The Nonexistence of Time-Optimal Turing Machines with Leading Coefficients

Evan Leach

September 16, 2025

1 Asymptotic Optimality

It is typical to ignore leading coefficients in runtime analysis of programs, since this allows complexity classes to be defined independently of any reasonable choice of computational model. However, many interesting phenomena can only be observed when we consider the leading coefficient, and this coefficient can have a huge implication on the practicality of an algorithm's feasibility (consider universal search, for example).

The downside of considering the leading coefficient is that it becomes necessary to fix a computational model, and unless stated otherwise, we will restrict our attention to deterministic one-tape Turing machines. We will study the existence or nonexistence of optimal Turing machines based on the following definitions:

Definition 1. Given a language L, we call a TM M **weakly optimal** if M decides L with worst-case time complexity $T_M(n)$ and there exists some fixed $\varepsilon > 0$ such that for any other TM M' deciding L with worst-case time complexity $T_{M'}(n)$, we have

$$\limsup_{n\to\infty} \frac{T_{M'}(n)}{T_M(n)} > \varepsilon. \tag{1}$$

Definition 2. In the context of the above definition, we call M **optimal** if (1) holds with $\varepsilon = 1$.

In [1], Hutter fixes a universal Turing machine U and shows that there exist weakly optimal algorithms for any language if we impose certain provability restrictions on the allowed algorithms. We take a different approach. Instead of considering inputs to a fixed universal Turing machine, we fix a tape alphabet Γ (to avoid the linear speedup theorem) and consider arbitrary Turing machines over Γ. In this setting, we find a wide array of languages for which no weakly optimal TM exists.

2 Powers of two

In this section, we set $\Sigma = \{\mathtt{1}\}$, $\Gamma = \{\mathtt{1}, \not{\mathtt{1}}, \sqcup\}$, and $L = \left\{\mathtt{1}^{2^k} : k \in \mathbb{N}\right\}$. We also use a base of 2 for all logarithms. To see that there is no weakly optimal decider for L, we first remark that this language is nonregular and therefore cannot be decided by a one-tape deterministic TM in $o(n \log n)$ time by [2]. This means that any fixed TM M deciding L must satisfy

$$T_M(n) \geq pn \log n \tag{2}$$

for some fixed constant $p > 0$ and all sufficiently large $n \in \mathbb{N}$. In order to show that no TM for L is weakly optimal, it therefore suffices to prove the following:

Theorem 1. *For any $c > 0$, there exists a one-tape deterministic TM M over Γ deciding L such that*

$$T_M(n) < cn \log n \tag{3}$$

for all sufficiently large $n \in \mathbb{N}$.

Proof. We define for $k \geq 1$ the TM M_k by the following procedure on an input string w:

Algorithm 1 Description of M_k

1: Repeatedly move right and replace 1s with $\not{\mathtt{1}}$, ignoring existing $\not{\mathtt{1}}$s and storing in state the number of 1s crossed out as j, until $j = 2^k - 1$ or we are reading a blank

2: If $j = 2^k - 1$ and we are not reading a blank:

- Move right one additional character without changing it to a $\not{\mathtt{1}}$ and increment j

3: If we are reading a blank:

- If $j = 1$, accept
- If $j = 2^p$ for some integer $p > 1$, move back to the beginning of w
- Otherwise, reject

4: Set $j = 0$ and return to step 1

This algorithm decides the language L for any $k \in \mathbb{N}$, and it can be implemented in $O(2^k)$ states. Notice that j is always at most 2^k, so checking whether it is a power of 2 can be handled in a single step by memorizing in state. For an input of size n, it takes $2n + 2$ steps for this algorithm to scan the entire input string ($n + 1$ steps for the forwards direction, and $n + 1$ for the backwards direction). After each scan, the remaining number

of uncrossed 1s in w is divided by 2^k, so there are $\lceil \log(n)/k \rceil$ scans in the worst case. Thus

$$\begin{aligned} T_{M_k}(n) &= \left\lceil \frac{\log(n)}{k} \right\rceil (2n+2) \\ &\leq \frac{2n\log(n)}{k} + 2\log(n) + 2n + 2 \\ &= \frac{2}{k} \cdot n \log n + O(n). \end{aligned}$$

Choosing k sufficiently large so that $2/k < c$ proves the claim.

Given any $\varepsilon > 0$ and TM M with time complexity $T_M(n) \geq pn \log n$ for some $p > 0$, Theorem 1 applied to $c = \varepsilon \cdot p$ gives us a TM M' with time complexity

$$T'_M(n) < \varepsilon \cdot pn \log n + d.$$

We have for this M' that

$$\limsup_{n\to\infty} \frac{T_{M'}(n)}{T_M(n)} \leq \limsup_{n\to\infty} \frac{\varepsilon \cdot pn \log n + d}{pn \log n} = \varepsilon,$$

so M is not weakly optimal. In other words, L has no weakly optimal decider over Γ.

□

3 Observations and a conjecture

We can perform a similar argument on a wide variety of languages. For example, the language $L = \{ww^{\mathcal{R}} : w \in \{0,1\}^*\}$ requires $\Omega(n^2)$ time for any fixed one-tape deterministic TM to decide (see [2]), but we can use a similar construction as the one above to find a sequence of Turing machines M_k deciding L with

$$T_{M_k} \leq k^{-1} \cdot n^2 + O(n)$$

by processing the input in chunks of increasingly large size. This implies that this language too has no weakly optimal decider. In fact, the speedup phenomenon appears in a wide variety of languages.

One place where this phenomenon does not occur is for regular languages. A finite or cofinite regular language can be computed in constant time, and all other regular languages can be computed in linear time. In both cases, there is an optimal Turing machine M with $T_M(n) \in O(1)$ or $T_M(n) = n$. This motivates the following conjecture:

Conjecture 1. *For any decidable language $L \notin \mathsf{DTIME}(O(n))$ and tape alphabet Γ, there is no weakly optimal deterministic one-tape decider for L over Γ.*

Geffert first posed this conjecture in [3] and proved that it is true in the case of nondeterministic one-tape Turing machines. Ben-Amram and others proved in [4] that the conjecture is false for multi-tape deterministic TMs. However, it remains open for deterministic one-tape TMs. Many fundamental building blocks of algorithms, such as checking for equality between two strings, exhibit this speedup phenomenon when the Turing machine uses only a single tape. For this reason, one might expect the conjecture to hold.

References

[1] M. Hutter. "The Fastest and Shortest Algorithm for All Well-Defined Problems". In: *International Journal of Foundations of Computer Science* 13.3 (2002), pp. 431–443.

[2] J. Hartmanis. "Computational complexity of one-tape Turing machine computations". In: *Journal of the ACM* 15.2 (1968), pp. 325–339.

[3] V. Geffert. "A speed-up theorem without tape compression". In: *Theoretical Computer Science* 118.1 (1993), pp. 49–65.

[4] A. M. Ben-Amram, N. H. Christensen, and J. G. Simonsen. "Computational Models with No Linear Speedup". In: *Chicago Journal of Theoretical Computer Science* 2012.07 (2012), pp. 1–24.