Lecture 21: Intractability - NP and coNP

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science
rafael.oliveira.teaching@gmail.com

November 23, 2023
Overview

- Complexity Classes & Complete Problems
  - NP
  - coNP
  - Completeness for NP

- Completing Karp Reductions/Polynomial Transformations
  - NP-completeness of 3SAT
  - Current Worldview

- Acknowledgements
Let $\Pi$ be a decision problem and let $L_\Pi$ be the set of all YES instances of $\Pi$. Then $L_\Pi \subseteq \{0, 1\}^*$

decision problems $\leftrightarrow$ subsets of all boolean strings
Let $\Pi$ be a decision problem and let $L_\Pi$ be the set of all YES instances of $\Pi$. Then $L_\Pi \subseteq \{0, 1\}^*$

decision problems $\leftrightarrow$ subsets of all boolean strings

$\text{NP} :=$ class of decision problems $\Pi$ with following property:

- There is a poly-time algorithm $V_\Pi$ and a constant $c > 0$ such that
  - For any $x \in L_\Pi$ (i.e., YES instance) of size $n$, there is a proof/witness $y$ of size $n^c$ such that $V_\Pi(x, y) = 1$.
  - For any $x' \not\in L_\Pi$ (i.e., NO instance) there is no such proof $z$ of size $n^c$ such that $V_\Pi(x', z) = 1$. 

NP
Let $\Pi$ be a decision problem and let $L_\Pi$ be the set of all YES instances of $\Pi$. Then $L_\Pi \subseteq \{0, 1\}^*$

decision problems $\leftrightarrow$ subsets of all boolean strings

NP := class of decision problems $\Pi$ with following property:
- There is a poly-time algorithm $V_\Pi$ and a constant $c > 0$ such that
  - For any $x \in L_\Pi$ (i.e., YES instance) of size $n$, there is a proof/witness $y$ of size $n^c$ such that $V_\Pi(x, y) = 1$
  - For any $x' \notin L_\Pi$ (i.e., NO instance) there is no such proof $z$ of size $n^c$ such that $V_\Pi(x', z) = 1$.

In other words, NP is the class of decision problems where the YES instances have a small proof that can be verified in poly-time
Problems in NP

- Clique
- Independent Set
- SAT (and 3SAT)
- TSP
- Hamilton cycle (and Hamilton path)
- Subset Sum
- Vertex Cover
- 3COLOR (and the graph coloring problem)
- *every* problem in P
Complexity Classes & Complete Problems
- NP
- coNP
  - Completeness for NP

Completing Karp Reductions/Polynomial Transformations
- NP-completeness of 3SAT
- Current Worldview

Acknowledgements
The class coNP is essentially the opposite of NP.

For a decision problem $\Pi$, let $\overline{\Pi}$ be the \textit{opposite} problem to $\Pi$, that is,

$$x \in L_{\Pi} \iff x \notin L_{\overline{\Pi}}$$

equivalently, $L_{\overline{\Pi}} = \overline{L_{\Pi}}$.

In simpler terms, every YES instance of $\Pi$ is a NO instance of $\overline{\Pi}$ (and vice-versa).
The class coNP is essentially the opposite of NP.

For a decision problem \( \Pi \), let \( \bar{\Pi} \) be the opposite problem to \( \Pi \), that is,

\[
x \in L_{\Pi} \iff x \notin L_{\bar{\Pi}}
\]

Equivalently, \( L_{\bar{\Pi}} = \overline{L_{\Pi}} \).

In simpler terms, every YES instance of \( \Pi \) is a NO instance of \( \bar{\Pi} \) (and vice-versa).

coNP := class of decision problems \( \Pi \) such that \( \bar{\Pi} \in \text{NP} \).
Relation between $P$, $NP$ and $coNP$

**Unknown:**

1) is $P = NP \cap coNP$?

2) is $NP = coNP$?

3) is $P = NP$?
- Complexity Classes & Complete Problems
  - NP
  - coNP
  - Completeness for NP

- Completing Karp Reductions/Polynomial Transformations
  - NP-completeness of 3SAT
  - Current Worldview

- Acknowledgements
A remark about reductions

- Given a particular reduction \( \leq \) (Turing, Karp), we can define a complete problem for a complexity class \( C \) as follows:
  - **Hardness**: \( \Pi \) is \( C \)-hard if for every problem \( \Gamma \in C \), we have \( \Gamma \leq \Pi \)
  - **Membership in** \( C \): \( \Pi \in C \)
A remark about reductions

- Given a particular reduction $\leq$ (Turing, Karp), we can define a complete problem for a complexity class $C$ as follows:
  - **Hardness**: $\Pi$ is $C$-hard if for every problem $\Gamma \in C$, we have $\Gamma \leq \Pi$
  - **Membership in $C$**: $\Pi \in C$

Complexity theorists prefer to define NP-completeness under *Karp reductions* (or polynomial transformations) because, as we will see, NP is closed under such reductions
  - Note that we *do not know* whether NP is closed under Turing reductions
  - The above would imply $NP = coNP$, which is considered unlikely

