Lecture 4: **NP** and beyond

Arijit Bishnu

04.02.2010

Outline

Outline

INTEGER PROGRAMMING (IPROG) is NP-complete

For a set of linear inequalities with rational coefficients over variables $x_1, x_2, \ldots, x_n$ is there an assignment of integer numbers in $\{0, 1, \ldots\}$ to $x_1, x_2, \ldots, x_n$ that satisfies it. IPROG $\in$ NP

Lemma

SAT $\leq_P$ IPROG

INTEGER PROGRAMMING (IPROG) is NP-complete

For a set of linear inequalities with rational coefficients over variables $x_1, x_2, \ldots, x_n$ is there an assignment of integer numbers in $\{0, 1, \ldots\}$ to $x_1, x_2, \ldots, x_n$ that satisfies it. IPROG $\in$ NP

Lemma

SAT $\leq_P$ IPROG

Proof

INTEGER PROGRAMMING (IPROG) is NP-complete

For a set of linear inequalities with rational coefficients over variables $x_1, x_2, \ldots, x_n$ is there an assignment of integer numbers in $\{0, 1, \ldots\}$ to $x_1, x_2, \ldots, x_n$ that satisfies it. IPROG $\in$ NP

Lemma

SAT $\leq_P$ IPROG

Proof

- Add the constraints $0 \leq x_i \leq 1$ for every i to ensure that the feasible assignments to the variables are only 0 and 1.

INTEGER PROGRAMMING (IPROG) is NP-complete

For a set of linear inequalities with rational coefficients over variables $x_1, x_2, \ldots, x_n$ is there an assignment of integer numbers in $\{0, 1, \ldots\}$ to $x_1, x_2, \ldots, x_n$ that satisfies it. IPROG $\in$ NP

Lemma

SAT $\leq_P$ IPROG

Proof

- Add the constraints $0 \leq x_i \leq 1$ for every i to ensure that the feasible assignments to the variables are only 0 and 1.
- Now, express every clause as an inequality. As an example, the clause $\overline{x_1} \vee x_2 \vee \overline{x_3}$ can be expressed as $(1 - x_1) + x_2 + (1 - x_3) \geq 1$.

HAMILTONIAN CYCLE (dHAMCYCLE) is NP-complete

Let dHAMCYCLE denote the set of all directed graphs that contain a cycle that visits each vertex exactly once. dHAMCYCLE $\in$ NP.

Lemma

3SAT $\leq_P$ dHAMCYCLE

HAMILTONIAN CYCLE (dHAMCYCLE) is NP-complete

Let dHAMCYCLE denote the set of all directed graphs that contain a cycle that visits each vertex exactly once. dHAMCYCLE $\in$ NP.

Lemma

3SAT $\leq_P$ dHAMCYCLE

Proof

- Construct the graph as follows.
- Show that a satisfying assignment to 3SAT implies a HAMCYCLE and vice versa.

Outline

Decision versus Search

- Any search problem is harder than the corr. decision problem.

Decision versus Search

- Any search problem is harder than the corr. decision problem.
- If P$\neq$NP, then both search and decision problems cannot be solved for an NP-complete problem.

Decision versus Search

- Any search problem is harder than the corr. decision problem.
- If P$\neq$NP, then both search and decision problems cannot be solved for an NP-complete problem.
- If P=NP, then search version of the corr. decision problem can be solved in polynomial time.

Decision versus Search

- Any search problem is harder than the corr. decision problem.
- If P$\neq$NP, then both search and decision problems cannot be solved for an NP-complete problem.
- If P=NP, then search version of the corr. decision problem can be solved in polynomial time.

Theorem

Suppose that P=NP. Then for every language L, $\exists$ a polynomial time TM B that on input $x \in L$ outputs a certificate for x.
That is, as per definition of class NP, $x \in L$ iff $\exists u \in \{0,1\}^{p(|x|)}$ s.t. $M(x,u) = 1$ where p is some polynomial and M is a poly-time TM, then on input $x \in L$, $B(x)$ will be a string $u \in \{0,1\}^{p(|x|)}$ satisfying $M(x, B(x)) = 1$.

