Complexity IBC028, Lecture 5

H. Geuvers

Institute for Computing and Information Sciences
Radboud University Nijmegen

Version: spring 2023

Outline

Proving that a problem is NP-complete

More **NP**-complete satisfiability problems

Some other **NP**-complete problems

Recap: **P** and **NP**

P :=
$\{A \subseteq \{0,1\}^* \mid \ \exists f, f \text{ polynomial}, x \in A \Longleftrightarrow f(x) = 1\}$

NP :=
$\{A \subseteq \{0,1\}^* \mid \ \exists f, f \text{ polynomial},$
$x \in A \Longleftrightarrow \exists y \in \{0,1\}^*(|y| \text{ polynomial in } |x| \ \wedge \ f(x,y) = 1)\}$

- **P** $=$ the class of polynomial time decision problems.
- **NP** $=$ the class of non-deterministic polynomial time decision problems.
- Property: **P** $\subseteq$ **NP** (Open question: **P** $\stackrel{??}{=}$ **NP**.)

Recap: NP-hard and NP-complete

Definition

A (polynomially) reduces to B, notation $A \leq_P B$ if there is a polynomial function $f : \{0,1\}^* \to \{0,1\}^*$ such that

$$x \in A \Longleftrightarrow f(x) \in B$$

Definition

- $\mathbf{NPH} := \{A \mid \forall X \in \mathbf{NP}(X \leq_P A)\}$
  A is **NP**-hard if $A \in \mathbf{NPH}$.
- $\mathbf{NPC} := \mathbf{NP} \cap \mathbf{NPH}$
  A is **NP**-complete if $A \in \mathbf{NP}$ and A is **NP**-hard.

Theorem

If $B \in \mathbf{NPH}$ and $B \leq_P A$, then $A \in \mathbf{NPH}$.

SAT

SAT is a known **NP**-complete problem. (Proof will be given in Lecture 7.)

- The boolean formulas are built from
  - Atoms, $p, q, r, \ldots$
  - Boolean connectives $\wedge$, $\vee$, $\neg$ (and possibly $\rightarrow$, $\leftrightarrow$, $\bot$, $\top$).
- A formula φ is satisfiable if there is an assignment $v : \text{Atoms} \rightarrow \{0, 1\}$ such that $v(\varphi) = 1$ ("φ is true in v").
- SAT is the problem of deciding whether a boolean formula is satisfiable.
- SAT is in **NP**: the assignment v is the certificate, polynomial in $|\varphi|$, and $v(\varphi) = 1$ can be decided in polynomial time.
- SAT is in **NPH**. This is the famous Cook-Levin theorem, (showing that every problem in **NP** can be reduced to a SAT-problem).

CNF-SAT

CNF-SAT is also **NP**-complete and will be used more often.

- The boolean formulas are built from atoms, $p, q, r, \ldots$ and the Boolean connectives $\wedge$, $\vee$, $\neg$.
- CNF-SAT: satisfiabilty of conjunctive normal forms (CNF):
  - A CNF is a conjunction of clauses
  - A clause is a disjunction of literals
  - a literal is an atom or a negated atom.
- A CNF-formula φ is satisfiable if there is an assignment $v : \text{Atoms} \to \{0, 1\}$ such that $v(\varphi) = 1$ ("φ is true in v").
- CNF-SAT is the problem of deciding if a CNF-formula is satisfiable.
- CNF-SAT is in **NP**: again the assignment v is the certificate.
- That CNF-SAT is in **NPH** will be shown in Lecture 7, and is a direct corollary of the proof of the Cook-Levin theorem.

Proving that a problem is **NP**-complete

How to prove that A is **NP**-complete?

1. Prove that $A \in$ **NP**: give a polynomial algorithm f such that f verifies A with polynomial certificates,
   that is:
   $x \in A \Longleftrightarrow \exists y \in \{0,1\}^*(|y| \text{ polynomial in } |x| \wedge f(x,y) = 1)\}$
2. Pick a well-known decision problem B which you know is **NP**-complete
3. Prove that $B \leq_P A$.

$\leq_3$CNF-SAT is **NP**-complete

Definition

A conjunctive normal form with ≤ 3 literals, $\leq_3$CNF, is

- a conjunction of clauses where
- every clause is a disjunction of at most three literals

$\leq_3$CNF-SAT is the problem of deciding for an $\leq_3$CNF formula whether it is satisfiable.

Theorem

$\leq_3$CNF-SAT is **NP**-hard

Proof

- $\leq_3$CNF-SAT is **NP**: an assignment $v : \text{Atoms} \to \{0, 1\}$ that makes the formula true is the certificate. (Checking is easy.)
- $\leq_3$CNF-SAT is **NP**-hard: We prove CNF-SAT $\leq_P$ $\leq_3$CNF-SAT.

