A note on time hierarchies for reasonable semantic classes without advice

Hao Wu*

December 1, 2017

Abstract

We show time hierarchies for reasonable semantic classes without advice by eliminating the constant bits of advice in previous results. The elimination is done by a contrapositive argument that for any reasonable computational model, let CTIME($f(n)$)/$g(n)$ denote the set of all languages decide by machines running in time $O(f(n))$ with advice of $g(n)$ bits in that model, if CTIME($t(n)$) $\subseteq$ CTIME($T(n))$/$A(n)$ then CTIME($t(n)$)/$a$ $\subseteq$ CTIME($T(n))$/$a + 2^a A(n)$ where $a$ is a constant integer.

Keywords: time hierarchy, semantic class, advice

1 Introduction

Time hierarchy theorems and other hierarchy theorems tell us, given more certain resource, we can strictly decide more problems. Unlike problems to compare the computational power between machines with different resources, such as P vs. NP problem, time hierarchy theorems are one of the successful stories of complexity theory serving as a base for many other important results such as recent breakthrough of ACC circuit lower bound for NEXP.

The first time hierarchy theorem was the time hierarchy for Turing machine\cite{7} which literally founded the field of computational complexity. This early day result followed by other hierarchy theorem such as space hierarchy\cite{11}, nondeterministic time hierarchy\cite{2,10,12}, etc. lead us toward the goal to establish hierarchies for certain source of any reasonable model of computation.

The missing parts of this big picture are the hierarchy theorems for (full uniform) BPTIME and other so called semantic models which can not be done with direct simulation-then-diagonalization argument, as oppose to those syntactic ones.

Take the case of hierarchy theorem of BPTIME for example\cite{1}, the first early attempt was using the transition method. With this method, Karpinski and Verbeek \cite{8} showed a rather
weak time hierarchy for probabilistic classes although with full uniformity. In a breakthrough, Barak \[1\] established a time hierarchy for BPP with slight non-uniformity with the idea of instance checker and optimal algorithm. Based on Barak’s method, Fortnow and Santhanam \[2\]. Goldreich et al. \[3\] managed to reduce the advice used by the more powerful class to only one bit.

Subsequently, by extending previous arguments, Fortnow et al. \[5\] showed hierarchies with small advice for a general class of reasonable semantic models. Finally, Melkebeek and Pervyshev \[6\] improved their general results with different techniques and showed that any reasonable semantic model has a hierarchy in the polynomial-time range with one bit of advice.

Now the situation is that we have hierarchies like $\text{CTIME}(n^d)/1 \not\subseteq \text{CTIME}(n^c)/1, 1 \leq c < d$ and even stronger but less tight one such as $\cup_{d \in \mathbb{N}} \text{CTIME}(n^d)/1 \not\subseteq \text{CTIME}(n^c)/O(\log n).$ We need to eliminating the last constant bits of advice in the more powerful class to achieved the full uniform one such as: $\text{CTIME}(n^d) \not\subseteq \text{CTIME}(n^d)$ or stronger one as $\cup_{d \in \mathbb{N}} \text{CTIME}(n^d)/1 \not\subseteq \text{CTIME}(n^c)/O(\log n).$ We call these kinds of time hierarchies as time hierarchies without advice, in which the more powerful class uses no advice while the less powerful class may use some advice. And to achieve this end, it is sufficient to prove the contrapositive, for example,

$$\text{CTIME}(n^d) \subseteq \text{CTIME}(n^c) \implies \text{CTIME}(n^d)/1 \subseteq \text{CTIME}(n^c)/1.$$ 

**Our Results:**

**Lemma 1.1.** For any reasonable model, $\text{CTIME}(t(n))/a \subseteq \text{CTIME}(T(n))/A(n)$ implies $\text{CTIME}(t(n))/a \subseteq \text{CTIME}(T(n))/a + 2^a A(n)$ where $a$ is a constant.

For a quick illustration of the idea, we show how to achieve this for BPP in a simpler case, that is

$$\text{BPTIME}(n^d) \subseteq \text{BPTIME}(n^c)/1 \implies \text{BPTIME}(n^d)/1 \subseteq \text{BPTIME}(n^c)/3, 1 \leq c < d.$$ 

For any language $L \in \text{BPTIME}(n^d)/1,$ there is a machine $M \in \text{BPTIME}(n^d)/1$ which uses only one bit advice $b,$ now fix the advice $b$ to 0 or 1 for all input length, and this results in two full uniform machines $M_0, M_1$ since for every input length $M_0$ works with the fix advice $b = 0$ and $M_1$ works with the fix advice $b = 1$. Note that for certain input length, the original machine $M$ with advice $b$ behaves exactly the same $M_b.$ And fixing the advice costs $M_0, M_1$ only constant overhead of time and machine size, so $M_0, M_1$ belong to $\text{BPTIME}(n^d)$ as well.

