18.405J/6.841J: Advanced Complexity Theory Spring 2016

Lecture 1: PH Time-Space Tradeoffs

Prof. Dana Moshkovitz

Scribe: Anonymous Student

Scribe Date: Fall 2012

1 Overview

The main result of this lecture is that $NTIME(n)$ has algorithms which do not run in time $n^{1.2}$ and use $n^{0.2}$ space. We begin by reviewing some major open problems in complexity theory and describing how this result fits into those open problems. We define the classes in basic complexity theory and the Polynomial Hierarchy. Then we prove the main theorem by using 2 claims. The method we use is an example of a general method for proving similar results, some of which are mentioned at the end of the lecture.

2 Open Problems in Complexity Thoery

The famous question of complexity is whether $P = NP$. Of course, it is known that $P \subseteq NP$, so really the question is whether $NP \subseteq P$. It is conjectured that $NP \not\subseteq P$, but nobody knows how to prove this.

To introduce some notation, NP is the same as the class $NTIME(n^{O(1)})$, the set of languages which are decided by a non-deterministic algorithm which runs in polynomial time. P is the same as $DTIME(n^{O(1)})$, the set of languages decided by a deterministic polynomial-time algorithm.

An slightly easier question is whether $NTIME(n) \not\subseteq DTIME(n^{1.1})$. The difference is that while the P vs. NP question asks about all polynomial-time algorithms, this question specifies which polynomials we are interested in. It is clear that we need to give the deterministic algorithm more time that the nondeterministic one, so for example $DTIME(n^{0.5})$ is certainly weaker than $NTIME(n)$.

What we're doing, then, is taking away the non-determinism, and asking how much more time is required by a deterministic algorithm. The $NP \not\subseteq P$ conjecture implies that in general, we need exponential time when we take away the non-determinism. The conjecture that $NTIME(n) \not\subseteq DTIME(n^{1.1})$ says that even when we specify which polynomials we're interested in, we can't get a polynomial tradeoff. But we can't even prove this– we don't know how to prove there's no containment here.

The results along these lines are much weaker: instead of getting a time increase of $n^{1.1}$, we can get things like $5n$ or $7n$.

However, if we limit the algorithms to small space, we can prove a result along these lines: $NTIME(n)$ contains algorithms which do not run in both $n^{1.2}$ time and $n^{0.2}$ space.

3 Definitions

Here we'll define the complexity classes that will be used in the proof.

3.1 Basic Complexity Theory

As a quick review, the complexity classes P, NP, and $coNP$ are defined as follows:

- P is the set of all languages L such that there exists a polynomial-time algorithm A with $\boxed{x \in L \Leftrightarrow A(x) = 1}$.
- NP is the set of all languages L such that there exists a polynomial-time algorithm A with $\boxed{x \in L \Leftrightarrow \exists y \text{ st. } A(x, y) = 1}$.
- $coNP$ is the set of all languages L such that there exists a polynomial-time algorithm A with $\boxed{x \in L \Leftrightarrow \forall y, A(x, y) = 0}$.

It is known that $P \subseteq NP$ and $P \subseteq coNP$, and it is conjectured that $NP \neq coNP$.

3.2 Polynomial Hierarchy

We can see that NP and $coNP$ simply add a quantifier ($\exists$ or $\forall$) to obtain more strength. We can iterate this process to create more complexity classes with successively more strength.

- Σ_i is the set of languages L for which there's a polynomial time algorithm A with

  $$\boxed{x \in L \Leftrightarrow \exists y_1 \forall y_2 \exists y_3 \cdots y_i \text{ st. } A(x, y_1, y_2, \ldots y_i) = 1}.$$

  Thus $\Sigma_1 = NP$.
- Π_i is the set of languages L for which there's a polynomial time algorithm A with

  $$\boxed{x \in L \Leftrightarrow \forall y_1 \exists y_2 \forall y_3 \cdots y_i \text{ st. } A(x, y_1, y_2, \ldots y_i) = 0}.$$

  Thus $\Pi_1 = coNP$.

