A Note on Time and Space

Leen Torenvliet       Peter van Emde Boas

*Department of Computer Science, University of Amsterdam*  
*Nieuwe Achtergracht 166, 1018 WV Amsterdam*
Abstract

In this paper we show that single-tape nondeterministic time $T(n)$ bounded Turing Machines can be simulated on single-tape nondeterministic $\sqrt{T(n)}$ space-bounded Turing Machines in time $T(n)$. 
0.1 Introduction

Determining the relationship between recognizing power of resource-bounded machines is a central and long standing open problem in Computational Complexity Theory. The study of different machine models for sequential computation is essential to obtain a model-independent complexity theory, or to translate results obtained for one model directly to other models without proving them again.

Many theorems have been proved on the relation between different variants of the Turing Machine model. For a Turing Machine (TM for short) time is measured by the number of transitions in a computation and space is measured by the number of different tape cells visited by the head(s) during the computation. As far as space is concerned all different models can simulate each other with constant factor overhead. Results concerning time-complexity show that all different multitape TMs can be represented by two tape TMs if one allows for a logarithmic time overhead (See theorem 12.6 in [2]). The single-tape TM forms an exception here since there is a quadratic lowerbound known for the recognition of palindromes (See theorem 10.7 in [1]). W. Maass [6] and independently P. Vitányi and M. Li [7] proved more general lowerbounds, viz.

- There is a language that is accepted by a deterministic 2-tape TM in linear (even real) time but which requires time $\Omega(n^2)$ time on any deterministic 1-tape TM [6] [7].
- There is a language that is accepted by a deterministic 2-tape TM in linear (even real) time but which is not accepted in time $O(n^2 / \log^5 n)$ by any nondeterministic 1-tape TM [6]
- There is a language that is accepted by a TM with 2 pushdown stores or 2 stacks or one queue but which requires $\Omega(n^2)$ on any 1-tape TM [7].

The present paper addresses the relation between time and space bounds for single-tape TMs. It is obvious that any TM that consumes $S(n)$ space to perform a computation must use up $\Omega(S(n))$ time. On the other hand:

Any language accepted by a $T(n)$ time bounded multitape TM can be accepted by a $T(n)/\log T(n)$ space bounded multi-tape TM

as is shown in [4]. For single-tape TMs even sharper bounds are known [3]:

1
Any language accepted by a single-tape TM in time \( T(n) \geq n^2 \) can be accepted by a single-tape TM in space \( \sqrt{T(n)} \).

Any language accepted by an off-line TM in time \( T(n) \geq n \) can be accepted by an off-line TM in space \( \sqrt{T(n)} \log n \).

In their paper [5], Ibarra and Moran showed that these simulations (which up till then required exponential time) can also be time-efficient; in particular they show:

1. Any language accepted by a single-tape TM in time \( T(n) \geq n^2 \) can be accepted by a single-tape TM in space \( \sqrt{T(n)} \) and time \( T^2(n) \).

2. Any language accepted by an off-line TM in time \( T(n) \geq n \) can be accepted by an off-line tape TM in space bounded by \( \sqrt{T(n)} \log n \) and time bounded by \( \sqrt{T^2(n)}(\sqrt{T(n)}) \log n \).

3. Any language accepted by a single tape nondeterministic TM (NTM) in time \( T(n) \geq n^2 \) can be accepted by a single-tape NTM in space \( S(n) \) and time \( \frac{T^2(n)}{S(n)} \) for any \( \sqrt{T(n)} \leq S(n) \leq T(n) \).

4. Any language accepted by an off-line NTM in time \( T(n) \geq n \) can be accepted by an off-line NTM in space \( S(n) \) and time \( T^2(n)(1 + \frac{n}{S(n)}) \) for any \( \sqrt{T(n)} \log n \leq S(n) \leq T(n) \).

In the present paper we sharpen their result 3, and show for machines operating within reasonably well-behaved—for a definition of space constructability see [2, page 297]—time bounds that:

**Theorem 1** Any language accepted by a single tape NTM in time \( T(n) \) can be accepted by a single-tape NTM in space \( \sqrt{T(n)} \) and time \( T(n) \) provided that \( T(n) \geq n^2 \) and \( \sqrt{T(n)} \) is fully time- and space-constructable.

