CS-E4530

Computational Complexity Theory

Janne H. Korhonen
Aalto University
Spring 2018

Lecture 8:

Beyond NP

Beyond NP

- **We have spent a lot of time on NP-complete problems**
  - Most common and natural type of **intractable** problems
  - Proving NP-hardness is often enough for establishing that there is no polynomial-time algorithm
- **There are also problems outside NP**
  - Useful to be able to recognise such problems
  - Many algorithmic techniques for NP problems do not apply

coNP

coNP: Definition 1

- coNP contains the **complements** of languages in NP
- Essentially problems where **no-instances** are easy to verify

- **Recall:** complement of language L is $\overline{L} = \{x \in \{0,1\}^* : x \notin L\}$

Definition

$$\text{coNP} = \left\{L \subseteq \{0,1\}^* : \overline{L} \in \text{NP}\right\}$$

coNP: Definition 2

Definition

The class **coNP** is the class of all languages $L \subseteq \{0,1\}^*$ for which there exists a polynomial-time Turing machine M and a polynomial function $p\colon \mathbb{N} \to \mathbb{N}$ such that for all $x \in \{0,1\}^*$ we have $x \in L$ if and only if for all $u \in \{0,1\}^*$ with $|u| \leq p(|x|)$ it holds $M(x,u) = 1$.

- For **no-instances** there is a certificate u such that $M(x,u) = 0$ (may assume M outputs 0/1)

coNP-completeness

Definition

We say that a language L is **coNP-complete** if $L \in$ coNP and for any language $L' \in$ coNP, we have $L' \leq_p L$.

Theorem

L is NP*-complete if and only if $\overline{L}$ is* coNP*-complete.*

- **Proof:** apply the same reduction.

coNP-completeness: Example

TAUTOLOGY

- **Instance:** A Boolean formula φ (not necessarily CNF).
- **Question:** Is φ satisfied by *all* possible assignments to its variables?

- **Tautology is coNP-complete:**
  - Let $L \in$ coNP
  - Apply the Cook–Levin reduction from $\overline{L} \in$ NP to CNF-SAT to map instance x to a CNF φ_x
  - Transform φ_x to $\neg\varphi_x$ to get a TAUTOLOGY instance

Structure of P, NP and coNP

coNP, NP and P

- **The following are open questions:**
  - **P** $\neq$ **NP**?
  - **P** $\neq$ **coNP**?
  - **NP** $\neq$ **coNP**?
  - **P** $=$ **NP** $\cap$ **coNP**?

- **Note the following relatioships:**
  - If P $=$ NP, then P $=$ coNP (exercise)
  - NP $=$ coNP **does not** imply P $=$ NP

NP-intermediate problems

Theorem (Ladner's Theorem)

If P $\neq$ NP, *then there is a language* $L \in$ NP $\setminus$ P *that is not* NP-*complete.*

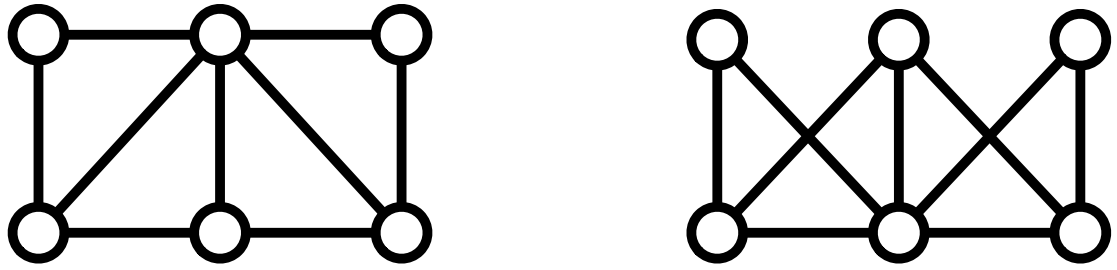
- **No natural problem known to be NP-intermediate**
- **One candidate: graph isomorphism**

Graph Isomorphism

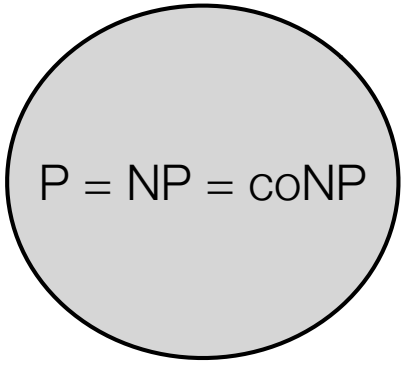
Graph Isomorphism

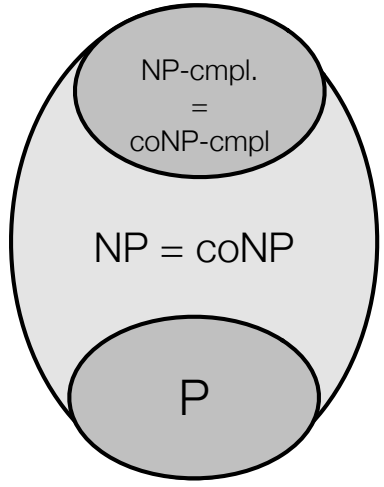
- **Instance:** Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $|V_1| = |V_2|$.
- **Question:** Is there a bijection $f : V_1 \to V_2$ such that