Proof of the Theorem for SAT

Proof

- We show that given an algorithm A that decides SAT, we can design an algorithm B.

Proof of the Theorem for SAT

Proof

- We show that given an algorithm A that decides SAT, we can design an algorithm B.
- B finds a satisfying assignment for a satisfiable CNF formula φ with n variables using $2n + 1$ calls to A and some additional poly-time computation.

Proof of the Theorem for SAT

Proof

- We show that given an algorithm A that decides SAT, we can design an algorithm B.
- B finds a satisfying assignment for a satisfiable CNF formula φ with n variables using $2n + 1$ calls to A and some additional poly-time computation.
- First, use A to check if φ is satisfiable.

Proof of the Theorem for SAT

Proof

- We show that given an algorithm A that decides SAT, we can design an algorithm B.
- B finds a satisfying assignment for a satisfiable CNF formula φ with n variables using $2n+1$ calls to A and some additional poly-time computation.
- First, use A to check if φ is satisfiable.
- If yes, set $x_1 = 1$ and $x_1 = 0$ in φ. This shortens the formula to using $n-1$ variables and can be done in poly-time.

Proof of the Theorem for SAT

Proof

- We show that given an algorithm A that decides SAT, we can design an algorithm B.
- B finds a satisfying assignment for a satisfiable CNF formula φ with n variables using $2n+1$ calls to A and some additional poly-time computation.
- First, use A to check if φ is satisfiable.
- If yes, set $x_1 = 1$ and $x_1 = 0$ in φ. This shortens the formula to using $n-1$ variables and can be done in poly-time.
- Use A to decide which one of the two is satisfiable. Say, the first one is satisfiable. Henceforth, fix $x_1 = 1$ and continue.

Proof of the Theorem for SAT

Proof

- We show that given an algorithm A that decides SAT, we can design an algorithm B.
- B finds a satisfying assignment for a satisfiable CNF formula φ with n variables using $2n+1$ calls to A and some additional poly-time computation.
- First, use A to check if φ is satisfiable.
- If yes, set $x_1 = 1$ and $x_1 = 0$ in φ. This shortens the formula to using $n-1$ variables and can be done in poly-time.
- Use A to decide which one of the two is satisfiable. Say, the first one is satisfiable. Henceforth, fix $x_1 = 1$ and continue.
- Continue this for n variables while ensuring that each intermediate formula is satisfiable. Thus, the final assignment to the variables satisfies φ. In all $2n+1$ calls to A were made.

Outline

Understanding Complement Problems

Complement of a Language

If $L \subseteq \{0,1\}^*$ is a language, we denote by $\overline{L}$ the complement of L. That is $\overline{L} = \{0,1\}^* \setminus L$.

Understanding Complement Problems

Complement of a Language

If $L \subseteq \{0,1\}^*$ is a language, we denote by $\overline{L}$ the complement of L. That is $\overline{L} = \{0,1\}^* \setminus L$.

Example

Let L be: Is a graph G 2-colorable? Then, $\overline{L}$ is: Is G not 2-colorable?

Understanding Complement Problems

Complement of a Language

If $L \subseteq \{0,1\}^*$ is a language, we denote by $\overline{L}$ the complement of L. That is $\overline{L} = \{0,1\}^* \setminus L$.

Example

Let L be: Is a graph G 2-colorable? Then, $\overline{L}$ is: Is G not 2-colorable?

Example

Let L be: SAT. Then, $\overline{L}$ is: Is there no assignment of truth values to satisfy a CNF φ? i.e., Is φ unsatisfiable?