**on-line Turing Machines** are Turing Machines with a single worktape, and a read only-right only input tape (i.e. any input symbol can only be read once.) For on-line TMs the restriction \( T(n) \geq n^2 \) is not required, and we can adapt the construction used in the proof of theorem 1 to show:

**Theorem 2** Any language accepted by a single tape NTM in time \( T(n) \) can be accepted by an online NTM in time \( T(n) \) and space \( \sqrt{T(n)} \), if \( \sqrt{T(n)} \) is fully time- and space-constructable.
\[
\begin{array}{c|c|c|c|c|c|c}
\mathcal{P}_0 & \mathcal{P}_1 & \mathcal{P}_2 & \mathcal{P}_3 & \cdots & \mathcal{P}_n & \mathcal{P}_{n+1} \\
\end{array}
\]

Figure 0.1: A legal partition of \( \mathcal{F} \)

Note that theorem 2 is in no way an improvement upon results 2 and 4 above, since theorem 2 deals with two different machine models, whilst the results of Ibarra and Moran are about simulations of the off-line model on the off-line model. However the method which is used to prove theorem 1 can be used in the case of off-line (and even online) simulation at the cost of logarithmic time overhead.

0.2 The Trace of a Computation

**Definition 1** A sequence \( \mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, \ldots) \) is a *legal partition* of a semi-infinite tape \( \mathcal{F} \) if for each \( i \):

- \( \mathcal{P}_i \) is a block of a finite number of consecutive cells of \( \mathcal{F} \).
- \( \mathcal{P}_{i+1} \) is directly to the right of \( \mathcal{P}_i \).
- each cell of \( \mathcal{F} \) belongs to a unique \( \mathcal{P}_i \).

(see fig. 1.)

**Definition 2** Let \( M \) be a single tape TM and \( w \) be in \( \Sigma^* \). Let \( \mathcal{P} \) be a legal partition of the tape of \( M \). Then for each nonnegative integer \( n \) the *trace of \( M \) on input \( w \) with respect to \( \mathcal{P} \) after \( n \) steps*, denoted by \( \text{TRACE}(M, w, \mathcal{P}, n) \), is a finite sequence \( bh_1bh_2\ldots bh_bb \) where each \( h_i \) is a finite sequence of pairs \( (d, q) \) with \( d \in \{l, r\} \) and \( q \) is a state of \( M \) and is defined as follows:

1. \( \text{TRACE}(M, w, \mathcal{P}, 0) \) is the empty sequence.
2. If M’s head does not cross a border between two consecutive blocks of \( \mathcal{P} \) during the \( n \)’th move then \( \text{TRACE}(M, w, \mathcal{P}, n) = \text{TRACE}(M, w, \mathcal{P}, n-1) \).

3. If M’s head \textit{does} cross a boundary during the \( n \)’th move:
   
   - If M’s head leaves block \( i \) and enters block \( i - 1 \) in state \( q \) during the \( n \)’th move then if \( \text{TRACE}(M, w, \mathcal{P}, n-1) = bh_1b \ldots bh_{i-1}bh_i \ldots bh_kb \) then \( \text{TRACE}(M, w, \mathcal{P}, n) = bh_1b \ldots bh_{i-1}^bh_i^b \ldots bh_kb \) where:
     
     \[
     h_{i-1}' = h_{i-1}(r, q)
     \]
     
     and
     
     \[
     h_i' = h_i(\ell, q)
     \]
     
   - If M’s head leaves block \( i \) and enters block \( i + 1 \) in state \( q \) during the \( n \)’th move and \( \text{TRACE}(M, w, \mathcal{P}, n-1) = bh_1b \ldots bh_i \ldots bh_{i+1}b \ldots bh_kb \) then \( \text{TRACE}(M, w, \mathcal{P}, n) = bh_1b \ldots bh_i^bh_{i+1}^b \ldots bh_kb \) where:
     
     \[
     h_{i+1}' = h_{i+1}(\ell, q)
     \]
     
     and
     
     \[
     h_i' = h_i(r, q)
     \]
     
     If \( \text{TRACE}(M, w, \mathcal{P}, n-1) = bh_1b \ldots bh_kb \) then \( \text{TRACE}(M, w, \mathcal{P}, n) = bh_1b \ldots bh_i^bh_{i+1}b \ldots bh_kb \) where
     
     \[
     h_{i+1} = (\ell, q)
     \]
     
     and
     
     \[
     h_i' = h_i(r, q)
     \]

