}(n)$. We proceed by proving technical results which will allow us to formalize this argument.

The following lemma shows that a nontrivial speedup of deterministic computation would also allow computations with a significant nondeterministic component to be sped up.

Lemma 15 (Weak Speedup) *If* $\mathit{DTIME}(t(n)) \subseteq \mathit{NTIME}(n)$, *then*

$$\mathit{TIWI}(t(n), n) \subseteq \mathit{NTIME}(n).$$

Proof Suppose $\mathsf{DTIME}(t(n)) \subseteq \mathsf{NTIME}(n)$ and let $J \in \mathsf{TIWI}(t(n), n)$. Then there is some $K \in \mathsf{DTIME}(t(n))$ such that $J = K[n]$. By our assumption, $K \in \mathsf{NTIME}(n)$. By Lemma 8 it follows that there is some $c > 0$ such that $K \in \mathsf{TIWI}(n, cn)$. Hence there is some $L \in \mathsf{DTIME}(n)$ such that $K = L[cn]$. We conclude via Lemma 5 that $J = K[n] = L[cn, n] = L[n{+}2cn]$, so again by Lemma 8, $J \in \mathsf{TIWI}(n, (2c{+}1)n) \subseteq \mathsf{NTIME}(n)$. □

Although Lemma 15 is closely related to Lemma 11, the weaker hypothesis of the Weak Speedup Lemma means that these results are not directly comparable.

8.1 Improved Algorithms For SAT From Effective Guessing

We now apply the Weak Speedup Lemma to show that effective guessing implies improved algorithms for SAT.

Corollary 16 *Suppose* $\mathrm{SAT} \in \mathit{TIWI}(t(n), n)$ *for some function* $t : \mathbb{N} \to \mathbb{N}$. *If further* $\mathit{DTIME}(t(n)) \subseteq \mathit{NTIME}(n)$, *then* $\mathrm{SAT} \in \mathit{NTIME}(n)$.

Proof From Lemma 15 it follows that if $\mathsf{DTIME}(t(n)) \subseteq \mathsf{NTIME}(n)$ then $\mathsf{TIWI}(t(n), n) \subseteq \mathsf{NTIME}(n)$. □

In Corollary 16, the time upper bound $t(n)$ for SAT enables the efficient guessing hypothesis to yield an improved algorithm for SAT. Classical results imply that $\mathrm{SAT} \in \mathsf{TIWI}(n(\lg n)^c, n)$ for some unspecified constant c. This is because a guess-and-check procedure can be implemented via sorting [5], and the number of variables determinines the witness size yet cannot exceed the input size. We could therefore conclude a linear time upper bound for SAT from an effective guessing hypothesis of the form $\mathsf{DTIME}(n(\lg n)^c) \subseteq \mathsf{NTIME}(n)$.

A smaller time bound for SAT permits a weaker effective guessing hypothesis. How weak can the hypothesis be made? It turns out that we can actually take $c = 1$ with some additional work. This sharper bound requires two ingredients.

The first ingredient is a reasonable encoding of SAT, which distinguishes between formulas in conjunctive normal form (CNF) which only differ by a permutation of their variable names. Reasonable encodings are used in Cook's original proof of the Cook–Levin theorem [20] and Karp's list of 21 NP-complete problems [21]. In fact,

we are not aware of any work which relies on a particular encoding of SAT yet does not use a reasonable encoding of SAT. Furthermore, the standard DIMACS CNF encoding used by SAT solvers[2] also qualifies as reasonable. We show that a reasonable encoding of SAT has the property that an n-bit CNF formula cannot represent more than $O(n/\lg n)$ different variables, eventually.

Second, we need sharp time bounds for sorting on a Turing machine. Standard mergesort algorithms are slightly wasteful when implemented on a Turing machine, so we take a closer look at Schnorr's classical approach (from [5]) to obtain a more precise time bound.

The saving in the witness size due to a reasonable encoding is offset by overhead from sorting, but combining these two ingredients allows us to conclude $c = 1$.

