1. **Time/Space tradeoff for SAT**

We don’t know whether SAT can be solved in polynomial time or linear space in deterministic machine. However, we know that SAT can *not* be solved simultaneously in polynomial time and polylog space in a deterministic machine. In particular, we know

**Theorem 1.** SAT $\not\in$ TISP$(n^c, \log n)$ if $c < \sqrt{2}$.

To start with, we prove the following theorem and then get the above result as a straightforward corollary.

**Theorem 2.** NTIME$(n^c) \not\in$ TISP$(n^c, \log^2 n)$ if $c < \sqrt{2}$.

We assume that $t(n)$ is a time constructible function and $s(n)$ is a space constructible function on input size $n$. As stated in the last lecture, we prove a “speed-up lemma” and a “slow-down lemma” using the power of alternations to prove the above theorems.

**Lemma 1 (Speed-up lemma).** TISP$(t(n), s(n)) \subseteq \Sigma_2$TIME$(\sqrt{t(n)s(n)})$.

*Proof.* Let a language $L \in$ TISP$(t(n), s(n))$ and $M$ be the corresponding $k$-tape deterministic turing machine that accepts $L$ simultaneously in time $t(n)$ and space $s(n)$. Fix an input $x$, $|x| = n$. Now consider the configuration of the machine $M$ on input $x$. Remember that the configuration of a machine is determined by the tape head positions and contents of each tape. So the number of bits required to describe a configuration of the machine $M$ is $k(s(n) + \log s(n)) \leq C.s(n)$, where $C$ is some constant. Suppose $C_0$ is the initial configuration and $C_A$ is the accepting configuration. Then we have the following.

$M$ accepts $x$ $\iff$ we can go from $C_0$ to $C_A$ in $t(n)$ steps

$\iff \exists C_m$ such that $M$ can go from $C_0$ to $C_m$ in $t(n)/2$ steps

and $C_m$ to $C_A$ in $t(n)/2$ steps

We write $C_i \xrightarrow{t} C_j$ to denote that $C_j$ can be reached from $C_i$ in $t$ steps.

Now consider the following $\Sigma_2$-machine:

1. (1) start in $\exists$ state.
2. (2) guess configuration $C_m$ of length $O(s(n))$.
3. (3) go to $\forall$ state.
4. (4) guess $x \in \{0, 1\}$.
5. (5) if $x = 0$, accept if $C_0 \xrightarrow{t(n)/2} C_m$.
6. (6) if $x = 1$, accept if $C_m \xrightarrow{t(n)/2} C_A$.

Guessing the configuration $C_m$ takes $O(s(n))$ time and the machine accepts in time $t(n)/2$. So we get $TISP(t(n), s(n)) \subseteq \Sigma_2$TIME$((s(n) + t(n))/2)$. This is only a constant factor improvement in time. Using this idea, we can achieve an asymptotic improvement. Note that instead of going from $C_0$ to $C_A$ in two steps, we can go $C_0 \xrightarrow{t(n)/b} C_1$, $C_1 \xrightarrow{t(n)/b} C_2$, $\ldots$, $C_{b-1} \xrightarrow{t(n)/b} C_b$, where $C_b$ is the accepting state. Now consider the following $\Sigma_2$-machine.
(1) input \(x\), \(|x| = n\).
(2) fix \(b\).
(3) start in \(\exists\) state.
(4) guess configurations \(C_1, \ldots, C_{b-1}\) of length \(O(s(n))\).
(5) go to \(\forall\) state.
(6) guess \(x \in \{0, 1, \ldots, b - 1\}\).
(7) accept if \(C_i \xrightarrow{t(n)/b} C_{i+1}\), else reject.

Total time taken by this new \(\Sigma_2\)-machine is \(O(b.s(n)) + \log b + t(n)/b\). Now our goal is to optimize \(b\) such that this expression is minimum. Simple calculations show that for \(b = \sqrt{t(n)/s(n)}\), the time bound is minimum, i.e., \(\sqrt{t(n)s(n)}\). Hence proved. \(\square\)

Note that \(t(n) \geq n^2\) is a necessary condition for this lemma to hold, as reading the input \((\sqrt{t(n)s(n)} = \sqrt{n\log n} \geq n)\) requires \(n\) time. We have an immediate corollary.

**Corollary 1.** If \(c \geq 2\), \(TISP\left(n^c, \log^2 n\right) \subseteq \Sigma_2 \text{TIME}\left(n^{c/2} \log^{1/2} n\right)\).

Now we prove the “slow-down lemma”.

**Lemma 2** (Slow-down lemma). If \(\text{NTIME}(n) \subseteq TISP(n^c, \log^2 n)\), then \(\Sigma_2 \text{TIME}(t(n)) \subseteq \text{NTIME}(t(n)^c)\).

