On The Time Complexity of Moving Turing Machines

Huaiin Chen
huailin AT gmail DOT com

Abstract

A main objective of theoretical computer science is to understand the time complexity needed to solve complex computational problems, and to understand the relations between different models of computation.
In this paper, we extend Alan Turing's theoretical model and discuss the time complexity of a moving turing machine, and conclude that time complexity is relative when machines are under different inertial frames of references. The time complexity of a moving turing machine to solve a problem is strongly correlated to Lorentz factor and the verification complexity. When its speed is close to the light speed, a moving machine can use a polynomial time to solve an intractable problem.

Keywords Turing Machine, Time Complexity, Lorentz Factor, P, NP.

1. Turing Machine

First proposed by Alan Turing in 1936, The Turing Machine(TM) is a powerful computing model, which can do everything that a real computer can do [1]. A single tape Turing machine can be formally defined as a 7-tuple as below [2, 3]:

$$M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$$

where
Q is a finite, non-empty set of states
Γ is a finite, non-empty set of tape alphabet symbols
$b \in \Gamma$ is the blank symbol (the only symbol allowed to occur on the tape infinitely often at any step during the computation)
$\Sigma \subseteq \Gamma \setminus \{b\}$is the set of input symbols
$\delta : (Q \setminus F) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is a partial function called the transition function, where L is left shift, R is right shift. (A relatively uncommon variant allows "no shift", say N, as a third element of the latter set.)
$q_0 \in Q$ is the initial state
$F \subseteq Q$ is the set of final or accepting states.

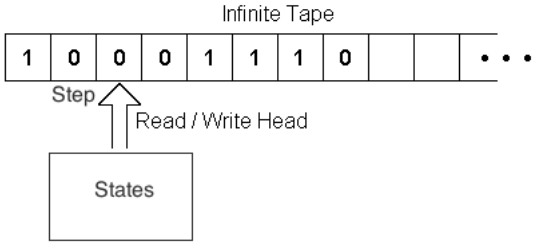

Figure 1. Alan Turing's Machine

2. Time Complexity

A main objective of theoretical computer science is to understand the amount of time and space needed to solve complex computational problems, and to understand the relations between different modes of computation.
Assume M be a deterministic Turing machine, the time complexity of M on input w is the total number of state transitions, or steps, the machine makes before it halts.
A standard convention is to call an algorithm "easy" if it runs in polynomial time. We denote by P the class of decision problems that are solvable in polynomial time. We denote by NP the class of NP decision problems. NP can be defined as the set of decision problems that are solvable in polynomial time by a non-deterministic Turing machine.
The $P =?NP$ problem has been open since the early 1970s. It was essentially first mentioned in a 1956 letter written by Kurt Gdel to John von Neumann [4]. Informally, it asks whether every problem whose solution can be quickly verified by a computer can also be quickly solved by a computer [5, 6].
In this paper, We introduce inertial frame reference onto Alan Turing's theoretical model and discuss the time com-

CONF 'yy, Month d–d, 20yy, City, ST, Country.

http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn

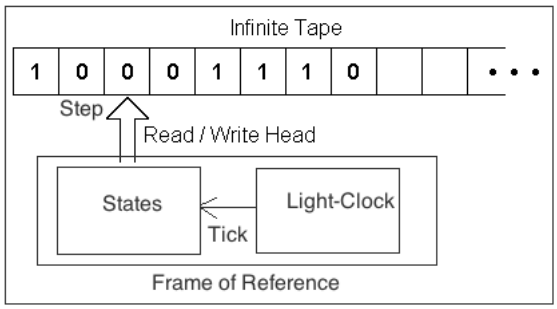

Figure 2. TM with Light-Clock

plexity of a moving turing machine, and conclude that the time complexity is relative when machines are under different inertial frames of references. The computing time complexity of a moving turing machine is tightly correlated to Lorentz factor and the verification complexity. When its speed is close to the light speed, a moving machine can use a polynomial time to solve an intractable problem.

3. Turing Machine under Inertial Frames

We extend Alan Turing's definition to 9-tuple:

$$M = \langle R, C, Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle \tag{1}$$

where R and C are two new elements:

- R is an inertial frame reference that this TM belongs to
- C is the light-clock that is responsible for driving the TM's read/write head.

As figure 2 shows, for an TM, it contains an Einstein light-clock C and the computing read/write head is driven by the light-clock. Every tick means that the machine moves one step. For the clock, we have a blip of light bouncing back and forth between two mirrors facing each other. We position a photocell at the upper mirror, so that it catches the edge of the blip of light. The photocell clicks when the light hits it, and this regular series of clicks drives the clock.
Every time when the clock moves one click, the control state part of the Turing Machine will move one step–change state; move to left, right or stay.