8.2 Bounding the Number of Variables in a CNF Formula

The following technical lemmas will be used to relate the number of variables in a SAT instance to its size.

Lemma 17 *Suppose $x_0 > 0$ and v is a real-valued function which satisfies the inequality $v(x)\lg v(x) \leq x$ for all $x \geq x_0$. Then for every $C > 1$ there is some $x_1 > 1$ such that for every $x \geq x_1$, $v(x) < Cx/\lg x$.*

Proof Instead of working with $v(x)$ such that $v(x)\lg v(x) \leq x$, let's work with an extremal function $w(x)$ such that $v(x) \leq w(x)$ and $w(x)\lg w(x) = x$ for all $x \geq x_0$. Further, put $w(x) = 2^{k(x)}$ and $x = w(x)\lg w(x) = k(x)2^{k(x)}$. Then $w(x)/(x/\lg x) = \lg x/\lg w(x) = (k(x)+\lg k(x))/k(x) = 1+(\lg k(x))/k(x)$, which tends to 1 as $k(x) \to \infty$ (which coincides with $x \to \infty$). However, this expression is strictly greater than 1 for $k(x) > 1$, i.e. for $w(x) > 2$. Eventually the fraction becomes arbitrarily close to 1, so we can say that eventually $w(x) < Cx/\lg x$ for any $C > 1$, and the result follows. □

Lemma 18 *Suppose $0 < d < 1$ and $x_0 > 0$. Further, suppose that v is a real-valued function such that $v(x) \geq 0$ and $(1-d)v(x)\lg v(x) \leq x$ for all $x \geq x_0$. Then for every $C \geq 4$ there is some $x_1 > 1$ such that for every $x \geq x_1$, $v(x) < Cx/\lg x$.*

Proof Given $C \geq 4$, let d be the smaller of the two solutions of the equation $d(1-d) = 1/C$. Since $0 < 1/C \leq 1/4$, we have that $0 < d \leq 1/2$, and it follows that $\lg(1-d) < 0$. Let $w(x) = (1-d)v(x)$. Then for all $x \geq x_0$, $w(x)\lg w(x) = (1-d)v(x)(\lg(1-d)+\lg v(x)) < (1-d)v(x)\lg v(x) \leq x$. Since $1/d \geq 2 > 1$, by Lemma 17 eventually $w(x) < (1/d)x/\lg x$. Therefore eventually $v(x) < x/(d(1-d)\lg x) = Cx/\lg x$. □

We now assert that an encoding of SAT which removes all symmetries due to variable names does not constitute a reasonable encoding. An unreasonable encoding

[2]See http://archive.dimacs.rutgers.edu/pub/challenge/satisfiability/doc/satformat.tex.

could represent a CNF formula by a binary encoded integer which represents one particular CNF formula out of a predetermined list of equivalence classes of CNF formulas, with formulas regarded as equivalent up to reordering and renaming of variables. We instead consider only reasonable encodings, which have the property that if two CNF formulas can be obtained from each other by simply permuting variable names, then these formulas will be represented by different words in the language. With this restriction on what constitutes a reasonable encoding of SAT, we now prove an upper bound on how many variables can appear in a SAT instance in terms of its size.

Lemma 19 *In any reasonable encoding, n-bit CNF formulas eventually contain at most $4n/\lg n$ distinct variables.*

Proof Suppose x is an n-bit input. We are only interested in inputs that are valid CNF formulas, so further suppose that x represents a propositional formula in CNF, and that this formula uses v distinct variables. We will show that $v \leq 4n/\lg n$ eventually.