\textbf{Definition 3} Let \( M, w, \mathcal{P} \) be as in definition 0.2. Then a (possibly infinite) sequence \( b(d_1^1, q_1^1) \ldots b(d_1^n, q_1^n) \ldots \) is the trace of \( M \) on input \( w \) with respect to \( \mathcal{P} \), denoted by \( \text{TRACE}(M, w, \mathcal{P}) \) if the following holds:

- For each positive integer \( n \), \( \text{TRACE}(M, w, \mathcal{P}) \) contains a subsequence which is equal to \( \text{TRACE}(M, w, \mathcal{P}, n) \).
• For any pair \((d,q)\) in \(\text{TRACE}(M, w, \mathcal{P})\) there is a positive integer \(n\) such that \((d,q) \in [\text{TRACE}(M, w, \mathcal{P}, n) - \text{TRACE}(M, w, \mathcal{P}, n - 1)]\).

The length of a trace is the number of pairs \((d,q)\) in it divided by 2. (This corresponds to the actual number of times \(M\) crosses a boundary).

In the sequel we shall be interested in a special type of legal partition which we now define.

Definition 4 Let \(s \geq 1\) and \(1 \leq j \leq s\). Then the \(j^{th}\) legal partition of size \(s\) (see fig. 2) is the partition \((\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2)\) defined by:

1. \(\mathcal{P}_0\) consists of cells \(1, 2, \ldots, j\).
2. For \(t > 0, \mathcal{P}_t\) consists of cells \(j + (t - 1)s + 1\) through \(j + t \times s\)

0.3 Single-tape NTM’s

We will need the following lemma which is similar to lemma 1 in [5].

Lemma 1 Let \(M\) be of time complexity \(T(n)\). Let \(w\) be in \(\Sigma^*, |w| = n\). For each \(j, 1 \leq j \leq \sqrt{T(n)}\) let \(\mathcal{P}^j\) be the \(j^{th}\) legal partition of size \(\sqrt{T(n)}\) and let \(\ell_j\) be the length of \(\text{TRACE}(M, w, \mathcal{P})\). Then there is a \(j_0\) such that \(1 \leq j_0 \leq \sqrt{T(n)}\) and \(\ell_{j_0} \leq \sqrt{T(n)}\).

Proof: It is not hard to verify that \(\sum_{i=1}^{\sqrt{T(n)}} \ell_i\) gives the total number of times \(M\)’s head, on input \(w\), crosses a boundary between adjacent cells. This number is at most \(T(n)\). Hence \(\ell_1 + \ldots + \ell_{\sqrt{T(n)}} \leq T(n)\). This implies that for some \(j_0, \ell_{j_0} \leq \frac{T(n)}{\sqrt{T(n)}} = \sqrt{T(n)}\).

We are now ready to present the proof of theorem 1.
0.3.1 Proof of the main theorem

Assume that $M$ is a single-tape TM which runs in time $T(n)$ on input $w$ with $|w| = n$, and $T$ a function such that $\sqrt{T(n)}$ is fully space-constructable. We will present a single-tape nondeterministic Turing Machine $M'$ which simulates a computation of $M$ on input $w$, running in time $T(n)$ and $\sqrt{T(n)}$ tape. The tape of $M'$ is divided into 3 tracks (See fig. 3.):

1. a track named INPUT, on which the input is stored.
2. a track named WORK, on which the simulation takes place.
3. a track named MARK, on which various markers can be kept to mark the INPUT and WORK track. Without loss of generality we assume that markers are separated, i.e., a marker meant for INPUT can not be confused with a marker meant for WORK and vice versa.

Informally $M'$ guesses a computation trace of size $\sqrt{T(n)}$, and next tries to verify this trace by subsequently copying blocks of the input tape to the WORK part of the tape and checking by simulation that the part of the trace corresponding to the present block is indeed correct. When all blocks of the trace have been checked in this fashion $M'$ checks that the parts of the trace corresponding to adjacent blocks are consistent with the program of $M$.

In order to be able to derive time bounds on this simulation we give the algorithm in some more detail below:

1. write 3 *s on the WORK track as follows:
   - The first * goes into cell 0.
• Leave $\sqrt{T(n)}$ cells blank to the right and write another $\star$. In the sequel the part of the WORK track which lies between the first and second $\star$ will be called SIMULATION-PART\textsuperscript{1}.

• Leave $3 \times \sqrt{T(n)} + 2$ cells blank to the right, and write the third $\star$. In the sequel the part of the WORK track which lies between the second and third $\star$ will be called TRACE-PART\textsuperscript{2}.

