Tighter Constant-Factor Time Hierarchies

Amir M. Ben-Amram*
The Academic College of Tel-Aviv Yaffo
Tel-Aviv, Israel
amirben@server.mta.ac.il

May 6, 2002

Abstract

For certain computational models, a constant-factor time hierarchy theorem is known, showing that a constant-factor difference in time bounds makes a difference in problem-solving power (unlike the situation with Turing machines). In this note we apply the classic "translational technique" (padding) to these hierarchies and show that they are tighter than indicated by the previous proofs.

Keywords: computational complexity, hierarchy theorem, padding argument.

1 Background

Many textbooks in Theoretical Computer Science include the *Turing machine speedup theorem*, stating that any desired constant factor can (almost) always be saved in the running time of a Turing machine. Hence constant factors do not matter in time complexity. In [4], Neil Jones claimed that in real-world programming, constant factors *do* matter, and hence a theorem which demonstrates this situation in a formal model may be more relevant than the Turing machine result. Such a theorem, which we call a *constant-factor time hierarchy theorem*, was proved in [4] for a somewhat untraditional computational model: a simple programming language called `I` (the untraditional aspect is that one considers a class of programs rather

*Work partially supported by DART, a project under the Danish research councils.

than a class of "machines"). For a detailed presentation of the proof, see [2] or [3]. In fact, the same property has been proved before for a more traditional model, the random access machine [7]. As a followup to Jones' first result, the proof has been reiterated for some additional programming-language models [4, 5]. All these results have a similar form, and are also proved in a similar way: by diagonalization, using an efficient interpreter (universal program). The discussion in this note applies to all the models mentioned, since it builds on the known hierarchy result using constructions that can be carried out with little change on various models. For concreteness, we use Jones' programming-language model; we now give the necessary definitions for presenting and then extending his result.

The programming language `I` is an imperative language (including assignment, loops and conditionals) where all data values belong to the set $\mathbb{D}$ of binary trees over the atom `nil` (LISP data with a single atom). Tree operators are `hd`, `tl` and `cons` (in LISP-speak: car, cdr and cons, respectively). The size of tree `d` is denoted by $|\mathtt{d}|$. A little oddity of the binary tree data-type (of no important consequence, except for requiring some care in calculations) is that $|\mathtt{d}|$ is always odd.

The input to an `I` program is a single element of $\mathbb{D}$, and on termination it returns such an element. Program `p` is said to *decide* a set $A \subseteq \mathbb{D}$ if it returns a tree different from `nil` for input in A and the tree `nil` for input not in A. Running time for `I` programs is defined by the ordinary unit-cost measure. The class of sets that are decidable by `I` programs in time bounded by $T(n)$ (n being the size of the input) is denoted by $\mathrm{TIME}^{\mathtt{I}}[T(n)]$. We allow the bound to be broken in a finite number of points; more precisely, in defining $\mathrm{TIME}^{\mathtt{I}}[T(n)]$ we allow any program whose running time on input of size n is bounded by $T(n)$ almost everywhere. This choice is standard in complexity theory; in [2] it is pointed out that the hierarchy result holds with or without this allowance.

Function $T : \mathbb{N} \to \mathbb{N}$ is called *time-constructible* (in `I`) if $T(n)$ can be computed (by an `I` program) in time $O(T(n))$, where numbers are represented in unary (this means that n is represented by a list of length n). Time constructible functions include all common functions expressing running time of algorithms.

In the following we assume for convenience that T is always non-decreasing and defined over the positive real numbers, while having integer values at integral points (to which time-constructibility relates).