Via the hypothesis $\text{BPTIME}(n^d) \subseteq \text{BPTIME}(n^c)/1,$ there are two machines: $M_0'$ with one bit advice $a_0$ and $M_1'$ with one bit advice $a_1$ which both run in $O(n^c)$ time and $L(M_0) = L(M_0'), L(M_1) = L(M_1').$

Now we can construct a new machine $M'$ by combining these two machines $M_0', M_1'$ using three bits advice $b, a_0, a_1$ as follows:

When give a input $x, M'$ calls $M_b'$ with advice $a_b$ and outputs whatever $M_b'$ outputs.

It is not hard to see that $M'$ recognizes the same language as $M.$ And the overhead of $M$’s running time over $M_b'$ is calling action which takes constant time in reasonable models, take probabilistic Turing machine with advice tape for example, the calling action is setting the advice tape to $a_b$ and transiting to control to $M_b'.$ So $M'$ also runs in $O(n^c)$ and this completes the argument.
This already allows us to achieve the time hierarchy for BPP without advice and in next section we take this argument to full generality and use it to obtain time hierarchies for reasonable semantic classes without advice.

2 Time hierarchies for reasonable semantic classes without advice

At first, we prove a general lemma eliminating constant bits of advice in time hierarchies for any reasonable computational model.

For any reasonable computational model which may use advice, let $\text{CTIME}(f(n))/g(n)$ denote the set of all languages decide by machines running in time $O(f(n))$ with advice of $g(n)$ bits in that model, we want to prove

$$\text{CTIME}(t(n)) \subseteq \text{CTIME}(T(n))/A(n) \text{ implies } \text{CTIME}(t(n))/a \subseteq \text{CTIME}(T(n))/a + 2^a A(n)$$

where $a$ is a constant. We need to show how to decide any language $L \in \text{CTIME}(t(n))/a$ with a machine $M' \in \text{CTIME}(T(n))/a + 2^a A(n)$ with the assumption $\text{CTIME}(t(n)) \subseteq \text{CTIME}(T(n))/A(n)$.

Since $L \in \text{CTIME}(t(n))/a$, then there a machine $M \in \text{CTIME}(t(n))/a$ can decide $L$. Now we fix the advice for $M$ to some string $s \in \{0,1\}^a$, this new machine $M_s$ will use advice $s$ for all input length, and this results in $2^a$ machines without advice: $M_0, \ldots, M_{2^a-1}$.

Note that for certain input length $n$, the original machine $M$ with advice $s$ for input length $n$ behaves exactly the same $M_s$ for every input $x$ from $\{0,1\}^n$. And since $M \in \text{CTIME}(t(n))/a$, all $M_0, \ldots, M_{2^a-1}$ belong to $\text{CTIME}(t(n))$ as well.

With the hypothesis $\text{CTIME}(t(n)) \subseteq \text{CTIME}(T(n))/A(n)$, there are $2^a$ corresponding machines $M'_0, \ldots, M'_{2^a-1}$ which belong to $\text{CTIME}(T(n))/A(n)$ and $L(M_i) = L(M'_i)$ for every $i \in \{0, \ldots, 2^a-1\}$. Now we can construct a machine $M'$ by combining all $M'_i$'s as follows:

**Construction 2.1.** $M'$ takes $a + 2^a A(n)$ bits advice. The first $a$ bits is used to choose the advice used by $M'_0, \ldots, M'_{2^a-1}$. The rest of advice is divided into $2^a$ blocks, the $i$th block stores the $A(n)$ bits of advice for $M'_i$.

The description of $M'$ is the combination of all $M'_i$'s and $M'$ works as follows:

When given an input $x$, $M'$ uses the first $a$ bits string $s$ of its advice to choose a machine $M'_i$ where $i$ is the value of $s$ when treated as the base 2 number, then feeds the $i$th block of the rest advice as the advice for $M'_i$ and transits control to $M'_i$ and outputs whatever $M'_i$ outputs.