We do not need to consider classes which have quantifiers in any other order. For example, if we're examining the class for which the quantifiers are arranged $\exists y_1 \exists y_2 \forall y_3 \forall y_4 \forall y_5 \exists y_6$, we can simply define $y_a = (y_1, y_2)$, $y_b = (y_3, y_4, y_5)$ and this is equivalent to the quantifiers being in the order $\exists y_a \forall y_b \exists y_6$, and thus this is the class Σ_3.

The general conjecture about these classes is **The polynomial hierarchy doesn't collapse.** That is, each new quantifier gives you new power that you didn't have without it. The class PH is the union of all Σ_is, or alternately of all Π_is:

$$PH = \bigcup_{i=1}^{\infty} \Sigma_i = \bigcup_{i=1}^{\infty} \Pi_i.$$

We know that $PH \subseteq PSPACE$.

4 Proof of Main Theorem

We will restate the main theorem of the lecture here for convenience. The notation $TISP(f(n), g(n))$ means the class of problems solvable by algorithms running in time $f(n)$ and space $g(n)$. It is important to note that, $TISP(f(n), g(n))$ **is not** necessarily the same as $DTIME(f(n)) \cap SPACE(g(n))$ – the latter is the class of problems solvable by *either* an algorithm running in time $f(n)$ *or* in space $g(n)$, not necessarily both. However, the following inclusion holds: $TISP(f(n), g(n)) \subseteq DTIME(f(n)) \cap SPACE(g(n))$.

Theorem 1. *$NTIME(n)$ contains algorithms not in $TISP(n^{1.2}, n^{0.2})$.*

Proof. The proof is by contradiction. We will assume that

$$NTIME(n) \subseteq TISP(n^{1.2}, n^{0.2}).$$

In the proof, we scale up all of the exponents by a factor of 10, so we're dealing with n^{12} and n^2. We do this by substituting n^{10}, a padding argument which is left as an exercise. So our assumption is $NTIME(n^{10}) \subseteq TISP(n^{12}, n^2)$.

The proof is built off of the following two claims.

Claim 2. *If $NTIME(n) \subseteq DTIME(n^{1.2})$, then $\Sigma_2 - TIME(n^8) \subseteq NTIME(n^{9.6})$.*

The notation $\Sigma_2 - TIME(n^8)$ simply means that the algorithm A in the definition of Σ_2 must run in time n^8. The precondition of this claim is true by our assumption. Basically, we're saying that if our assumption is true, giving away your nondeterminism means you only need to pay a factor of 1.2 in the exponent. So if we add extra quantifiers, turning the $NTIME$ into a $\Sigma_2 - TIME$ and the $DTIME$ into an $NTIME$, this property persists: the tradeoff for fewer quantifiers is a factor of 1.2 in the exponent of the time. The proof of this is left as an exercise.

Claim 3. $TISP(n^{12}, n^2) \subseteq \Sigma_2 - TIME(n^8)$

The idea is that adding more quantifiers, a process called *alternation*, adds a lot of power. This claim will be proved in Section 5.

With these two claims, we are ready to prove our theorem. We know

$$\begin{aligned} NTIME(n^{10}) &\subseteq TISP(n^{12}, n^2) && \text{by assumption} \\ TISP(n^{12}, n^2) &\subseteq \Sigma_2 - TIME(n^8) && \text{by Claim 3} \\ \Sigma_2 - TIME(n^8) &\subseteq NTIME(n^{9.6}) && \text{by Claim 2} \end{aligned}$$

Thus stringing the containments along, we have

$$NTIME(n^{10}) \subseteq NTIME(n^{9.6}),$$

which is a contradiction of the nondeterministic time-heirarchy theorem: when we have more time, we necessarily have more power.