Theorem 1 ([2]) *For every time-constructible function $T(n) \geq n$, there is a constant b such that $\mathrm{TIME}^{\mathtt{I}}[b \cdot T(n)] \setminus \mathrm{TIME}^{\mathtt{I}}[T(n)]$ is not empty.*

The constant b appearing in the theorem depends on the function T as well as on the efficiency of a universal program that is adhered to in the proof. Interested in finding out how big this constant is, we repeated in [3] the proof from [2] with explicit calculation of all constants, obtaining the result $b = a + 125$, where a is such that $T(n)$ can be computed in time bounded by $a \cdot T(n)$. It was obvious, though, that this value is an artifact of the proof and not necessarily a property of the computational model. Possibly, a tighter result could be obtained with better analysis. We say that a computational model possesses a *multiplicatively dense* time hierarchy if for (appropriate) time bounds $T(n)$, multiplying $T(n)$ by $1+\varepsilon$ allows more sets to be decided, for arbitrarily small $\varepsilon > 0$. Perhaps the hierarchies for the above computational models are actually dense?

A simple corollary of Theorem 1 shows that "the constant that matters" can *sometimes* be as near to one as desired. Again, we quote [2], while [7] includes an equivalent corollary for the RAM model.

Corollary 1 ([2]) *For every constant $\varepsilon > 0$, and time-constructible $T(n)$, there is a constant $k \geq 1$ such that*

$$\mathrm{TIME}^{\mathtt{I}}[(1+\varepsilon)^k \cdot T(n)] \setminus \mathrm{TIME}^{\mathtt{I}}[(1+\varepsilon)^{k-1} \cdot T(n)]$$

is not empty.

Proof. Fix the function $T(n)$ and the constant ε. For $k = 1, 2, \ldots$ define

$$S_k = \mathrm{TIME}^{\mathtt{I}}[(1+\varepsilon)^k \cdot T(n)] \setminus \mathrm{TIME}^{\mathtt{I}}[(1+\varepsilon)^{k-1} \cdot T(n)]$$

and let b be given by the last theorem; $\mathrm{TIME}^{\mathtt{I}}[b \cdot T(n)] \setminus \mathrm{TIME}^{\mathtt{I}}[T(n)]$ is not empty. Let $\ell = \lceil \log_{1+\varepsilon} b \rceil$. Then

$$\mathrm{TIME}^{\mathtt{I}}[b \cdot T(n)] \setminus \mathrm{TIME}^{\mathtt{I}}[T(n)] \subseteq \bigcup_{k=1}^{\ell} S_k$$

hence the last union is not empty. Therefore, there is a k such that S_k is non-empty. $\square$

Note that this corollary does *not* show the hierarchy to be dense; there could be a gap between $\mathrm{TIME}^{\mathtt{I}}[T(n)]$ and $\mathrm{TIME}^{\mathtt{I}}[(1+\varepsilon)^{k-1} \cdot T(n)]$. In this note we obtain a multiplicatively dense hierarchy for superlinear times of polynomial growth, and come close for linear time. These results are obtained by applying the classic "translational technique" (padding) to the above hierarchy theorems; this technique was developed and used in the past for obtaining similar results, though not with constant-factor hierarchies (an early reference is [6]; a recent, concise presentation can be found in [1]).

2 The translational lemma

For a set $A \subseteq \mathbb{D}$, and a rational constant $c > 1$, define

$$A_{c\text{-pad}} = \{\mathtt{d} \mid \mathtt{d} = (\mathtt{y.z}), \mathtt{y} \in A, |\mathtt{d}| \geq c|\mathtt{y}|\}\,.$$

Lemma 1 *There are constants k_1, k_2 such that, for every rational constant $c > 1$, and every non-decreasing time bound $T(n)$,*

$$A \in \mathrm{TIME}^{\mathtt{I}}[T(n)] \Longrightarrow A_{c\text{-pad}} \in \mathrm{TIME}^{\mathtt{I}}[k_1 n + T(n/c)]$$

and

$$A_{c\text{-pad}} \in \mathrm{TIME}^{\mathtt{I}}[T(n)] \Longrightarrow A \in \mathrm{TIME}^{\mathtt{I}}[k_2 cn + T(cn+2)]$$