2. Shift to the beginning of the TRACE-PART and write a (guessed) computation trace. If the third $\star$ is encountered during this step then REJECT.

3. Put a $\downarrow$ mark on the MARK track above the first symbol on the INPUT track and put a $\downarrow$ mark on the MARK track above the first cell of the WORK tape which lies directly to the right of the first $\star$ mark on this tape. Also put a $\uparrow$ mark above this spot on the MARK track.

4. Local Verification: Repeat the following steps while the $\uparrow$ mark has not yet reached the right hand $\star$ mark of the SIMULATION-PART.

(a) Copy the next $\sqrt{T(n)}$ cells from INPUT to the SIMULATION-PART using and shifting the two $\downarrow$ marks to count the cells\textsuperscript{3}. The first time this is done this process is interrupted nondeterministically to simulate guessing a partition index, and a temporary mark $\uparrow$ is used to indicate the right hand border of block $0$. Find the first $b$ in the TRACE-PART directly to the right of the SIMULATION-PART, and replace it with a $B$. Find the first pair $(d,q)$ in the TRACE-PART which is not marked with either $\uparrow$ or $\downarrow$. If such a pair cannot be found without crossing another $b$, then the simulation for the present block is finished; Return to step 4a. Else leave a $\downarrow$ mark above this pair to signify that this pair was interpreted as an entrance pair. If $d = \ell$ then move the head to the left-hand side of the SIMULATION-PART, else move the head to the right-hand side of the SIMULATION-PART.

\textsuperscript{1}This space will be used to store consecutive blocks of $P^i$ during simulation of a computation of $M$

\textsuperscript{2}This space will be used to store a computation trace of a computation of $M$

\textsuperscript{3}The length of the SIMULATION-PART is used here to count both the length of the block copied into this part, and to count the number of blocks copied into this part in the entire simulation.
(b) Use the program of machine $M$ to simulate a computation on
the simulation part starting in state $q$ up until the point $M$ tries
to cross the border of the SIMULATION-PART or $M$ reaches a
final state.

c) Find the next unmarked pair $(d, q)$ in the TRACE-PART, and
mark it with $\uparrow$ signifying that this pair was interpreted as an exit
pair if $M$ has crossed a border, or mark it with $\blacklozenge$ if $M$ reached
a final state. In this last case check that there are no more pairs
$(d, q)$ between this pair and the next $B$ if there are REJECT, else
goto step 4d. If

- $d$ does not match the border or
- $q$ does not match the state found in the simulated computa-
tion or
- a pair as indicated cannot be found without crossing a $b$ on
the input tape

then REJECT.

d) shift the $\uparrow$ which is above one of the cells of the SIMULATION-
PART one cell to the right. If this mark is above the $\star$ then goto
step 4a else goto step 5.

5. **Global verification** Check the TRACE-PART to see that indeed all $b$s
—except for the last $bb$ mark which we replace by $BB$ in this step—
have been replaced by $B$s. If not REJECT. Find the first pair $p_0$ in
the TRACE-PART which does not yet have a $+$ mark If this pair is
not marked with $\downarrow$ or then REJECT. Repeat the following steps:

- Find the pair $p_1$ directly to the right of the pair $p_0$ if it is not
marked with $\uparrow$ or $\star$ then REJECT, else if the mark is $\star$ goto
step 6, else replace the mark by $+$.

- Suppose $p_1 = (d, q)$ If
  - $d = r$ find the next pair $p_2$ to the right of $p_1$ beyond the
      next $B$ which is not marked with $\downarrow$. If such a pair cannot be
      found then REJECT. Say $p_2 = (d_2, q_2)$. If $p_2$ is not marked
      with $\downarrow$ or $d_2 \neq \ell$ or $q_2 \neq q$ then REJECT, else mark $p_2$ with
      $\downarrow$ and let $p_0 = p_2$.
  - $d = \ell$ find the next pair $p_2$ to the left of $p_1$ beyond the next $B$
      which is not marked with $\downarrow$. If such a pair cannot be found
then REJECT. Say \( p_2 = (d_2, q_2) \) if \( p_2 \) is not marked with \( \downarrow \) or \( d_2 \neq \ell \) or if \( q_2 \neq q \) then REJECT, else mark \( p_2 \) with + and let \( p_0 = p_2 \).