Let s_x be the v-element sequence formed by listing the first occurrence of each variable in the formula encoded by x. (Note that v depends on x.) Any reasonable representation must be able to distinguish each of the $v!$ possible ways that s_x can occur, one for each permutation of the variables. Hence at least $\lg(v!)$ bits are required in the worst case, for any reasonable encoding of SAT. By the Robbins bounds [22]

$$\lg v! = v \lg v - v \lg e + (1/2) \lg(2\pi) + (1/2) \lg v + r_v$$

where $(1/(12v+1)) \lg e < r_v < (1/12v) \lg e$, and so for $n = |x|$ we have

$$v \lg v - v \lg e + (1/2) \lg(2\pi) + (1/2) \lg v + (1/(12v+1)) \lg e < n.$$

Hence for any $0 < d < 1$, eventually $(1-d)v \lg v < n$. By Lemma 18 there is then some $n_1 \geq n_0 + 1$ such that for all $n \geq n_1$, $v < 4n/\lg n$. □

8.3 A More Precise Sorting Time Bound

Results about sorting on a Turing machine are used in many classical papers. However, as far as we are aware, a time bound has not been expressed in the literature in the precise form that we will present here. We do not claim originality for such a result, but also have not been able to locate a proof with this bound. We therefore provide a proof for completeness.

Lemma 20 *A deterministic multitape Turing machine can sort a list of m non-negative integers, each represented in binary encoding using $\lg n$ bits, in $O(m(\lg m)(\lg n))$ steps.*

Proof We use a form of bottom-up mergesort. Instead of a random access algorithm such as that of Batcher [23], we use a procedure that uses a fixed number

of tapes and only sequential access, and can therefore be efficiently implemented on a deterministic multitape Turing machine. This algorithm is a more detailed version of that outlined by Schnorr [5, Program p_1]. These additional details allow a more precise analysis of the time bound, which Schnorr was not attempting to optimise.

The algorithm proceeds in stages. At each stage we use three tapes containing permutations of the list of m integers: Result, Source, and Target. During each stage, Source and Target are piecewise merged to form Result. Result then becomes the Source for the next stage, and is copied to Target to begin the next stage. Half the elements to be merged in each stage are on Source and the other half on Target: the actual contents of Source and Target are identical but we pay attention to a different pattern of sequences on Source compared to Target. We use two copies of the list (one on Source and one on Target), rather than a single source tape, to avoid back-and-forth tape head moves. This is key to keeping the runtime under control.

After $\lceil \lg m \rceil \leq 2 \lg m$ stages the current Result tape contains a sorted list. Moreover, each stage uses $O(m \lg n)$ steps. This is because we can use a small fixed number of tapes to keep track of various unary quantities, and two tapes as temporary workspace to copy the integers on Source and Target that are the current focus of attention. This allows the machine to move the heads on Result, Source, and Target tapes only in one direction, with no backward motion required. Backward motion is only used when the heads are repositioned to the start of each tape, at the end of each stage. Moreover, the head movements on the auxiliary tapes only require a constant factor overhead. The overall time bound then follows.

We first pad the input with dummy values that represent a number larger than the largest integer represented using $\lg n$ bits, so that the number of values in the list is a power of 2 (and, in particular, $\lg m$ is a non-negative integer). The overhead of this padding stage is included in the unspecified constant factor in the overall time. (Moreover this also only increases the space used by at most a factor of 2.) We now outline the key steps for the case where m is a power of 2.

For each $i = 0, 1, \ldots, (\lg m) - 1$, if Source and Target at the start of stage i contain $m/2^i$ sequences of sorted sublists, each sublist of length 2^i, then at the end of the stage Result will contain $m/2^{i+1}$ sorted subsequences, each containing 2^{i+1} elements. At stage i, the Source tape head is at the start of the tape, and we move the Target tape head to the position after the 2^ith entry in the list (position $2^i \lg n$ if the first position on the tape is numbered 0). Once the first sublist has been processed, we move the heads forward by $2^i \lg n$ positions. We proceed until the Target tape head reaches the position after the end of the whole list, position $m \lg n$.