Proof. *First implication*: Let `p` be a program deciding A in time $T(n)$. Consider the following program, deciding $A_{c\text{-pad}}$:

```
read X;
if |X| < c · |hd X| then X := nil
else X := p(hd X);
write X
```

As we show below, the test $|\mathtt{d}| < c \cdot |\mathtt{hd\ d}|$, for any tree `d`, can be implemented in time bounded almost everywhere by $k_1|\mathtt{d}|$ for a constant k_1 independent of c. Thus, the above program establishes that $A_{c\text{-pad}} \in \mathrm{TIME}^{\mathtt{I}}[k_1 n + T(n/c)]$.

Implementation of the test. Let $c = m/n$ with $m > n$ natural. Let $\mathtt{x} = \mathtt{hd\ d}$. The condition to test can be rewritten as

$$\frac{1}{m}|\mathtt{d}| < \frac{1}{n}|\mathtt{x}|\,. \tag{1}$$

The test can thus be implemented in three stages:

1. Flatten each of the trees to be compared, namely `d` and `x`, into a linear list, so that instead of dealing with tree *sizes* we can deal with list *lengths*. Flattening a tree takes linear time, so the time of this stage is $O(|\mathtt{d}| + |\mathtt{x}|) = O(|\mathtt{d}|)$.

2. Perform the divisions. Dividing a list by a constant, say m, amounts to a loop that in each iteration removes m elements from the list. It takes $O(s)$ time to divide a list of length s, independent of m (which only affects the length of the code). Thus this stage too takes $O(|\mathtt{d}|)$, independent of c. When the loop terminates we have the quotient and the remainder, represented as lists (unary notation).

3. Let q_m, r_m (respectively q_n, r_n) be the quotient and remainder from dividing $|\mathtt{d}|$ by m (respectively $|\mathtt{x}|$ by n). To test (1), first compare q_m with q_n. This obviously takes $O(q_m + q_n) = O(|\mathtt{d}|)$ time. Now, if $q_m < q_n$ the relation holds; if $q_m > q_n$ it does not; and if $q_m = q_n$ we have to test the remainders. Relation (1) will hold if and only if $r_m/m < r_n/n$, or equivalently $r_m n < r_n m$. These multiplications and comparison can be carried out with trite loops in time $O(r_m n + r_n m) = O(mn)$, which can be ignored for $|\mathtt{d}|$ large enough.

We remark that a simpler implementation, computing $n|\mathtt{d}|$ and $m|\mathtt{x}|$ in unary, would not satisfy our needs because the constant factor in its running time would depend on m and n.

Second implication: Let `p` be a program that accepts $A_{c\text{-pad}}$ in time bounded almost everywhere by $T(n)$. We create from `p` a program `q` to accept A. On input `x`, the program first pads it to obtain `d = (x.y)` such that $|\mathtt{d}| \geq c|\mathtt{x}|$, then runs `p` on `d`. Using the minimum amount of padding necessary to exceed $c|\mathtt{x}|$ guarantees that $|\mathtt{d}| < c|\mathtt{x}| + 2$ (the additive 2 is due to the fact that tree sizes are odd numbers). Below, we show that this padding can be carried out in time bounded by $k_2 c|\mathtt{x}|$, for a constant k_2 independent of c. Now the running time of `q` is bounded by

$$k_2|\mathtt{d}| + T(|\mathtt{d}|) \leq k_2 c|\mathtt{x}| + T(c|\mathtt{x}| + 2),$$

proving the second implication in the lemma.