Under Turing reductions, $UNSAT \equiv SAT$
A boolean circuit is a DAG with:

- input gates
- AND/OR/NOT gates,
- and a special gate (the output gate)
CIRCUIT-SAT

- A *boolean circuit* is a DAG with:
  - input gates
  - AND/OR/NOT gates,
  - and a special gate (the output gate)

- **CIRCUIT-SAT** problem:
  - **Input:** a boolean circuit $\Phi$
  - **Output:** YES, if there is a truth assignment $\alpha$ such that $\Phi(\alpha) = 1$, NO otherwise.
Cook-Levin Theorem: CIRCUIT-SAT is NP-complete

Theorem (Cook-Levin)

*CIRCUIT-SAT is NP-complete under polynomial transformations.*
Cook-Levin Theorem: CIRCUIT-SAT is NP-complete

Theorem (Cook-Levin)

*CIRCUIT-SAT is NP-complete under polynomial transformations.*

- Want to prove that for any $\Pi \in \text{NP}$, we have $\Pi \leq_m \text{CIRCUIT-SAT}$

**Proof sketch:** computation is local

- $\Pi \in \text{NP} \Rightarrow \exists$ poly-time verification algorithm $V_\Pi$ and $c > 0$ such that for any instance $x \in \{0, 1\}^n$,

  $$x \in L_\Pi \iff \exists y \in \{0, 1\}^{nc} \text{ s.t. } V_\Pi(x, y) = 1$$
Cook-Levin Theorem: CIRCUIT-SAT is NP-complete

Theorem (Cook-Levin)

CIRCUIT-SAT is NP-complete under polynomial transformations.

- Want to prove that for any $\Pi \in \text{NP}$, we have $\Pi \leq_m \text{CIRCUIT-SAT}$
- **Proof sketch:** computation is local
  - $\Pi \in \text{NP} \Rightarrow \exists$ poly-time verification algorithm $V_\Pi$ and $c > 0$ such that for any instance $x \in \{0,1\}^n$,
    
    $$x \in L_\Pi \iff \exists y \in \{0,1\}^{nc} \text{ s.t. } V_\Pi(x, y) = 1$$
  
  - If $V_\Pi(x, y)$ runs in time $O(n^\gamma)$ (since it is polynomial in terms of the input size), there is circuit of size $O(n^\gamma)$ simulating computation of $V_\Pi$
    
    Can construct this circuit (from description of $V_\Pi$) in $\text{poly}(n)$-time!
Cook-Levin Theorem: CIRCUIT-SAT is NP-complete

**Theorem (Cook-Levin)**

*CIRCUIT-SAT is NP-complete under polynomial transformations.*

- Want to prove that for any \( \Pi \in \mathbb{NP} \), we have \( \Pi \leq_m \text{CIRCUIT-SAT} \)
- **Proof sketch:** *computation is local*
  - \( \Pi \in \mathbb{NP} \Rightarrow \exists \text{ poly-time verification algorithm } V_{\Pi} \text{ and } c > 0 \text{ such that for any instance } x \in \{0,1\}^n, \)

\[
x \in L_{\Pi} \iff \exists y \in \{0,1\}^{nc} \text{ s.t. } V_{\Pi}(x,y) = 1
\]

- If \( V_{\Pi}(x,y) \) runs in time \( O(n^\gamma) \) (since it is polynomial in terms of the input size), there is circuit of size \( O(n^\gamma) \) simulating computation of \( V_{\Pi} \)
- So, we get a poly\((n)\)-sized circuit \( \Phi_x(y) \) which is satisfiable iff \( x \in L_{\Pi} \)!
Cook-Levin Theorem: CIRCUIT-SAT is NP-complete

Theorem (Cook-Levin)

*CIRCUIT-SAT is NP-complete under polynomial transformations.*

Want to prove that for any \( \Pi \in \text{NP} \), we have \( \Pi \leq_m \text{CIRCUIT-SAT} \)

**Proof sketch:** *computation is local*

- \( \Pi \in \text{NP} \Rightarrow \exists \) poly-time verification algorithm \( V_\Pi \) and \( c > 0 \) such that for any instance \( x \in \{0,1\}^n \),

\[
x \in L_\Pi \iff \exists y \in \{0,1\}^{nc} \text{ s.t. } V_\Pi(x, y) = 1
\]

- If \( V_\Pi(x, y) \) runs in time \( O(n^\gamma) \) (since it is polynomial in terms of the input size), there is circuit of size \( O(n^\gamma) \) simulating computation of \( V_\Pi \)

So, we get a poly\( (n) \)-sized circuit \( \Phi_x(y) \) which is satisfiable iff \( x \in L_\Pi \)!