**Proof.** The hypothesis \(\text{NTIME}(n) \subseteq \text{DTIME}(n^c)\) implies \(\text{co-NTIME}(n) \subseteq \text{DTIME}(n^c)\), as deterministic classes are closed under complement. Now consider a language \(L \in \Sigma_2 - \text{TIME}(t(n))\). Then there exists a constant \(d\) and relation \(R(\ldots, \ldots)\) such that

\[x \in L \iff \exists u, \ |u| = d.t(n) \ \forall v, \ |v| = d.t(n) \ R(x, u, v) = 1\]

Note that the relation \(R\) can be decided in linear time in the input size in deterministic machine, i.e., in \(O(t(n))\). Now define another language \(L'\)

\[L' = \{<x, u> \mid |x| = n \text{ and } |u| = d.t(n) \text{ such that } \forall v, \ |v| = d.t(n) \ R(x, u, v) = 1\}\]

It’s easy to see that \(L' \in \text{co-NTIME}(n)\). So, \(L' \in \text{DTIME}(n^c)\). Then we have the following equivalent statements:

\[x \in L \iff \exists u, |u| = d.t(n) < x, u \in L' \]
\[\iff \exists \text{machine } M' \text{ that decides } L' \text{ in } n^c \text{ time} \]
\[\iff \exists u, |u| = d.t(n)[M'(x, u) \text{ accepts}] \]
\[\iff \exists u, |u| = d.t(n)[R'(x, u) = 1]\]

where the deterministic machine \(M'\) accepts in time \(O((n + t(n))^c) = O(t(n)^c)\) and the relation \(R'\) can be decided in \(O(t(n)^c)\) time. Hence \(L \in \text{NTIME}(t(n)^c)\). \(\square\)

Now the proof the main theorem has the following flavor: . We have the following chain of inclusions. \(\text{NTIME}(n) \subseteq TISP(n^c, \log^2 n) \subseteq \Sigma_2 - \text{TIME}(n^{c/2} \log n) \subseteq \text{NTIME}(n^{c/2} \log^{1/2} n)\). If \(c < \sqrt{2}\), this is a contradiction. This reasoning is almost correct, except that if \(c < 2\), we cannot apply the speed-up lemma. We will see how to get over this minor technical difficulty.
Lemma 3. If \( \text{NTIME}(n) \subseteq \text{TISP}(n^c, \log^2 n) \), then \( \text{NTIME}(n^4) \subseteq \text{TISP}(n^{4c}, \log^2 n) \).

Proof. Let \( L \in \text{NTIME}(n^4) \) and \( \text{NTM} \ M \) decides it. Let \( L_{pad} = \{ < x, 1^{|x|-|x|} > : x \in L \} \). We claim that \( L_{pad} \in \text{NTIME}(n) \) and following is the \( \text{NTM} \ M' \) that accepts \( L_{pad} \).

(1) input \( y \).
(2) check if \( y \) is of the form of strings in \( L_{pad} \).
(3) if yes, then get \( x \) from \( y \), else reject.
(4) check if \( x \in L \) using machine \( M' \).
(5) accept if \( M' \) accepts.

Note that \(|y| = |x|^4\). Hence \( M' \) takes \( O(|x|^4) = O(|y|) \) time. So \( L_{pad} \in \text{NTIME}(n) \). Using the hypothesis, \( L_{pad} \in \text{TISP}(n^c, \log^2 n) \).

Now we claim that \( L \in \text{TISP}(n^{4c}, \log^2 n) \) and to do that we use the \( \text{TISP}(n^c, \log^2 n) \) machine for \( L_{pad} \). Here is the corresponding machine.

(1) input \( x, |x| = n \).
(2) Set \( y = < x, 1^{|x|-|x|}> \).
(3) Accept \( x \) if and only if \( y \in L_{pad} \).

Since \(|y| = n^4\) and \( L_{pad} \in \text{TISP}(n^c, \log^2 n) \), it follows that \( L \in \text{TISP}(n^{4c}, \log^2 n) \).

Proof of theorem 2. We will prove by contradiction. Assume \( \text{NTIME}(n) \subseteq \text{TISP}(n^c, \log^2 n) \). Then if \( c < \sqrt{2}, \)

\[
\text{NTIME}(n^4) \subseteq \text{TISP}(n^{4c}, \log^2 n) \\
\subseteq \text{Σ}_2\text{TIME}(n^{2c} \log n) \\
\subseteq \text{NTIME}(n^{2c} \log^c n)
\]

All of the inclusions follow directly from the above corollaries and lemmas. But this contradicts the nondeterministic time hierarchy, as for \( c < \sqrt{2}, n^{2c^2} \log^c n \in O(n^4) \). Hence proved.

This leads to the following corollary.

Corollary 2. \( SAT \notin \text{TISP}(n^{1.4}, \log n) \).