The way we carry out the padding is similar to the unpadding procedure described above, so we describe it now in less detail. If $c = m/n$, the requirement $|\mathtt{d}| \geq c|\mathtt{x}|$ can be written as

$$\frac{1}{m}|\mathtt{d}| \geq \frac{1}{n}|\mathtt{x}|$$

or, since $|\mathtt{d}| = |(\mathtt{x}.\mathtt{y})| = |\mathtt{x}| + |\mathtt{y}| + 1$,

$$1 + |\mathtt{y}| \geq \frac{m-n}{n}|\mathtt{x}|\,.$$

To find the minimal such `y`, we first use $O(|\mathtt{x}|)$ time to divide $|\mathtt{x}|$ by n, obtaining q_n and r_n, as above. We then multiply q_n by $m-n$ in the straightforward way, and in time dominated by the size of the resulting product. It rests to increase $|\mathtt{y}|$ by $\lfloor \frac{m-n}{n} r_n \rfloor$. To do this, compute $(m-n)r_n$ and divide by n. The time for this last step, $O((m-n)n)$, can be neglected for large inputs. We conclude that the running time of this procedure is linear in the time of the result, i.e., bounded almost everywhere by $k_2 c|\mathtt{x}|$ for an independent constant k_2. $\square$

3 Improved hierarchy for superlinear time

Definition 1 *For a real constant α, function $T(n)$ has* pseudo-degree α *if for every $c > 0$, $\lim_{x\to\infty} T(cx)/T(x) = c^{\alpha}$.*

This property is satisfied, in particular, by polynomials of degree α.

Theorem 2 *Let $T(n)$ be a time-constructible, non-decreasing, superlinear function of pseudo-degree $\alpha \geq 1$. For every $\delta > 0$, $\mathrm{TIME}^{\mathtt{I}}[(1+\delta)T(n)] \setminus \mathrm{TIME}^{\mathtt{I}}[T(n)]$ is not empty.*

Proof. Let $\varepsilon > 0$ satisfy $(1+\varepsilon)^4 \leq 1+\delta$. By Corollary 1, there are a set A and a number $k > 0$ such that $A \in \mathrm{TIME}^{\mathtt{I}}[(1+\varepsilon)^k \cdot T(n)]$, but $A \notin \mathrm{TIME}^{\mathtt{I}}[(1+\varepsilon)^{k-1} \cdot T(n)]$. If $k \leq 4$, our claim is proved; so assume $k > 4$. Let $c_2 > c_1 > 1$ be rational constants such that

$$(1+\varepsilon)^{k-3} < c_1^{\alpha} < c_2^{\alpha} < (1+\varepsilon)^{k-2} \tag{2}$$

(note that such numbers always exist).

According to Definition 1, for all n large enough we have

$$\begin{aligned} T(n)/T(n/c_1) &> (1+\varepsilon)^{k-3} \quad \text{and} &(3)\\ T(c_2 n)/T(n) &< (1+\varepsilon)^{k-2}. &(4)\end{aligned}$$

Superlinearity of T implies that, for any constants $k, \varepsilon > 0$, for all n large enough we have

$$kn < \varepsilon T(n)\,. \tag{5}$$

We first apply the first part of Lemma 1 to the assumption

$$A \in \mathrm{TIME}^{\mathtt{I}}[(1+\varepsilon)^k \cdot T(n)].$$

It shows that

$$A_{c_1\text{-pad}} \in \mathrm{TIME}^{\mathtt{I}}[k_1 n + (1+\varepsilon)^k \cdot T(n/c_1)].$$

By (5) and (3), for n large enough, we have

$$\begin{aligned} k_1 n + (1+\varepsilon)^k \cdot T(n/c_1) &< \varepsilon T(n) + (1+\varepsilon)^k \cdot T(n)/(1+\varepsilon)^{k-3}\\ &= \varepsilon T(n) + (1+\varepsilon)^3 T(n)\,. \end{aligned}$$

Since for all $\varepsilon > 0$, $\varepsilon + (1+\varepsilon)^3 < (1+\varepsilon)^4$, we have

$$A_{c_1\text{-pad}} \in \mathrm{TIME}^{\mathtt{I}}[(1+\varepsilon)^4 T(n)] \subseteq \mathrm{TIME}^{\mathtt{I}}[(1+\delta)T(n)].$$

