Various Problems with Time Complexity

Xin Li

lixints@cs.jhu.edu
Topics

• CoNP

• NEXP

• Decision vs Search

• NP-Intermediate languages

• Dealing with NP-completeness
CoNP and NEXP

• The class coNP:

  • A language $L \in \text{coNP}$ if its complement $\neg L \in \text{NP}$.

  • coNP consists of all languages whose complement are in NP.

• Note: coNP is not the complement of NP! In fact, $P \subseteq \text{NP} \cap \text{coNP}$. 
CoNP Second Definition

• Recall the certificate based definition of NP

• A language L is in the class NP if

  \[ \exists \text{ a polynomial function } P: \mathbb{N} \rightarrow \mathbb{N} \text{ and a polynomial time TM } M \text{ s.t. } \forall x \in \{0, 1\}^* \]
  \[ x \in L \iff \exists u \in \{0, 1\}^{P(|x|)} \text{ s.t. } M(x, u) = 1 \text{ (accept)} \]

• Similarly, a language L is in the class coNP if

  \[ \exists \text{ a polynomial function } P: \mathbb{N} \rightarrow \mathbb{N} \text{ and a polynomial time TM } M \text{ s.t. } \forall x \in \{0, 1\}^* \]
  \[ x \in L \iff \forall u \in \{0, 1\}^{P(|x|)} \text{ s.t. } M(x, u) = 1 \text{ (accept)} \]
CoNP-complete

- \( \neg \text{SAT} \) is CoNP-complete (already show \( \in \text{coNP} \)).

- \( \forall L \in \text{coNP}, \text{we have } \neg L \in \text{NP} \).

- By Cook-Levin, \( \neg L \leq_p \text{SAT} \) by a reduction \( f \).

- The same reduction implies that \( L \leq_p \neg \text{SAT} \).
CoNP-complete

• Thus, showing $\neg \text{SAT} \in \text{NP}$ would imply that $\text{coNP} \subseteq \text{NP}$ and in fact $\text{NP} = \text{coNP}$.

• However, whether $\text{NP} = \text{coNP}$ is an open problem.

• Most people believe that $\text{NP} \neq \text{coNP}$, since it seems hard to give a short certificate for the fact “a CNF does not have any satisfying assignment”.
NEXP

• Recall $\text{EXP}=\bigcup_{c \geq 0} \text{DTIME}(2^{n^c})$.

• Can define $\text{NEXP}=\bigcup_{c \geq 0} \text{NTIME}(2^{n^c})$.

• Clearly, $P \subseteq \text{NP} \subseteq \text{EXP} \subseteq \text{NEXP}$.
NEXP

• Theorem: If $\text{EXP} \neq \text{NEXP}$, then $P \neq \text{NP}$.

• Prove the contrapositive, i.e., if $P=\text{NP}$, then $\text{EXP} = \text{NEXP}$.

• Simple argument of padding.
• Suppose $P=NP$. For any language $L \in \text{NEXP}$, we show $L \in \text{EXP}$.

• Say $L \in \text{NTIME}(2^{nc})$ and NDTM $N$ decides it.

• Define $L_{\text{pad}} = \{<x, 1^t> : x \in L\}$ where $t = 2|x|^{c}$

• Then $L_{\text{pad}} \in \text{NP} \Rightarrow L_{\text{pad}} \in \text{P}$.

• Thus, a DTM $M$ decides in $L_{\text{pad}}$ poly time, which implies $L \in \text{EXP}$. 
Decision vs Search

• So far, we are only concerned with decision problems.

• E.g., SAT=given a CNF formula $\psi$, decide if $\psi$ is satisfiable.

• More generally, can consider search problems.

• E.g., given a CNF formula $\psi$, find a satisfying assignment if $\psi$ is satisfiable.


Decision vs Search

• Search is harder than decision.

• However, it is not much harder.

• We can “reduce” search to decision.
**Decision vs Search**

Theorem: If \( \exists \) an algorithm \( A \) that decides SAT in time \( T(A) \), then \( \exists \) an algorithm \( B \) which on input \( \psi \in \text{SAT} \), outputs a satisfying assignment in time \( O(nT(A)) \), where \( n \) is the \# of variables.

Also works for other NP language, as we can first reduce to SAT, and in most cases the reduction also maps certificates to certificates (Levin reduction).
NP-Intermediate Languages

- Ladner’s Theorem (1975): Suppose $P \neq NP$, then $\exists$ a language $L \in NP \setminus P$ that is not NP-complete.

Proof is by diagonalization, which we omit here.

- Some potential examples: factoring (efficiently solvable by quantum computers), graph isomorphism (quasi-polynomial time classical algorithm).
Deal with NP-Completeness

- Many practical problems turn out to be NP-complete.

- Cannot hope to solve them efficiently unless \( P=NP \).

- There are ways to deal with NP-completeness.
Deal with NP-Completeness

- NP-completeness is a **worst-case** analysis.

- E.g., SAT, no efficient algorithm can decide if a given CNF $\psi$ is satisfiable for **every** $\psi$.

- There may be an efficient algorithm that does this for **most** $\psi$.

- Average-case analysis: define some probability distribution on the input.
Deal with NP-Completeness

- Still study the problem in the worst-case

- However, instead of solving the problem exactly, look for an approximation.

  - E.g., find an independent set with size $\geq 1/2 \text{ max independent set}$.

  - E.g., find a vertex cover with size $\leq 2 \text{ min vertex cover}$. 
Deal with NP-Completeness

• In the world of approximation, different NP-complete problems behave quite differently.

• E.g., find a vertex cover with size ≤ 2 \( \min \) vertex cover.

• Easy to achieve by (e.g.) finding a maximum matching in \( G \) (\( \in P \)).
Approximating Vertex Cover

• A maximum matching in G implies a vertex cover with size $\leq 2 \times \text{min vertex cover}$.

• What is the vertex cover?

• Take all vertices in the matching.
Approximating Independent Set

- Approximating Independent Set is a different story.

- Naive $1/n$ approximation: take any vertex.

- Thm: it is NP-hard to obtain a $1/n^{1-\alpha}$ approximation of Independent Set for any constant $\alpha > 0$. 