Closure under Complementation

Definition: Closed under Complementation

A class $\mathcal{C}$ is closed under complementation if for a problem $A \in \mathcal{C}$, $\overline{A} \in \mathcal{C}$

Closure under Complementation

Definition: Closed under Complementation

A class $\mathcal{C}$ is closed under complementation if for a problem $A \in \mathcal{C}$, $\overline{A} \in \mathcal{C}$

Theorem

The class P is closed under complementation, i.e. coP=P.

Closure under Complementation

Definition: Closed under Complementation

A class $\mathcal{C}$ is closed under complementation if for a problem $A \in \mathcal{C}$, $\overline{A} \in \mathcal{C}$

Theorem

The class P is closed under complementation, i.e. coP=P.

Definition: coNP

$\text{coNP} = \{L \mid \overline{L} \in \text{NP}\}$

Closure under Complementation

Definition: Closed under Complementation

A class $\mathcal{C}$ is closed under complementation if for a problem $A \in \mathcal{C}$, $\overline{A} \in \mathcal{C}$

Theorem

The class P is closed under complementation, i.e. coP=P.

Definition: coNP

$\text{coNP} = \{L \mid \overline{L} \in \text{NP}\}$

What about NP and coNP?

Understanding coNP

A Proof for $\overline{\text{SAT}} \in \text{NP}$

- Design an NDTM as follows. On input φ, the machine guesses an assignment c.

Understanding coNP

A Proof for $\overline{\text{SAT}} \in \text{NP}$

- Design an NDTM as follows. On input φ, the machine guesses an assignment c.
- If c does not satisfy φ, then it accepts; and if it satisfies φ then it halts without accepting.

Understanding coNP

A Proof for $\overline{\text{SAT}} \in \text{NP}$

- Design an NDTM as follows. On input φ, the machine guesses an assignment c.
- If c does not satisfy φ, then it accepts; and if it satisfies φ then it halts without accepting.
- Does this NDTM do the job?

Understanding coNP

A Proof for $\overline{\text{SAT}} \in \text{NP}$

- Design an NDTM as follows. On input φ, the machine guesses an assignment c.
- If c does not satisfy φ, then it accepts; and if it satisfies φ then it halts without accepting.
- Does this NDTM do the job?
- NO, because it accepts every unsatisfiable φ but it also accepts satisfiable formulae, e.g. every formula that has a single unsatisfying assignment and that was the guess. So, the proof is wrong.

Understanding coNP

A Proof for $\overline{\text{SAT}} \in \text{NP}$

- Design an NDTM as follows. On input φ, the machine guesses an assignment c.
- If c does not satisfy φ, then it accepts; and if it satisfies φ then it halts without accepting.
- Does this NDTM do the job?
- NO, because it accepts every unsatisfiable φ but it also accepts satisfiable formulae, e.g. every formula that has a single unsatisfying assignment and that was the guess. So, the proof is wrong.

What do we take home?

NP and coNP are not that simply related.

Understanding coNP

Recall Definition of class NP

A language $L \subseteq \{0,1\}^*$ is in NP if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0,1\}^*$,

$$x \in L \iff \exists u \in \{0,1\}^{p(|x|)} \text{ such that } M(x,u) = 1$$

Understanding coNP

Recall Definition of class NP

A language $L \subseteq \{0,1\}^*$ is in NP if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0,1\}^*$,

$$x \in L \iff \exists u \in \{0,1\}^{p(|x|)} \text{ such that } M(x,u) = 1$$

Simply Speaking

An input string x is a YES instance iff $\exists$ a short u such that $M(x,u) = 1$.

Understanding coNP

Recall Definition of class NP

A language $L \subseteq \{0,1\}^*$ is in NP if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0,1\}^*$,

$$x \in L \iff \exists u \in \{0,1\}^{p(|x|)} \text{ such that } M(x,u) = 1$$

Simply Speaking

An input string x is a YES instance iff $\exists$ a short u such that $M(x,u) = 1$.