We now show that $A_{c_1\text{-pad}} \notin \mathrm{TIME}^I[T(n)]$. Assume the contrary. Then by the second part of Lemma 1, $A \in \mathrm{TIME}^{\mathtt{I}}[k_2c_1n + T(c_1n+2)]$. For n large enough, using (5) and (4), we obtain

$$k_2c_1n + T(c_1n+2) \le k_2c_1n + T(c_2n) \le \varepsilon T(n) + T(n) \cdot (1+\varepsilon)^{k-2}.$$

Since for all $\varepsilon > 0$, $\varepsilon + (1+\varepsilon)^{k-2} < (1+\varepsilon)^{k-1}$, we have

$$A \in \mathrm{TIME}^{\mathtt{I}}[(1+\varepsilon)^{k-1}T(n)],$$

a contradiction. □

4 The linear time hierarchy

Previous work, in particular by Jones, put special emphasis on the hierarchy that exists *within linear time*. What Theorem 1 tells us in this case is that there is a constant b such that for all $a \ge 1$, $\mathrm{TIME}^{\mathtt{I}}[abn] \setminus \mathrm{TIME}^{\mathtt{I}}[an] \neq \emptyset$. Trying to squeeze the constant factor down by the method of Theorem 2 does not result in a multiplicatively dense hierarchy in this case, but it does give an interesting improved version of the former result.

Theorem 3 *There is a constant Δ such that for every $a \ge 1$,*

$$\mathrm{TIME}^{\mathtt{I}}[(a+\Delta)n] \setminus \mathrm{TIME}^{\mathtt{I}}[an] \neq \emptyset\,.$$

Note that the multiplicative constant was changed into an additive one, so the ratio between the upper and the lower time bounds approaches 1 as $a \to \infty$.

Proof. Let k_1 and k_2 be as in Lemma 1. We define $\Delta = k_1 + 2k_2 + 1$.

Let $\varepsilon = 1/(2a)$. Use Corollary 1 to obtain A and k such that $A \in \mathrm{TIME}^{\mathtt{I}}[(1+\varepsilon)^k an]$, and $A \notin \mathrm{TIME}^{\mathtt{I}}[(1+\varepsilon)^{k-1}an]$. A simple calculation shows that, whatever the value of k_2,

$$\frac{1+\frac{2k_2+1}{a}}{1+\frac{k_2}{a}} \ge 1 + \frac{1}{a} > 1+\varepsilon\,;$$

therefore we can find rational numbers c_1, c_2 satisfying

$$\frac{(1+\varepsilon)^k}{1+\frac{2k_2+1}{a}} < c_1 < c_2 < \frac{(1+\varepsilon)^{k-1}}{1+\frac{k_2}{a}}\,.$$

Consider $A_{c_1\text{-pad}}$. By setting $T(n) = (1+\varepsilon)^k an$, we obtain by Lemma 1 that

$$A_{c_1\text{-pad}} \in \mathrm{TIME}^{\mathtt{I}}[k_1 n + (1+\varepsilon)^k an/c_1].$$

Now,

$$k_1 n + (1+\varepsilon)^k an/c_1 \leq k_1 n + (1 + \frac{2k_2+1}{a})an = (a+\Delta)n,$$

So $A_{c_1\text{-pad}} \in \mathrm{TIME}^{\mathtt{I}}[(a+\Delta)n]$.

We now prove that $A_{c_1\text{-pad}} \notin \mathrm{TIME}^{\mathtt{I}}[an]$. Assume the contrary; then, using the second part of Lemma 1, we obtain

$$\begin{aligned} A &\in \mathrm{TIME}^{\mathtt{I}}[k_2 c_1 n + a(c_1 n + 2)] \\ &\subseteq \mathrm{TIME}^{\mathtt{I}}[k_2 c_1 n + ac_2 n] \\ &\subseteq \mathrm{TIME}^{\mathtt{I}}[(k_2 + a)c_2 n] \\ &\subseteq \mathrm{TIME}^{\mathtt{I}}[(1+\varepsilon)^{k-1} an], \end{aligned}$$

a contradiction. □

5 A dense linear-time hierarchy?

The above padding technique does not seem to enable a proof showing that the I linear time hierarchy is multiplicatively dense. However, the technique yields such a result if some deliberate assumptions are made on input representation. These assumptions are more natural for the RAM model so in this section we switch to the RAM.