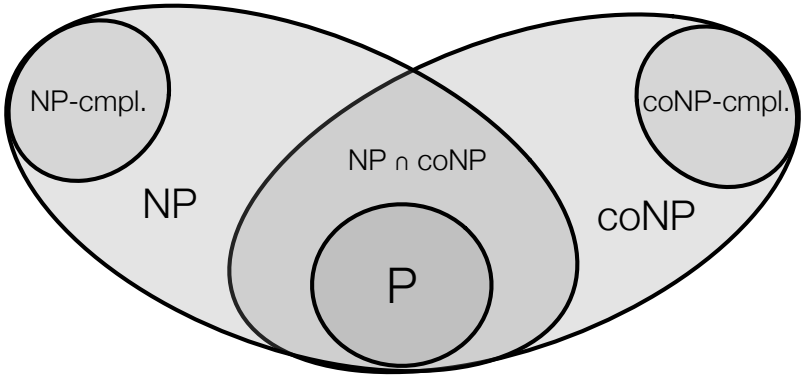
$$\{u, v\} \in E_1 \text{ if and only if } \{f(u), f(v)\} \in E_2\,?$$

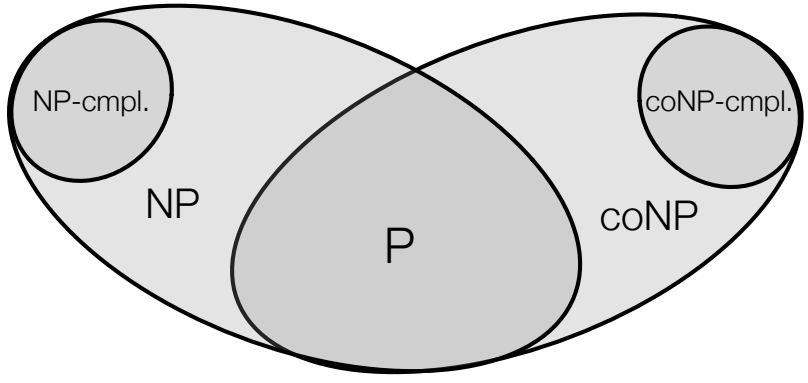


Possible Worlds









Polynomial Hierarchy

Flavours of Independent Set Problem

Maximum independent set (MaxIS)

- **Instance:** Graph $G = (V, E)$, an integer $k \geq 1$.
- **Question:** Is there an independent set of size at least k in G?

Exact independent set (ExactIS)

- **Instance:** Graph $G = (V, E)$, an integer $k \geq 1$.
- **Question:** Is the size of the largest independent set in G exactly k?

Flavours of Independent Set Problem

- **Maximum independent set**
  - Does there **exist** an independent set I with $|I| \geq k$?
- **Complement of maximum independent set**
  - Does it hold **for all** independent sets I that $|I| < k$?
- **Exact independent set**
  - Does there **exist** an independent set I such that **for all** independent sets J we have $|I| \geq |J|$?

Classes Σ_2^p and Π_2^p

Definition

The class Σ_2^p is the class of all languages $L \subseteq \{0,1\}^*$ for which there exists a polynomial-time Turing machine M and a polynomial function $p\colon \mathbb{N} \to \mathbb{N}$ such that for all $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{\leq p(|x|)} \forall v \in \{0,1\}^{\leq p(|x|)} M(x,u,v) = 1\,.$$

Definition

$$\Pi_2^p = \mathrm{co}\Sigma_2^p = \left\{L \subseteq \{0,1\}^* \colon \overline{L} \in \Sigma_2^p\right\}$$

The Polynomial Hierarchy

Definition

The class Σ_k^p is the class of all languages $L \subseteq \{0,1\}^*$ for which there exists a polynomial-time Turing machine M and a polynomial function $p\colon \mathbb{N} \to \mathbb{N}$ such that for all $x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \exists u_1 \forall u_2 \cdots Q u_k M(x, u_1, u_2, \ldots, u_k) = 1\,,$$

where each u_i ranges over binary strings of length at most $p(|x|)$ and Q is either $\exists$ or $\forall$, depending on whether k is odd or even.