6. Check that all pairs in the TRACE-PART received either a + or * mark. If so accept, if not reject

Correctness of the simulation described above is operational, if \( M' \) accepts then an accepting computation for \( M \) can be replayed by reorganizing the above steps (instead of performing the computation a block at a time load each block in turn, simulate the computation, check the trace pair, and restore the block). As by assumption the contents of a block does not change between a store and a reload this simulation is equivalent to (though more time consuming then) the simulation above, and its correctness is evident. As far as the complexity is concerned we promised that the simulation would run in time \( T(n) \) time\(^4\), and \( \sqrt{T(n)} \) tape\(^5\). We will have to do some careful counting:

- **Step 1** is preprocessing and costs \( O\left(\sqrt{T(n)}\right) \) time.

- In **Step 2** a trace part of length \( O\left(\sqrt{T(n)}\right) \) time is guessed for a cost of \( O\left(\sqrt{T(n)}\right) \) time.

- **Step 3** is also a preprocessing step in which the head is moved a constant number of times over \( O\left(\sqrt{T(n)}\right) \) cells for a cost of \( O\left(\sqrt{T(n)}\right) \) time.

- In **Step 4a** INPUT is copied to the SIMULATION-PART. As there are \( O\left(\sqrt{T(n)}\right) \) blocks relevant this is done \( O\left(\sqrt{T(n)}\right) \) times. Blocks on the input tape are potentially \( O(T(n)) \) cells apart, but as \( T(n) \geq n^2 \) there are only 2 blocks of the INPUT which actually have to be copied to the SIMULATION-PART. (The rest of the blocks are initially blank) Hence the total cost of the copy operation is bounded by \( O(T(n)) \).

In this step also an unmarked pair is fetched from the TRACE-PART for a cost of \( O\left(\sqrt{T(n)}\right) \) however as this pair is marked immediately

\(^4\)Actually this is time \( O(T(n)) \) but constant factor speedup makes that OK.

\(^5\)Same here via tape compression.
and the TRACE-PART consists initially of $O\left(\sqrt{T(n)}\right)$ unmarked pairs the total cost of this operation cannot exceed $O(T(n))$.

- In step 4b an actual computation of $M$ is simulated hence there must exists a computation of $M'$ in which this step totally consumes $O(T(n))$ time.

- In step 4c the TRACE-PART is entered to mark the exit pair, as in step 4a this also cannot amount to a total cost exceeding $O(T(n))$.

- Step 4d evidently also consumes $O\left(\sqrt{T(n)} \times \sqrt{T(n)}\right) = O(T(n))$.

- In step 5 $O\left(\sqrt{T(n)}\right)$ pairs are checked against each other for a total cost of $O\left(\left(\sqrt{T(n)}\right)^2\right) = O(T(n))$

- In step 6 marks are checked for a cost of $O\left(\sqrt{T(n)}\right)$.

This completes our proof.

0.3.2 On-line NTMs

The simulation described above goes through for the case of on-line Turing Machines. An online Turing Machine is a Turing Machine with a read only-right only input tape and a single work tape. For an on-line Turing Machine simulating a single-tape Turing Machine the simulation works exactly as described above except for step 4a. Instead of copying from the INPUT track, we now copy from the input tape. We can of course leave no markers on the input tape as it is read-only, but there is now an extra head, which can be left in its place on the input tape until the next copy action takes place. The facility of having two heads to copy the input also removes the need to carry information over large portions of tape, and therefore the restriction $T(n) \geq n^2$ can also be dropped in this simulation.

At first sight there seems to be a constructability problem here. Since the length of the input is not know on beforehand, how can we mark a block of $\sqrt{T(n)}$ cells? The answer is that, being nondeterministic, the simulating machine can guess the value of $\sqrt{T(n)}$ and since one of the possible guesses is correct the simulating machine is space-efficient.
0.4 Conclusions

In this paper we discussed properties of single-tape and on-line Turing machines in the context of the relation between time and space. We showed that for nondeterministic Turing Machines the use of space can be optimized to $\sqrt{T(n)}$. In view of the result of Ibarra and Moran [5] this does not seem to be a great improvement, since they already showed that space can be optimized to $\sqrt{T(n)}$ if one allows for a $\sqrt{T(n)}$ time overhead. However our result achieves the same bounds without any time overhead and therefore it is rather dramatic since it cannot be improved upon. It shows on the other hand that any single-tape nondeterministic Turing machine operating in time $T(n) \geq n^2$ using $\omega \left( \sqrt{T(n)} \right)$ time is space-inefficient.
Bibliography