Definition 3.1. Step
A step of a TM machine is equal to a tick of its light-clock. And vice visa.

Definition 3.2. Time Complexity
Suppose an M with light-clock be a deterministic Turing machine that halts on all inputs. The time complexity of M is the function $f : N \to N$, where $f(n)$ is the maximum number of ticks that M spends on an input of length n.

4. On the Relativity of Time Complexity

In this section, we discuss about the time relations among Turing Machines under different frame of references.

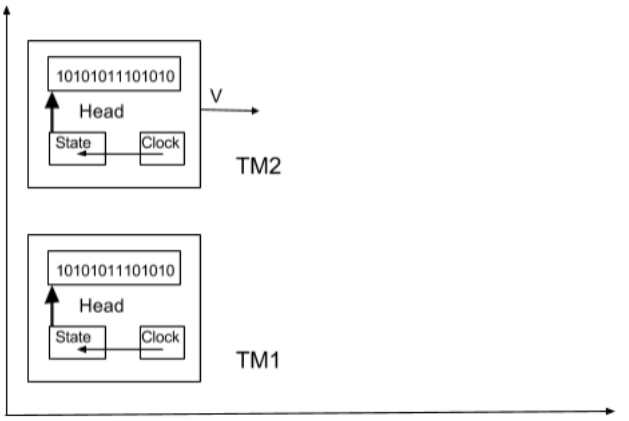

Figure 3. TMs with different references

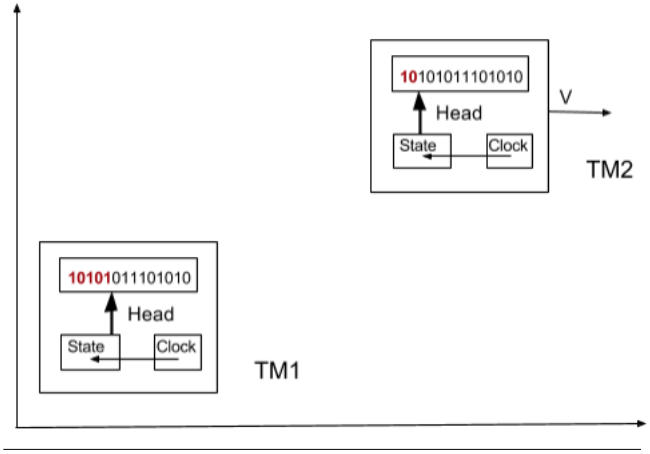

Figure 4. TMs with different references

As shown in figure 3 and 4, we consider that there are two Turing Machines, $TM1$ and $TM2$, where

1. $TM1$ is at rest under a frame of reference $R1$;
2. $TM2$ moves at speed V under a frame of reference $R2$;

From Einstein's relativity theory, we can have that the time (Δt) between two ticks in turing machine $TM2$, is longer than the time (Δt) between these ticks in $TM1$. In other words, $TM2$ light clock is slower than $TM1$ light clock.

$$\Delta T_2 = \gamma * \Delta T_1$$

Where γ is Lorentz factor.

$$\gamma = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}}$$

We conclude TM2 moves fewer steps than TM1 does:

Theorem 1.

$$STEPS_{TM2} = \frac{1}{\gamma} * STEPS_{TM1} \tag{2}$$

In the rest of this paper, we will construct a theoretical computing scenario. As illustrated in figure 5 and 6, there are three TMs to be involved. Two of them are at rest while the another TM is moving with the speed of v. We will discuss the corresponding time complexity issues when these three TMs are under different inertial frames of references.

1. TM1 and TM$\acute{1}$ are at rest in the same reference.
2. TM2 is moving with speed v toward TM$\acute{1}$.
3. There is an unlimited oracle tape in the middle. Whenever TM1 or TM$\acute{1}$ accepts or rejects a string, print results on the corresponding places on the oracle tape before halts.
4. Every TM machine's internal tape has a same input string w, which is a decidable problem.
5. Every TM machine has a read/write head for internal tape and a read/write head for outside
6. TM1 and TM$\acute{1}$ are the exactly clones, able to decide w with time ticks of t.
7. The distance between TM1 and TM$\acute{1}$ is vt

Now lets define the TMs behavior as below:
$TM1(\acute{1}) = \ On\ input\ string\ w:$
{
$Step\ 1: Run\ with\ w$
$Step\ 2: Print\ result\ on\ its\ internal\ tape$
$Step\ 3: Print\ result\ on\ oracle\ tape$
$Step\ 4: Halt$
}