Thus we've reached a contradiction and the theorem is proved. □

5 Proof of Claim 3

It remains to prove the second claim, which we'll do here.

Proof. Let $L \in TISP(n^{12}, n^2)$. Thus, there is some TM M which decides L and runs in time n^{12} and space n^2.

Consider the configuration graph $G_{M,x}$. Each node in this graph is a configuration, i.e. all the data needed to store the current state of the turing machine. This means we must store what is on the tape, where the head is on the tape, and which internal state we are in. Since M runs in space n^2, we only need $O(n^2)$ space to represent each configuration.

The configuration graph has edges from each configuration to the configuration it visits next. But since this is a deterministic TM, and the configuration graph specifies which x we are working with, the graph is just a path, starting with C_0, the initial cofiguration, and taking a step to the next configuration and the next. Since M runs in time n^{12}, the length of the path is $O(n^{12})$.

Now, partition this length $O(n^{12})$ path into n^6 subpaths, each of length $O(n^6)$. Since we are constructing a Σ_2 algorithm, and Σ_2 algorithms can guess, we'll simply guess on each of these subpaths. More formally:

$$x \in L \Leftrightarrow \exists c_1^*, \ldots, c_{n^6}^* \forall i \in [n^6] \text{ st. the following conditions are satisfied:}$$

- c_i^* is reached from c_{i-1}^* in $O(n^6)$ steps
- $c_{n^6}^*$ is accepting

Essentially, we ensure that each configuration can reach the next one in a small enough amount of steps, but we can guess the steps using our non-determinism to find a shortcut. So long as we end in a configuration which is accepting, we know $x \in L$.

In this algorithm, we have n^6 guesses for the n^6 different c_i^*s. Each of those configurations is of size n^2. Thus this algorithm runs in time $O(n^8)$. Since it is a Σ_2 algorithm because of the ordering of the quantifiers, we know that $L \in \Sigma_2 - TIME(n^8)$. Since L was an arbitrary language, this proof holds for all languages $L \in TISP(n^{12}, n^2)$ and thus $TISP(n^{12}, n^2) \subseteq \Sigma_2 - TIME(n^8)$ as desired.

□

6 General Technique

There were several choices we made in the proof above, such as which particular exponents to use, which could be massaged to possibly produce a better result than this. In fact, the general technique can be described as:

- Claim 1: by the assumption, a certain resource buys me a certain amount of time and/or space
- Claim 2: the resource can't buy you that amout by heirarchy theorems.

6.1 Applications of General Technique

Williams (see [1], [2], or[3]) wrote a computer program to find the best result we can get from the above methods, rather than going through it by hand. He proved the non-inclusion above in the case of $n^{o(1)}$ space and $n^{2\cos\pi/7-o(1)} \geq n^{1.8}$. So while we got $n^1.2$ in class, results are known for about $n^{1.8}$.

Hopcroft, Paul, and Valiant[5] proved in 1977 that $SPACE(n) \not\subseteq DTIME(o(n \lg n))$.

Paul, Pippenger, Szemeredi, and Trotter[4] proved in 1983 that $NTIME(n) \neq DTIME(n)$.

References

[1] R. Ryan Williams, *Time-Space Tradeoffs for Counting NP Solutions Modulo Integers* IEEE 2007: 70-82.

[2] R. Ryan Williams, *Automated proofs of time lower bounds* 2007.

[3] Ryan Williams, *Alternation-Trading Proofs, Linear Programming, and Lower Bounds* STACS 2010: 669-680.

[4] Wolfgang J. Paul, Nicholas Pippenger, Endre Szemerédi, William T. Trotter, *On Determinism versus Non-Determinism and Related Problems* FOCS 1983: 429-438.

[5] John E. Hopcroft, Wolfgang J. Paul, Leslie G. Valiant, *On Time Versus Space.* J. ACM 24(2): 332-337 (1977).

18.405J / 6.841J Advanced Complexity Theory

Spring 2016