Theorem 4 *Consider a unit-cost random access machine that can add, subtract, and compare whole numbers. Assume that the input of length n is provided to the program in memory cells* BASE + 1 *through* BASE + n, *where* BASE *is some fixed address, and the value of n is provided in another, fixed address. The initial contents of other memory cells are unspecified. For this model,* TIME(bn) *properly includes* TIME(an), *for all $b > a \geq 1$.*

Proof. We refer to the proof of Theorem 2. This proof relies on the super-linearity of T to bound the time for padding and unpadding, which is linear, by εT. The proof will apply for linear T if padding and unpadding can be done in asymptotically sub-linear time. In fact, the assumptions in the present theorem imply that padding and unpadding need only adjust the value of n: specifically, we define (using a hopefully self-explanatory notation)

$$A_{c_1\text{-pad}} = \{(n; x_1 \ldots x_n) \mid (\lfloor n/c_1 \rfloor; x_1 \ldots x_{\lfloor n/c_1 \rfloor}) \in A\}\,.$$

The program deciding this set computes $\lfloor n/c_1 \rfloor$ and then calls the given program for A (called `pk` in previous sections). In the second part of the proof, when we construct a program `q` to accept A from `p` accepting $A_{c_1\text{-pad}}$, program `q` only needs to multiply the given input length by c_1 before executing `p`.

Since c_1 is a rational constant, by representing it as a fraction we observe that both calculations involve one multiplication and one division by integer constants. Multiplication can be performed with a fixed number of addition operations; division can be implemented in time $O(\log n)$ (in essence by a binary search for the quotient). Thus, the proof of Theorem 2 can be freed of the requirement for super-linearity of T. □

Open problems:

1. Is the `I` linear time hierarchy dense (perhaps, above a certain bound)?
2. What about non-polynomial running times?

Acknowledgments

I am grateful to Neil Jones for the long-lasting cooperation that gave rise to this work. Thanks are also due to Holger Petersen for helpful discussions, and to the anonymous referees, whose comments have been very useful in improving the presentation of this paper.

References

[1] Eric Allender, Michael C. Loui, and Kenneth W. Regan. Complexity classes. In Mikhail J. Atallah, editor, *Algorithms and Theory of Computation Handbook*. CRC Press, 1998.

[2] A. M. Ben-Amram and N. D. Jones. Computational complexity via programming languages: Constant factors do matter. *Acta Informatica*, 37(2):83–120, 2000.

[3] A. M. Ben-Amram and N.D. Jones. A precise version of a time hierarchy theorem. *Fundamenta Informaticae*, 38:1–15, 1999. Special issue dedicated to Arto Salomaa.

[4] N. D. Jones. Constant time factors *do* matter. In Steven Homer, editor, *STOC '93. Symposium on Theory of Computing*, pages 602–611. ACM Press, 1993.

[5] Eva Rose. Linear time hierachies for a functional language machine model. In Hanne Riis Nielson, editor, *Programming Languages and Systems – ESOP'96*, volume 1058 of *LNCS*, pages 311–325, Linköping, Sweden, Apr 1996. Linköping University, Springer-Verlag.

[6] S. Ruby and P. Fischer. Translational methods and computational complexity. In *Proc. Sixth Annual IEEE Symp. on Switching Circuit Theory and Logical Design*, pages 173–178, 1965.

[7] I. H. Sudborough and A. Zalcberg. On families of languages defined by time-bounded random access machines. *SIAM J. Comput.*, 5(2):217–230, June 1976.