Negate the above

An input string x is a NO instance iff $\forall$ short u, it is the case that $M(x,u) = 0$.

An Alternate Definition of coNP

Another Definition of coNP

For every $L \subseteq \{0,1\}^*$, we say that $L \in$ coNP if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0,1\}^*$,

$$x \in L \iff \forall u \in \{0,1\}^{p(|x|)} \text{ such that } M(x,u) = 0$$

[Note the use of $\forall$ in coNP definition instead of $\exists$ in NP definition]

An Alternate Definition of coNP

Another Definition of coNP

For every $L \subseteq \{0,1\}^*$, we say that $L \in$ coNP if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0,1\}^*$,

$$x \in L \iff \forall u \in \{0,1\}^{p(|x|)} \text{ such that } M(x,u) = 0$$

[Note the use of $\forall$ in coNP definition instead of $\exists$ in NP definition]

Definition: coNP-complete

A language is coNP-complete if it $\in$ coNP and every coNP language is poly-time reducible to it.

Exploring Relations between P, NP and coNP

Lemma

$P \subseteq NP \cap coNP$

Exploring Relations between P, NP and coNP

Lemma

$P \subseteq NP \cap coNP$

Proof

$coP\ (= P) \subseteq coNP$. So, the result follows.

Exploring Relations between P, NP and coNP

Lemma

$P \subseteq NP \cap coNP$

Proof

$coP\ (= P) \subseteq coNP$. So, the result follows.

Good characterizations and NP ∩ coNP

If $L \in NP \cap coNP$, then L has the following property:

Exploring Relations between P, NP and coNP

Lemma

$P \subseteq NP \cap coNP$

Proof

$coP\ (= P) \subseteq coNP$. So, the result follows.

Good characterizations and NP ∩ coNP

If $L \in NP \cap coNP$, then L has the following property:

- For a YES answer, there is a short proof.

Exploring Relations between P, NP and coNP

Lemma

$P \subseteq NP \cap coNP$

Proof

$coP\ (= P) \subseteq coNP$. So, the result follows.

Good characterizations and NP ∩ coNP

If $L \in NP \cap coNP$, then L has the following property:

- For a YES answer, there is a short proof.
- For a NO answer, there is also a short proof.

Exploring Relations between P, NP and coNP

Lemma

$P \subseteq NP \cap coNP$

Proof

$coP\ (= P) \subseteq coNP$. So, the result follows.

Good characterizations and NP ∩ coNP

If $L \in NP \cap coNP$, then L has the following property:

- For a YES answer, there is a short proof.
- For a NO answer, there is also a short proof.
- Look at the decision version of Max-Flow problem.

Exploring Relations between P, NP and coNP

Lemma

$P \subseteq NP \cap coNP$

Proof

coP $(= P) \subseteq$ coNP. So, the result follows.

Good characterizations and NP $\cap$ coNP

If $L \in NP \cap coNP$, then L has the following property:

- For a YES answer, there is a short proof.
- For a NO answer, there is also a short proof.
- Look at the decision version of Max-Flow problem.
- There is a short proof of the YES answer via Max-Flow algorithm and there is also a short proof of the NO answer via exhibiting a cut.

Exploring Relations between P, NP and coNP

Is $P = NP \cap coNP$?

So, is $P = NP \cap coNP$? No one knows till now. Neither there is any strong opinion on this.

Exploring Relations between P, NP and coNP

Is $P = NP \cap coNP$?

So, is $P = NP \cap coNP$? No one knows till now. Neither there is any strong opinion on this.

What about the relation between NP and coNP?

Exploring Relations between P, NP and coNP

Is $P = NP \cap coNP$?

So, is $P = NP \cap coNP$? No one knows till now. Neither there is any strong opinion on this.

What about the relation between NP and coNP?

- People believe $NP \neq coNP$ just like the belief of $P \neq NP$.