Definition

$$\Pi_k^p = \text{co}\Sigma_k^p = \left\{L \subseteq \{0,1\}^*\colon \overline{L} \in \Sigma_k^p\right\}$$

The Polynomial Hierarchy

Definition (The Polynomial Hierarchy)

$$\mathsf{PH} = \bigcup_{k=1}^{\infty} \Sigma_k^p$$

- **Some basic properties of the polynomial hierarchy:**
  - $\Sigma_1^p = \mathsf{NP}$
  - $\Pi_1^p = \mathsf{coNP}$
  - $\Sigma_i^p \subseteq \Pi_{i+1}^p \subseteq \Sigma_{i+2}^p$
  - $\mathsf{PH} = \bigcup_{k=1}^{\infty} \Pi_k^p$

The Polynomial Hierarchy

- **Generally believed that:**
  - $\boldsymbol{\Sigma_k^p \neq \Sigma_{k+1}^p}$ for all $k \geq 1$ ("*polynomial hierarchy does not collapse*")
  - $\boldsymbol{\Sigma_k^p \neq \Pi_k^p}$
- Generalised versions of P $\neq$ NP and NP $\neq$ coNP

Theorem

- *For* $k \geq 1$, *if* $\Sigma_k^p = \Pi_k^p$, *then* PH $= \Sigma_k^p$ (*"hierarchy collapses to level* k*"*).
- *If* P $=$ NP, *then* P $=$ PH (*"hierarchy collapses to* P*"*).

Complete Problems in PH

- **Completeness for Σ_k^p, Π_k^p and PH is defined in terms of polynomial-time many-one reductions**

- **Complete problem for Σ_k^p: Σ_kSAT**
  - Satisfiability for Boolean formulas of form

$$\exists u_1 \forall u_2 \cdots Q u_k \varphi(u_1, u_2, \ldots, u_k) \,,$$

    where φ is a Boolean formula (not necessarily CNF), each u_i is a vector of variables and Q is either $\exists$ or $\forall$, depending on whether k is odd or even.

Complete Problems in PH

- **For PH, complete problems are believed not to exist**

Theorem

If there is a PH*-complete problem, then there exists* k *such that* PH $= \Sigma_k^p$.

- **Proof sketch:**
  - Let L be PH-complete
  - Since $L \in$ PH, we have $L \in \Sigma_k^p$ for some k
  - Let $L' \in$ PH. Since $L' \leq_p L$, we have $L' \in \Sigma_k^p$.

PH: Alternative Definitions

- **Common alternative definitions for PH:**
  - In terms of **oracle Turing machines**:
    Σ_{k+1}^p corresponds to polynomial time with oracle access to a Σ_k^p-complete problem
  - In terms of **alternating Turing machines**:
    Σ_k^p corresponds to $k-1$ alternations

EXP and NEXP

EXP

Definition (EXP)

$\mathsf{EXP} = \bigcup_{d=1}^{\infty} \mathsf{DTIME}(2^{n^d})$

- **Problems solvable in exponential time**
- **P $\subseteq$ NP $\subseteq$ PH $\subseteq$ EXP**

Problems in EXP

- **Contains problems such as determining who wins in generalised versions of games**
- **Canonical problems:** time-bounded halting

Time-bounded halting problem

- **Instance:** A Turing machine M, an integer t (encoded in binary)
- **Question:** Does M halt on empty input in at most t steps?

- **Can be solved by simulating M for t steps**
- **Note:** $t \leq 2^{|x|}$

NEXP

Definition (NEXP)

The class **NEXP** is the class of all languages $L \subseteq \{0,1\}^*$ for which there exists a Turing machine M and polynomial functions $p, q \colon \mathbb{N} \to \mathbb{N}$ such that

- M halts on any input (x, u) in time $O(2^{q(|x|)})$,
- for all $x \in \{0,1\}^*$ we have $x \in L$ if and only if there is $u \in \{0,1\}^*$ with $|u| \leq 2^{p(|x|)}$ such that $M(x,u) = 1$.

- **Equivalent definition:** problems solvable in exponential time with **nondeterministic Turing machines**
- **Unknown if EXP $=$ NEXP**

EXP-completeness and NEXP-completeness

- **Completeness for EXP and NEXP is defined in terms of polynomial-time many-one reductions**

- **Typical complete problems: succinct versions of P-complete and NP-complete problems**
  - **Succinct** means that the input is a representation of an exponential-sized instance, e.g. as a **circuit**
  - EXP-complete problems include generalised versions of **some** games

Polynomial vs. Exponential Time

Theorem

It holds that $\mathrm{P} \subsetneq \mathrm{EXP}$ *and* $\mathrm{NP} \subsetneq \mathrm{NEXP}$.

- **Follows from the time hierarchy theorems** (next week)

Padding and 'Scaling Up'

Theorem

If P $=$ NP, *then* EXP $=$ NEXP.

- **Proof sketch:**
  - Assume P $=$ NP and let $L \in$ NEXP be a language can be verified in time $O(2^{n^c})$
  - Define $L_{\text{pad}} = \{(x, 1^{2^{|x|^c}}) : x \in L\}$
  - $L_{\text{pad}} \in$ NP: any certificate for x (as an instance of L) has length at most $2^{|x|^c}$, which is polynomial in $\left|\llcorner(x, 1^{2^{|x|^c}})\lrcorner\right|$.
  - Since P $=$ NP, we have $L_{\text{pad}} \in$ P, implying there is a polynomial-time Turing machine M deciding L_{pad}
  - $L \in$ EXP: on input x, pad x and solve with M

Lecture 8: Summary

- Complexity classes beyond P and NP
- coNP
- Σ_k^p, Π_k^p and PH
- EXP and NEXP