Penn State University
CSE 564 Complexity of Combinatorial Problems

*Instructor:* Pei Wu

**Module**
**1**

# Diagonalization

A list of contents mentioned in the first week. All proofs are one-two sentences sketching high-level ideas. Details can be found in AB's book (Ch. 1–3)

## 1.1 Turing machines and languages (AB Ch. 1–2)

A (single-tape) Turing machine can be specified as

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}),$$

with transition function

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}.$$

Decision problems are modeled as languages $L \subseteq \{0,1\}^*$.

**Church–Turing thesis.** Any "physically computable" function or language can be computed/decided by a Turing machine.

**Diagonalization (Cantor).** Cantor's argument shows $|\mathbb{R}| > |\mathbb{N}|$; by an analogous diagonalization, there exist uncomputable languages.

## 1.2 Complexity classes

For a function $f : \mathbb{N} \to \mathbb{N}$,

$$\text{DTIME}(f) = \{L \subseteq \{0,1\}^* : \exists \text{ DTM } M \text{ that decides } L \text{ in } O(f(n)) \text{ steps}\},$$

$$\text{NTIME}(f) = \{L \subseteq \{0,1\}^* : \exists \text{ NTM } M \text{ that decides } L \text{ in } O(f(n)) \text{ steps}\}.$$

**Standard classes.**

$$\mathbf{P} = \bigcup_{c \geq 1} \mathrm{DTIME}(n^c), \qquad \mathbf{NP} = \bigcup_{c \geq 1} \mathrm{NTIME}(n^c).$$

Similarly, **PSPACE** and **NPSPACE** are defined using polynomial space bounds.

A language $L$ is in **NP** if there exists a deterministic polynomial-time verifier $V$ and a polynomial $p$ such that:

- (Completeness) if $x \in L$, then $\exists w \in \{0,1\}^{p(|x|)}$ such that $V(x,w) = 1$.
- (Soundness) if $x \notin L$, then for all $w \in \{0,1\}^{p(|x|)}$, $V(x,w) = 0$.

The satisfiability problem SAT is **NP**-complete.

A language $L$ is in **BPP** if there is a probabilistic polynomial-time machine $M$ such that for all $x$:

$$x \in L \implies \Pr[M(x) = 1] \geq 2/3, \qquad x \notin L \implies \Pr[M(x) = 1] \leq 1/3.$$

A central open problem is $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$.

## 1.3 The Power of Diagonalization

**Time Hierarchy Theorem.** The diagonalization argument can be used show separations between complexity classes. For example, Let $\mathbf{EXP} = \bigcup_{c \geq 1} \mathrm{DTIME}(2^{n^c})$.

Theorem 1.1. $\mathbf{P} \neq \mathbf{EXP}$.

*Sketch.* Construct a machine $M$ that runs in exponential time and diagonalizes against every polynomial-time machine. On input $x$:

1. Interpret $x$ as an encoding $\langle N \rangle$ of a DTM $N$ (if invalid, fix a default $N$).
2. Simulate $N$ on input $x$ for $2^{|x|}$ steps.
3. If $N$ accepts within this bound, reject; otherwise accept.

This runs in exponential time. For any polynomial-time $N$, on input $x = \langle N \rangle$ the simulation is long enough to determine $N(x)$, and $M$ flips the outcome, so $L(M) \notin \mathbf{P}$. □

Generalizing this idea, one gets the full-fledged time hierarchy theorem.

Theorem 1.2 (Hartmanis-Stearns). *For any "time-constructible" $f$ and $g$ such that $f \log f = o(g)$,*

$$\mathrm{DTIME}(f) \subsetneq \mathrm{DTIME}(g).$$

## 1.4 Ladner's theorem

Theorem 1.3 (Ladner, 1975). *If $\mathbf{P} \neq \mathbf{NP}$ then there exists $L \in \mathbf{NP} \setminus \mathbf{P}$ and $L$ is not* **NP***-complete*

*Sketch.* Define Padded SAT

$$\mathrm{SAT}_H := \{\varphi\, 01^{n^{H(|\varphi|)}} : \varphi \in \mathrm{SAT}\},$$

where $H$ is chosen to be time-constructible and to grow in a controlled way. The idea is to pick $H$ so that (i) $\mathrm{SAT}_H$ is not in **P** (else SAT $\in$ **P**), while (ii) padding prevents $\mathrm{SAT}_H$ from being **NP**-complete.

∎

**Remark.**

- "Diagonalize against **NP**" is subtle: diagonalization yields hierarchy theorems but does not directly resolve **P** vs **NP**.
- "Diagonalize against **BPP**" is also unclear in general and interacts with derandomization.

## 1.5 Limitations: the relativization barrier

A proof technique is said to *relativize* if it remains valid when all machines are given access to an arbitrary oracle $O \subseteq \{0,1\}^*$ (writing $\mathbf{P}^O$, $\mathbf{NP}^O$, etc.).

Theorem 1.4 (Baker–Gill–Solovay (1975).). *There exist oracles* $A, B \subseteq \{0,1\}^*$ *such that*

$$\mathbf{P}^A = \mathbf{NP}^A \qquad \text{and} \qquad \mathbf{P}^B \neq \mathbf{NP}^B.$$

*Thus, any relativizing proof cannot resolve* **P** *vs* **NP**.

*Sketch.* For an oracle $O \subseteq \{0,1\}^*$, define

$$L^O := \{1^n : \exists y \in \{0,1\}^n \text{ such that } y \in O\}.$$

Then $L^O \in \mathbf{NP}^O$ (guess $y$ and query the oracle). To diagonalize against $\mathbf{P}^O$ machines, define $O$ in stages: for each polynomial-time oracle machine $M_n$, simulate $M_n^O(1^n)$ long enough to see all oracle queries it makes, and then set membership of some length-$n$ strings in $O$ to force $M_n^O$ to err on input $1^n$. Ensure consistency of the oracle across stages. ∎