To process a single pair of sublists S and T (on the Source and Target tapes, respectively), we first set up a unary counter using an auxiliary tape to keep track of the length of these sublists, then scan the elements sequentially and write the sublist formed by merging S and T to the Result tape. At each step we are deciding which of a pair of elements to write to the result tape. We write the smaller of the two current elements to the Result tape. We do this by copying the current elements to two auxiliary tapes, and during copying flagging which tape contains the smaller of the two

elements. The auxiliary tapes are then rewound, and the flagged tape is copied to the Result tape. □

Schnorr proved a time bound of $O(m(\lg m)^c(\lg n))$ steps for some unspecified c [5]. By a more detailed analysis of the tape head motion than was considered in Schnorr's argument we have obtained this more precise exponent for the logarithmic factor of $c = 1$.

8.4 Improving the Time-Witness Bound For SAT

We are now able to prove a time-witness upper bound on SAT.

Lemma 21 SAT $\in$ *TIWI*$(n \lg n, 4n/\lg n)$.

Proof By Lemma 19, SAT can be decided nondeterministically by guessing an assignment to the eventually at most $4n/\lg n$ variables, and then verifying that the assignment satisfies the input formula. The verification is deterministic, and can be done by first making a copy of the input formula while annotating every literal with a clause number, then sorting the literals by variable identifier, replacing each literal by its value in the guessed assignment, and finally sorting the values by clause number and scanning to check that at least one literal in each clause is set to true. This procedure is a special case of the more general algorithm suggested by Schnorr [5, Program p_3]. The augmented formula is at most twice as long as the original, and by Lemma 20 it can be sorted in at most $O((n/\lg n)(\lg n)^2)$ steps, which is $O(n \lg n)$ steps. It follows that SAT $\in$ TIWI$(n \lg n, 4n/\lg n)$. □

We restate this in terms of the NTIGU notation.

Theorem 2 SAT $\in$ *NTIGU*$(n \log n, O(n/\log n))$.

Proof Follows immediately from Lemmas 21 and 9. □

Our time-witness bound for SAT then yields the main result of this section.

Theorem 3 *If* *DTIME*$(n \log n) \subseteq$ *NTIME*(n), *then* SAT $\in$ *NTIME*(n).

Proof By Lemmas 21 and 7, we conclude that SAT $\in$ TIWI$(n \lg n, 4n/\lg n) \subseteq$ TIWI$(n \lg n, n)$. Now we can apply Corollary 16, and so if DTIME$(n \lg n) \subseteq$ NTIME(n) then TIWI$(n \lg n, n) \subseteq$ NTIME(n). □

9 Conclusion and Further Work

Our contributions in this work demonstrate that effective guessing has unlikely consequences. We therefore propose an *ineffective guessing conjecture*, that it is not

in general possible to speed up a computation significantly by using more nondeterminism.

More precisely, we propose the following ineffective guessing conjecture:

Conjecture 22 (IGC) $\mathit{DTIME}(t(n)) \not\subseteq \mathit{NTIME}(n)$ *for all time-constructible functions* $t(n)$ *such that* $t(n) = \omega(n \lg n)$.

According to this ineffective guessing conjecture, it is not in general possible to obtain even a slightly greater than logarithmic speedup by making essentially every step of a computation nondeterministic. Furthermore, our ineffective guessing conjecture implies that the effective guessing hypothesis from Theorem 1, with a polynomial speedup, is too strong while the weaker effective guessing hypothesis from Theorem 2 could still hold.

To put Theorems 14 and 2 into context, the effective guessing hypotheses used in these theorems fall between two extremes.

$$\begin{aligned}
\textbf{excessively-weak EGH} &: \ \mathsf{DTIME}(n) \subseteq \mathsf{NTIGU}(n, 0) \\
\textbf{weak EGH} &: \ \mathsf{DTIME}(n \lg n) \subseteq \mathsf{NTIME}(n) \\
\textbf{strong EGH} &: \ (\exists c > 1)\ (\forall d > 0)\ \mathsf{DTIME}(n^c) \subseteq \mathsf{NTIGU}(n, n^d) \\
\textbf{excessively-strong EGH} &: \ (\exists c > 0)\ \mathsf{DTIME}(n^{2+c}) \subseteq \mathsf{NTIGU}(n, \lg n)
\end{aligned}$$

The hypothesis from Theorem 14 is the strong EGH, while Theorem 2 posits the weak EGH. To be clear, both of these hypotheses currently remain open, although we have shown that they have somewhat unlikely consequences.