Since $M'$ mimics $M$'s behavior by calling the correct $M'_i$ with proper advice, $M'$ recognizes the same language as $M$. We only need the new machine $M'$ to be of the same computational model and the overhead of $M$’s running time over any $M'_i$, the calling action, takes constant time in that model, thus we turn this demand into following definition.

**Definition 2.2.** A model of computation is called compatible with construction 2.1 if the result of construction 2.1, $M'$ is of the same computational model and the overhead of $M$’s running time over any $M'_i$ only takes constant time.

To summarise, we have following lemma:
Lemma 2.3. [Elimination lemma for constant advice] For any computational model which is compatible with construction 2.1:

\[ \text{CTIME}(t(n)) \subseteq \text{CTIME}(T(n))/A(n) \text{ implies } \text{CTIME}(t(n))/a \subseteq \text{CTIME}(T(n))/a + 2^a A(n) \]

where \( a \) is a constant. Via contraposition, we have

\[ \text{CTIME}(t(n))/a \not\subseteq \text{CTIME}(T(n))/a + 2^a A(n) \text{ implies } \text{CTIME}(t(n)) \not\subseteq \text{CTIME}(T(n))/A(n) \]

It is crucial that the size of advice used by the more powerful class is constant, fixing the advice to some string is independent of input length thus after the fixing we have a uniform machine, otherwise the argument will not work.

We can check this argument works for many reasonable semantic classes which is compatible with construction 2.1. It is not hard to see it works for models like randomized machines, quantum machines, nondeterministic machines, unambiguous machines, Arthur-Merlin games with time-bounded Arthur, etc.

For symmetric alternation class, which is the intersection of two reasonable semantic classes, we observe following simple fact:

**Fact 2.4.** If two complexity classes are fit in our lemma, their intersection is also fit in our lemma.

**Proof.** For two complexity classes \( C, C' \), let \( C \cap C'-\text{TIME}(f(n)/g(n)) \) denote

\[ \text{CTIME}(f(n)/g(n)) \cap C'-\text{TIME}(f(n)/g(n)). \]

For any \( L \in C \cap C'-\text{TIME}(t(n))/a \), and if \( C \cap C'-\text{TIME}(t(n)) \subseteq C \cap C'-\text{TIME}(T(n))/A(n) \), by applying the lemma twice, \( L \in \text{CTIME}(T(n))/a + 2^a A(n) \) and \( L \in C'-\text{TIME}(T(n))/a + 2^a A(n) \)

Finally, to apply our lemma, we need the time hierarchies with advice to satisfy following:

- the class with more running time of hierarchy uses only \( a \) bits of advice and \( a \) is a constant.
- the class with less running time is allowed to use no less than \( a \) bits of advice.

We note the following result due to Melkebeek and Pervyshev [9] is the most general time hierarchy for semantic classes with constant bits of advice.

**Theorem 2.5.** [General time hierarchies for reasonable semantic classes with constant bits of advice [2]] For any reasonable semantic model of computation, any constant \( \delta > 1 \), and any monotone constructible time bound \( t(n) \) satisfying

\[ t(2^{n^{O(1)}}) \geq 2^{t(n)^{O(1)}} \text{ and } t(n^{O(1)}) \geq (t(n^{O(1)}))^{O(1)} \tag{2.1} \]

we have \( \text{CTIME}(t(n^\delta))/a \not\subseteq \text{CTIME}(t(n))/a \) where \( a \) is a constant integer.

To call a semantic model of computation reasonable, which was defined in [9], the model has to satisfy follow properties:

---

\(^2\)See more details in their paper [9].
it has an efficient universal machine that can be complemented deterministically in exponential time
it is efficiently closed under deterministic transducers.

And to complement their definition of reasonable model, we demand that a reasonable model should also be compatible with construction 2.1 thus fit in our lemma.

With theorem 2.5 and lemma 2.3 with \( A(n) = 0 \) we have:

**Theorem 2.6** (Time hierarchies for reasonable semantic classes without advice). For any reasonable semantic model of computation, any constant \( \delta > 1 \), and any monotone constructible time bound \( t(n) \) satisfying (2.1), we have \( \text{CTIME}(t(n^\delta)) \not\subseteq \text{CTIME}(t(n)) \).

There are also time hierarchies with one bit advice for BPP, RP that is stronger in the sense they allow the class with less running time using larger size of advice, although these hierarchies are not as tight as above one.