Exploring Relations between P, NP and coNP

Is $P = NP \cap coNP$?

So, is $P = NP \cap coNP$? No one knows till now. Neither there is any strong opinion on this.

What about the relation between NP and coNP?

- People believe $NP \neq coNP$ just like the belief of $P \neq NP$.
- The reason is: It is difficult to believe that as there exists short proofs of YES instances, there will also exist short proofs of the NO instances.

Exploring Relations between P, NP and coNP

Is NP $\neq$ coNP? Proving this would be a bigger step than proving P $\neq$ NP. The next theorem shows that.

Theorem

If NP $\neq$ coNP, then P $\neq$ NP.

Exploring Relations between P, NP and coNP

Is NP $\neq$ coNP? Proving this would be a bigger step than proving P $\neq$ NP. The next theorem shows that.

Theorem

If NP $\neq$ coNP, then P $\neq$ NP.

Proof (via the contrapositive, i.e. P = NP $\implies$ NP = coNP)

Exploring Relations between P, NP and coNP

Is NP $\neq$ coNP? Proving this would be a bigger step than proving P $\neq$ NP. The next theorem shows that.

Theorem

If NP $\neq$ coNP, then P $\neq$ NP.

Proof (via the contrapositive, i.e. P = NP $\implies$ NP = coNP)

- $L \in \text{NP} \implies L \in \text{P} \implies \overline{L} \in \text{P} \implies \overline{L} \in \text{NP} \implies L \in \text{coNP}$.

Exploring Relations between P, NP and coNP

Is NP $\neq$ coNP? Proving this would be a bigger step than proving P $\neq$ NP. The next theorem shows that.

Theorem

If NP $\neq$ coNP, then P $\neq$ NP.

Proof (via the contrapositive, i.e. P = NP $\implies$ NP = coNP)

- $L \in \text{NP} \implies L \in \text{P} \implies \overline{L} \in \text{P} \implies \overline{L} \in \text{NP} \implies L \in \text{coNP}$.
- $L \in \text{coNP} \implies \overline{L} \in \text{NP} \implies \overline{L} \in \text{P} \implies L \in \text{P} \implies L \in \text{NP}$.

Outline

Exponential Analogue of P and NP

Definition: The Class EXP

$$\mathrm{EXP} = \bigcup_{c \geq 0} \mathrm{DTIME}(2^{n^c})$$

Exponential Analogue of P and NP

Definition: The Class EXP

$$\mathrm{EXP} = \bigcup_{c \geq 0} \mathrm{DTIME}(2^{n^c})$$

Definition: The Class NEXP

$$\mathrm{NEXP} = \bigcup_{c \geq 0} \mathrm{NTIME}(2^{n^c})$$

Exponential Analogue of P and NP

Definition: The Class EXP

$$\mathrm{EXP} = \bigcup_{c \geq 0} \mathrm{DTIME}(2^{n^c})$$

Definition: The Class NEXP

$$\mathrm{NEXP} = \bigcup_{c \geq 0} \mathrm{NTIME}(2^{n^c})$$

Lemma

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{EXP} \subseteq \mathrm{NEXP}$$

Exponential Analogue of P and NP

Definition: The Class EXP

$\mathrm{EXP} = \bigcup_{c \geq 0} \mathrm{DTIME}(2^{n^c})$

Definition: The Class NEXP

$\mathrm{NEXP} = \bigcup_{c \geq 0} \mathrm{NTIME}(2^{n^c})$

Lemma

$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{EXP} \subseteq \mathrm{NEXP}$

Proof

Trivial.

Interplay of EXP, NEXP and P, NP

Theorem

If EXP $\neq$ NEXP, then P $\neq$ NP

Interplay of EXP, NEXP and P, NP

Theorem

If EXP $\neq$ NEXP, then P $\neq$ NP

Proof (via the contrapositive, i.e. P = NP $\implies$ EXP = NEXP)

Left as an exercise.