Excessively weak forms of effective guessing are always true such as $\mathsf{DTIME}(n) \subseteq \mathsf{NTIGU}(n, w(n))$ which holds for any function $w(n)$, even $w(n) = 0$. This therefore forms one extreme, a hypothesis about effective guessing that is too weak to be interesting. On the other hand, we show in the following that excessively strong forms of effective guessing (such as that stated above) can be ruled out unconditionally.

Lemma 23 *If* $t(n) = \omega(n^2)$ *is a function that is computable in* $t(n)$ *steps, then*

$$\mathit{DTIME}(t(n)) \not\subseteq \mathit{NTIGU}(n, \lg n).$$

Proof Suppose $t(n) = \omega(n^2)$ such that $t(n)$ is computable in $t(n)$ steps and $\mathsf{DTIME}(t(n)) \subseteq \mathsf{NTIGU}(n, \lg n)$. By trying all $2^{\lg n} = n$ possible values for the witness and checking each in $O(n)$ time we have $\mathsf{NTIGU}(n, \lg n) \subseteq \mathsf{DTIME}(n^2)$. Thus $\mathsf{DTIME}(t(n)) \subseteq \mathsf{DTIME}(n^2)$. As $t(n) = \omega(n^2)$ this then contradicts the deterministic time hierarchy theorem [3]. □

Proposition 24 $\mathit{DTIME}(n^{2+c}) \not\subseteq \mathit{NTIGU}(n, \lg n)$ *for all* $c > 0$.

Proof The result follows from Lemma 23 for $t(n) = n^{2+c}$. □

Since the strong EGH trivially implies the weak EGH, we can therefore rank the hypotheses in terms of logical strength as follows:

$$\begin{aligned} &\textbf{excessively-strong EGH} \quad \textbf{[false]} \\ &\Downarrow \\ &\textbf{strong EGH} \quad \textbf{[open]} \\ &\Downarrow \\ &\textbf{weak EGH} \quad \textbf{[open]} \\ &\Downarrow \\ &\textbf{excessively-weak EGH} \quad \textbf{[true]} \end{aligned}$$

Finally, our effective guessing hypotheses focus on nondeterministic linear time because the Paul et al. [11] result that $\mathsf{DTIME}(n) \subsetneq \mathsf{NTIME}(n)$ invites many questions about the potential computational power of nondeterministic linear time. A natural future direction would be to consider the computational power of $\mathsf{NTIME}(n^k)$ for $k > 1$. In particular, it is still is not known whether $\mathsf{DTIME}(n^k) \subsetneq \mathsf{NTIME}(n^k)$ for any $k > 1$. Furthermore, there are many additional open questions such as whether any languages exist in $\mathsf{NTIME}(n^k) \setminus \mathsf{NTIGU}(n^k, o(n^k))$.

Acknowledgements The authors thank Ryan Williams, Rahul Santhanam, and Kenneth Regan for useful discussions. We also acknowledge the helpful discussion and comments from [19] which have helped us to provide a detailed treatment of Theorem 3.

Funding The first author's work was supported by EPSRC grant EP/P015638/1.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Homer, S., Selman, A.L. Computability and Complexity Theory, 2nd edn. Springer, New York (2011)
2. Hartmanis, J., Stearns, R.E.: On the computational complexity of algorithms. Transactions of the AMS **117**, 285–306 (1965). https://doi.org/10.1090/S0002-9947-1965-0170805-7
3. Fürer, M.: The tight deterministic time hierarchy. In: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing. STOC 1982, pp. 8–16. Association for Computing Machinery (1982). https://doi.org/10.1145/800070.802172