**Theorem 2.7.** [5, 9] For any constants \( a \) and \( c \), \( \text{BPP}/1 \not\subseteq \text{BPTIME}(n^c)/a \log n \)

**Theorem 2.8.** [5, 9] For any constants \( a \) and \( c \), \( \text{RP}/1 \not\subseteq \text{RTIME}(n^c)/a(\log n)^{1/c} \)

With our lemma and above two theorems, we also have stronger but less tight time hierarchies for BPP, RP without advice:

**Theorem 2.9** (Stronger but less tight Time hierarchies for BPP and RP without advice). For any constants \( a \) and \( c \):

- \( \text{BPP} \not\subseteq \text{BPTIME}(n^c)/a \log n \)
- \( \text{RP} \not\subseteq \text{RTIME}(n^c)/a(\log n)^{1/c} \)

### 3 Conclusions

In this paper, we use a simple contrapositive argument to eliminate constant size advice by fixing the advice and this results in completely uniform time hierarchies for many reasonable semantic models.

**Acknowledgement**

This work is supported by National Natural Science Foundation of China (NSFC) 11601322.

**References**

[1] B. Barak. A probabilistic-time hierarchy theorem for ”slightly non-uniform” algorithms. In Proceedings of the 6th International Workshop on Randomization and Approximation Techniques, RANDOM ’02, pages 194–208, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44147-6. URL [http://dl.acm.org/citation.cfm?id=646978.711702](http://dl.acm.org/citation.cfm?id=646978.711702)
[2] S. A. Cook. A hierarchy for nondeterministic time complexity. *J. Comput. Syst. Sci.*, 7(4):343–353, Aug. 1973. ISSN 0022-0000. doi: 10.1016/S0022-0000(73)80028-5. URL http://dx.doi.org/10.1016/S0022-0000(73)80028-5

[3] L. Fortnow and R. Santhanam. Hierarchy theorems for probabilistic polynomial time. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, FOCS ’04*, pages 316–324, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2228-9. doi: 10.1109/FOCS.2004.33. URL http://dx.doi.org/10.1109/FOCS.2004.33

[4] L. Fortnow and R. Santhanam. Time hierarchies: A survey. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(004), 2007. URL http://eccc.hpi-web.de/eccc-reports/2007/TR07-004/index.html

[5] L. Fortnow, R. Santhanam, and L. Trevisan. Hierarchies for semantic classes. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC ’05, pages 348–355, New York, NY, USA, 2005. ACM. ISBN 1-58113-960-8. doi: 10.1145/1060590.1060642. URL http://doi.acm.org/10.1145/1060590.1060642

[6] O. Goldreich, M. Sudan, and L. Trevisan. Studies in complexity and cryptography. chapter From Logarithmic Advice to Single-bit Advice, pages 109–113. Springer-Verlag, Berlin, Heidelberg, 2011. ISBN 978-3-642-22669-4. URL http://dl.acm.org/citation.cfm?id=2028116.2028130

[7] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[8] M. Karpinski and R. Verbeek. On the monte carlo space constructible functions and separation results for probabilistic complexity classes. *Inf. Comput.*, 75(2): 178–189, Nov. 1987. ISSN 0890-5401. doi: 10.1016/0890-5401(87)90057-5. URL http://dx.doi.org/10.1016/0890-5401(87)90057-5

[9] D. Melkebeek and K. Pervyshev. A generic time hierarchy with one bit of advice. *Comput. Complex.*, 16(2):139–179, May 2007. ISSN 1016-3328. doi: 10.1007/s00037-007-0227-8. URL http://dx.doi.org/10.1007/s00037-007-0227-8

[10] J. I. Seiferas, M. J. Fischer, and A. R. Meyer. Separating nondeterministic time complexity classes. *J. ACM*, 25(1):146–167, Jan. 1978. ISSN 0004-5411. doi: 10.1145/322047.322061. URL http://doi.acm.org/10.1145/322047.322061

[11] R. E. Stearns, J. Hartmanis, and P. M. Lewis. Hierarchies of memory limited computations. In *Proceedings of the 6th Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1965)*, FOCS ’65, pages 179–190, Washington, DC, USA, 1965. IEEE Computer Society. doi: 10.1109/FOCS.1965.11. URL https://doi.org/10.1109/FOCS.1965.11

[12] S. Žák. A turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327 – 333, 1983. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(83)90015-4. URL http://www.sciencedirect.com/science/article/pii/0304397583900154