Thus, we have a transformation

\[
x \mapsto \Phi_x
\]

such that \( x \in L_\Pi \iff \Phi_x \in \text{CIRCUIT-SAT} \).
Complexity Classes & Complete Problems
- NP
- coNP
- Completeness for NP

Completing Karp Reductions/Polynomial Transformations
- NP-completeness of 3SAT
- Current Worldview

Acknowledgements
3SAT is NP-complete

To prove this, by Cook-Levin theorem, need to show that
\[ \text{CIRCUIT-SAT} \leq_m \text{3SAT} \]
3SAT is NP-complete

- To prove this, by Cook-Levin theorem, need to show that
  \[ \text{CIRCUIT-SAT} \leq_m 3\text{SAT} \]
- By transitivity of polynomial transformations, enough to show
  \[ \text{CIRCUIT-SAT} \leq_m \text{SAT} \]
3SAT is NP-complete

- To prove this, by Cook-Levin theorem, need to show that
  \[ \text{CIRCUIT-SAT} \leq_m 3\text{SAT} \]
- By transitivity of polynomial transformations, enough to show
  \[ \text{CIRCUIT-SAT} \leq_m \text{SAT} \]
- Let \( \Phi \in \text{CIRCUIT-SAT} \) of size \( n \) (i.e., \( n \) gates and wires). We will construct CNF \( \Psi \) with \( O(n) \) clauses such that
  \[ \Phi \text{ is satisfiable } \iff \Psi \text{ is satisfiable.} \]
3SAT is NP-complete

- To prove this, by Cook-Levin theorem, need to show that
  \[ \text{CIRCUIT-SAT} \leq_m 3\text{SAT} \]
- By transitivity of polynomial transformations, enough to show
  \[ \text{CIRCUIT-SAT} \leq_m \text{SAT} \]
- Let \( \Phi \in \text{CIRCUIT-SAT} \) of size \( n \) (i.e., \( n \) gates and wires). We will construct CNF \( \Psi \) with \( O(n) \) clauses such that
  \[ \Phi \text{ is satisfiable } \iff \Psi \text{ is satisfiable.} \]
- Can do the above simulating gate-by-gate (wire-by-wire):
  - each gate has a new variable, which will tell us the value of the gate
  - Simulate each gate operation (AND/OR/NOT) as a CNF
  - ensure that output gate variable should be true
Gate Simulations

- AND: CNF

\[(\bar{g} \lor u_1) \land (\bar{g} \lor u_2) \land (g \lor \bar{u}_1 \lor \bar{u}_2)\]

If \( u_1 \) or \( u_2 = 0 \) then must have \( g = 0 \) to satisfy above

\[(\bar{g} \lor \bar{u}_1, \lor \bar{u}_2)\]

If both \( u_1 = u_2 = 1 \) then \( g = 1 \)
Gate Simulations

**OR:** CNF

\[(g \lor \overline{u_1}) \land (g \lor \overline{u_2}) \land (\overline{g} \lor u_1 \lor u_2)\]

The diagram illustrates the logic gates and the corresponding CNF expression. The expression is split into two clauses:

1. \((g \lor \overline{u_1}) \land (g \lor \overline{u_2})\) - This clause is satisfied if either \(g\) is 1 or both \(u_1\) and \(u_2\) are 0.

2. \((\overline{g} \lor u_1 \lor u_2)\) - This clause is satisfied if \(u_1 = u_2 = 0\) and \(g\) is 0, or if any of the variables are 1.

For the diagram:
- If \(u_1 = u_2 = 1\), then \(g\) must be 1 to satisfy the above clause.
- If \(u_1 = u_2 = 0\), then \(g\) must be 0 to satisfy the above clauses.
**NOT:** CNF

\[(\overline{g} \lor \overline{u}) \land (g \lor u)\]

\[\begin{align*}
\overline{g} \lor \overline{u} \\
\text{if } u = 1 \text{ then } g = 0 \text{ to satisfy above clause}
\end{align*}\]

\[g \lor u \]

\[\text{if } u = 0 \text{ then } g = 1 \text{ to satisfy above clause}\]
Gate Simulations

1: CNF is simply literal $g$
Gate Simulations

- 0: CNF is simply literal $\overline{g}$
All NP-complete problems!
Where do we go next?

- CS 360/365
  Formalization of Algorithms, full proof of Cook-Levin & much more!
  (Prof. Blais teaching it next term)
- Are there harder problems?
  For sure! See CS 360/365 or more advanced courses
Acknowledgement

Based on

- [Erickson 2019, Chapter 12]
- Prof. Lau’s Lecture 18 notes
  https://cs.uwaterloo.ca/~lapchi/cs341/notes/L18.pdf
References

Cormen, Thomas and Leiserson, Charles and Rivest, Ronald and Stein, Clifford (2009)
*MIT Press*

Dasgupta, Sanjay and Papadimitriou, Christos and Vazirani, Umesh (2006)
Algorithms

Erickson, Jeff (2019)
Algorithms
*https://jeffe.cs.illinois.edu/teaching/algorithms/*

Kleinberg, Jon and Tardos, Eva (2006)
Algorithm Design.
*Addison Wesley*