$TM2 = \ On\ input\ string\ w:$
{
$Step\ 1: Check\ oracle\ tape$
$Step\ 2: Repeat\ step\ 1\ if\ no\ result\ available$
$Step\ 3: Verify\ result\ on\ its\ internal\ tape$
$Step\ 4: Print\ result\ on\ its\ internal\ tape$
$Step\ 5: Halt$
}
Suppose that the time complexity for TM$\acute{1}$ to decide(accept or reject) string w is t, where $t = f(n)$, n is the length of w. Then we can formally obtain the time complexity of the moving TM2 as follows:
For step 1 and step 2, the time complexity or steps will be costed is:

$$\frac{1}{\gamma} * f(n) = \sqrt{1 - \frac{v^2}{c^2}} * f(n)$$

For step 3, assume that we have an v(n) time complexity for verifying an algorithm for language w.
The time complexity for the moving machine TM2 is:

Theorem 2. *The time complexity for a moving TM is:*

$$\sqrt{1 - \frac{v^2}{c^2}} * f(n) + v(n)$$

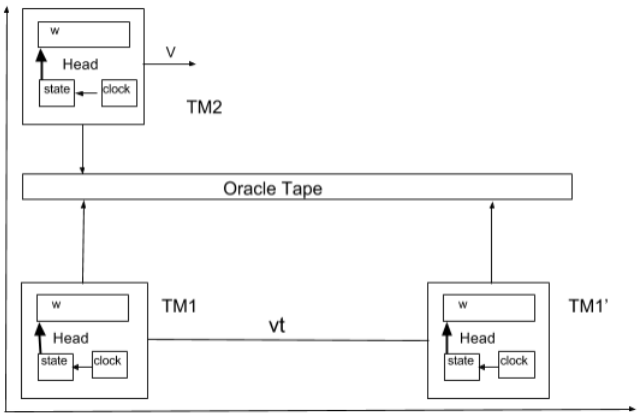

Figure 5. TMs with different references

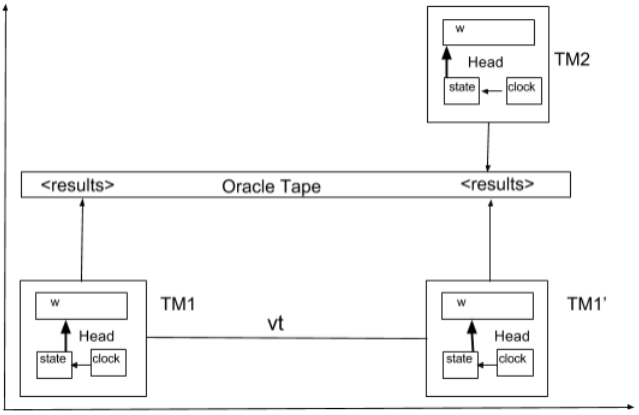

Figure 6. TMs with different references

Where

1. f(n) is the time complexity of the TM at rest frame for deciding a string w.

2. v(n) is the time complexity of the TM for verification an algorithm.

5. Discussion

1. The time complexity for a moving TM is correlated to its moving speed v and the length n of the computing string w.

2. When the moving machine's speed is close to the light speed, its time complexity for w is equal to the verification complexity when under rest.

3. If a moving TM wants to have a constant amount of steps to finish the reading results from the oracle tape,

$$\sqrt{1-\frac{V^2}{C^2}} * f(n) = K$$

Then, it means, the TM need move as fast as below:

Theorem 3.

$$V = \sqrt{1-\frac{K^2}{f(n)^2}} * C \tag{3}$$

Then the time complexity of the moving TM for w is: K+v(n). When v(n) is big enough, the time complexity of the moving TM is simply equal to v(n).
In other words, the computing complexity to decide w for a moving TM2 is equal to the verification complexity of TM1 at rest.

Acknowledgments

References

[1] Turing, A. M. (1937) [Delivered to the Society November 1936]. "On Computable Numbers, with an Application to the Entscheidungsproblem" (PDF). Proceedings of the London Mathematical Society. 2 42. pp. 230-265.

[2] Sipser, Michael: Introduction to the Theory of Computation, Second Edition, International Edition, page 270. Thomson Course Technology, 2006.

[3] Hopcroft, John E.; Rajeev Motwani; Jeffrey D. Ullman (2001). Introduction to Automata Theory, Languages, and Computation (2nd ed.). Reading Mass: AddisonWesley. ISBN 0-201-44124-1. Distinctly different and less intimidating than the first edition.

[4] Juris. "Gdel, von Neumann, and the P = NP problem" (PDF). Bulletin of the European Association for Theoretical Computer Science 38: 101-107.

[5] Fortnow, Lance (2009). "The status of the P versus NP problem" (PDF). Communications of the ACM 52 (9): 78-86. doi:10.1145/1562164.1562186.

[6] Cook, Stephen (1971). "The complexity of theorem proving procedures". Proceedings of the Third Annual ACM Symposium on Theory of Computing. pp. 151-158.