Proof. Assume otherwise, i.e., \( SAT \) can be solved in time \( n^{1.4} \) and space \( \log n \) simultaneously. Now let \( L \) be a language in \( \text{NTIME}(n) \) and it reduces to \( SAT \) via a function \( f \), i.e., \( L \leq_f SAT \). Using Cook’s reduction, we can show that \( f \) can be computed in time \( O(n \log n) \) and space \( O(\log n) \). Now consider the following algorithm for \( L \):

(1) Input \( x, |x| = n \).
(2) Compute \( f(x) = \phi, |\phi| = n \log n \).
(3) Accept if \( \phi \in SAT \).

Time to run this algorithm is \( n \log n + (n \log n)^{1.4} = O(n \log n)^{1.4} \in O(n^{1.41}) \) and the space requirement is \( \log n + \log(n \log n) = o(\log^2 n) \). Hence \( L \in \text{TISP}(n^{1.41}, \log^2 n) \), which contradicts the Theorem 2.

We can generalize Theorem 2 as follows.

Theorem 3. \( \text{NTIME}(n) \notin \text{TISP}(n^c, n^d) \), if \( c(c + d) < 2 \).

Corollary 3. \( SAT \notin \text{TISP}(n^c, n^d) \), if \( c(c + d) < 2 \).
2. Improvements to the time bounds

The previous proof works as long as $c < \sqrt{2}$. It turns out that we can improve this constant $c$. We will give a very high-level overview of the ideas. Suppose $s(n)$ is negligible.

Our argument had the following structure.

$$
\text{NTIME}(n) \subseteq \text{TISP}(t(n), s(n)) \text{assumption} \\
\subseteq \Sigma_2 \text{TIME}(\sqrt{t(n)}s(n)), \text{from speed-up} \\
\simeq \Sigma_2 \text{TIME}(\sqrt{t(n)})s(n) \text{ is negligible} \\
\subseteq \text{NTIME}(\sqrt{t(n)^c}), \text{from slow-down}
$$

We get a contradiction if $c \geq \sqrt{2}$. We can obtain an improvement if we can speed-up the speed-up lemma or speed-up the slowdown lemma. It seems that the bound on slow-down for is not easy to improve. But we can improve the speed-up bound easily using the power of alternations.

Consider the $\Sigma_2$ machine from the speed-up lemma. Observe that the condition “accept if $C_i \xrightarrow{t(n)/b} C_{i+1}$” is computable in time $t(n)/b$ and space $s(n)$. Instead of checking for this in time $t(n)/b$, we can again use the speed-up lemma. This will increase the number of alternations to 4 and reduce the time. By repeating this, we can obtain a $\Sigma_{2k}$ machine that (approximately) runs in time $t(n)^{1/k+1}$. What happens if we apply the slow down lemma? Each application of the slow-down lemma increase time from $k$ to $k^c$, and we have to apply the slow down lemma $(2k - 1)$ times. It turns out that this will not yield an improvement. However, if we can reduce the number alternations (from $2k$) of the final machine, we will get an improvement.

Consider the condition $C_A$ is reachable from $C$ in $t$ steps. This can be written as follows: For every $C \neq C_A$, $C$ is not reachable from $C$ within $t$ steps. This in turn can be written as follows: For every $C \neq C_A$, for every $C_1, \ldots, C_{b-1}$, $\exists i$ such that $C_i \rightarrow C_{i+1}$ is not reachable in $t(n)/b$ steps. Thus we can express the original condition as a $\forall \exists$-condition also. This mean that we can achieve the speed-up either with a $\Sigma_2$ machine or with a $\Pi_2$ machine. Observe that a $\Sigma_2$ computation followed by a $\Pi_2$ computation can be expressed as a $\Sigma_3$ computation. Using this idea we can bring down the number of alternations from $2k$ to $k$ the time roughly remains that same. This improve the bound on $c$ to the golden ratio. The best known bound on $c$ is $2 \cos(\pi/7) \simeq 1.8$.

3. Open Questions

Open Question 1. Can you get a better bound on $c$, like $c \simeq 1.9$?

Open Question 2. Can you show that $\text{NTIME}(n) \not\subseteq \text{BPTISP}(n^c, \log^2 n)$?

Open Question 3. $\text{SAT} \not\in \text{TIME}(n^c)$, for $c > 1$.

Melkebeek and Raz showed that $\text{SAT} \not\in \text{TIME}(n^a)$, for $a < 1.2$. They used a 2-tape model of Turing machine where random access to the read-only input tape is allowed.

Open Question 4. $\text{SAT} \not\in \text{BPTIME}(n^a)$?
4. REFERENCES

The results presented here are due to Fortnow (JCSS 2000) and Fortnow, Lipton, van Melkebeek and Viglas (JACM 2005). The best know bound on $c \approx 1.8$ is due to Williams(STACS 2010, CCC 2005).