4. Fortnow, L., Santhanam, R.: New Non-Uniform lower bounds for uniform classes. In: Raz, R. (ed.) CCC 2016: 31st Conference on Computational Complexity. Leibniz International Proceedings in Informatics (LIPIcs), vol. 50, pp. 19–11914. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2016). https://doi.org/10.4230/LIPIcs.CCC.2016.19
5. Schnorr, C.P.: Satisfiability is quasilinear complete in NQL. J. ACM **25**(1), 136–145 (1978). https://doi.org/10.1145/322047.322060
6. Santhanam, R.: Lower bounds on the complexity of recognizing SAT by Turing machines. Inf. Process. Lett. **79**(5), 243–247 (2001). https://doi.org/10.1016/S0020-0190(00)00227-1
7. Fortnow, L., Santhanam, R., Trevisan, L.: Hierarchies for semantic classes. In: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing. STOC 2005, pp. 348–355. Association for Computing Machinery (2005). https://doi.org/10.1145/1060590.1060642
8. Impagliazzo, R., Wigderson, A.: P = BPP if E requires exponential circuits: derandomizing the XOR lemma. In: Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing. STOC 1997, pp. 220–229. Association for Computing Machinery (1997). https://doi.org/10.1145/258533.258590
9. Williams, R.: Non-uniform ACC circuit lower bounds. J. ACM **61**(1), 2–1232 (2014). https://doi.org/10.1145/2559903
10. Bloch, S.A., Buss, J.F., Goldsmith, J.: How hard are n^2-hard problems? SIGACT News **25**(2), 83–85 (1994). https://doi.org/10.1145/181462.181465
11. Paul, W.J., Pippenger, N., Szemerédi, E., Trotter, W.T.: On Determinism versus Non-Determinism and Related Problems. In: 24Th Annual Symposium on Foundations of Computer Science. FOCS 1983, pp. 429–438 (1983). https://doi.org/10.1109/SFCS.1983.39
12. Žák, S.: A Turing machine time hierarchy. Theor. Comput. Sci. **26**(3), 327–333 (1983). https://doi.org/10.1016/0304-3975(83)90015-4
13. Santhanam, R.: On separators, segregators and time versus space. In: Proceedings of the Sixteenth Annual Conference on Computational Complexity. CCC 2001, pp. 286–294 (2001). https://doi.org/10.1109/CCC.2001.933895
14. Karp, R.M., Lipton, R.J.: Turing machines that take advice. L'Enseignement Mathématique (second series) **28**, 191–209 (1982). https://doi.org/10.5169/seals-52237
15. Hennie, F.C., Stearns, R.E.: Two-tape simulation of multitape Turing machines. J. ACM **13**, 533–546 (1966). https://doi.org/10.1145/321356.321362
16. Alman, J., Williams, V.V.: A refined laser method and faster matrix multiplication. In: Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 522–539 (2021). https://doi.org/10.1137/1.9781611976465.32
17. Vazirani, V.V.: A Simplification of the MV Matching Algorithm and its Proof. arXiv: 1210.4594 (2013)
18. Book, R.V., Greibach, S.A., Wegbreit, B.: Time- and tape-bounded Turing acceptors and AFLs. J. Comput. Syst. Sci. **4**(6), 606–621 (1970). https://doi.org/10.1016/S0022-0000(70)80031-9
19. Wehar, M.: Is there a non-deterministic linear time algorithm for CNF-SAT? StackExchange. Accessed 29 Nov 2022. https://cstheory.stackexchange.com/questions/32873 (2022)
20. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the Third Annual ACM Symposium on Theory of Computing. STOC 1971, pp. 151–158. Association for Computing Machinery (1971). https://doi.org/10.1145/800157.805047
21. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Proceedings of a Symposium on the Complexity of Computer Communications. The IBM Research Symposia Series, pp. 85–103. Plenum Press (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
22. Robbins, H.: A remark on Stirling's formula. Am. Math. Mon. **62**(1), 26–29 (1955). https://doi.org/10.2307/2308012
23. Batcher, K.E.: Sorting networks and their applications. In: Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference. AFIPS 1968 (Spring), pp. 307–314. Association for Computing Machinery (1968). https://doi.org/10.1145/1468075.1468121