Proof of CNF-SAT $\leq_P$ $\leq_3$CNF-SAT (I)

We define a function f : CNF $\rightarrow$ $\leq_3$CNF such that φ is satisfiable iff $f(\varphi)$ is satisfiable.

The definition of f is based on the following Lemma:

Lemma

$\varphi \wedge \bigvee_{i=1}^{n} \ell_i$ is satisfiable $\iff$
$\varphi \wedge (\ell_1 \vee \ell_2 \vee a) \wedge (\neg a \vee \bigvee_{i=3}^{n} \ell_i)$ is satisfiable, where a is a fresh atom.

Proof. $\Rightarrow$:

Proof of CNF-SAT $\leq_P$ $\leq_3$CNF-SAT (II)

We define a function f : CNF $\to$ $\leq_3$CNF such that φ is satisfiable iff $f(\varphi)$ is satisfiable.

The definition of f is based on the following Lemma:

Lemma

$\varphi \wedge \bigvee_{i=1}^{n} \ell_i$ is satisfiable $\iff$
$\varphi \wedge (\ell_1 \vee \ell_2 \vee a) \wedge (\neg a \vee \bigvee_{i=3}^{n} \ell_i)$ is satisfiable, where a is a fresh atom.

Proof. $\Leftarrow$:

Proof of CNF-SAT $\leq_P \leq_3$CNF-SAT (III)

Definition

Define $f(\varphi)$ by recursively replacing in φ every disjunction $\bigvee_{i=1}^{n} \ell_i$ where $n > 3$ by $(\ell_1 \vee \ell_2 \vee a) \wedge (\neg a \vee \bigvee_{i=3}^{n} \ell_i)$ for a fresh atom a.

- The Lemma (previous slide) proves that φ is satisfiable iff $f(\varphi)$ is satisfiable. Therefore CNF-SAT $\leq_P \leq_3$CNF-SAT and so $\leq_3$CNF-SAT is **NP**-hard. We have already shown $\leq_3$CNF-SAT $\in$ **NP**, so $\leq_3$CNF-SAT is **NP**-complete.
- Also f is polynomial. It doesn't blow up the formula φ: $|f(\varphi)| = \mathcal{O}(|\varphi|)$.
- NB. One could require that all literals in a clause are different. That isn't needed for the definition of f to function properly, but we will sometimes assume that.

3CNF-SAT is **NP**-complete

Definition

A 3CNF is a $\leq_3$CNF where every clause is a disjunction of exactly three literals. 3CNF-SAT is the problem of deciding for an 3CNF whether it is satisfiable.

We prove that 3CNF-SAT is **NP**-complete (Thm. 34.10 of CLRS) by:

- 3CNF-SAT $\in$ **NP**. Again, the assignment is the certificate.
- Showing that $\leq_3$CNF-SAT $\leq_P$ 3CNF-SAT by defining $f : \leq_3\text{CNF} \to 3\text{CNF}$. Choose fresh atoms a, a', p, q.
  - Add clauses $A = a \vee p \vee q$, $a \vee p \vee \neg q$, $a \vee \neg p \vee q$, $a \vee \neg p \vee \neg q$, and similarly A' for a'. (Note: $A \wedge A'$ can only be satisfied if $v(a) = v(a') = 1$.)
  - Replace every clause c with two literals by $c \vee \neg a$.
  - Replace every clause c with one literal by $c \vee \neg a \vee \neg a'$.
- Then φ is satisfiable iff $f(\varphi)$ is satisfiable.

Satisfiability problems that are **NP**-complete

- We have seen that SAT, CNF-SAT, $\leq_3$CNF-SAT and 3CNF-SAT are **NP**-complete. (Proof for SAT, CNF-SAT in Lecture 7.)
- This has been proven by showing that they are in **NP** (easy) and by showing that they are **NP**-hard (the real work).
- For showing **NP**-hardness we have used the following chain of reductions.

$$\text{CNF-SAT} \ \leq_P \ \leq_3\text{CNF-SAT} \ \leq_P \ \text{3CNF-SAT}$$

- Not all satisfiability problem are **NP**-hard! For example, 2CNF-SAT is polynomial.
  (2CNF-SAT is the problem of deciding satisfiability of CNFs where every clause has exactly 2 literals.)

There are also lots of other ("real") problems that are **NP**-complete.

ILP is **NP**-complete (I)

Definition

Integer Linear Programming, ILP is the problem of deciding if a finite set of inequalities with coefficients in $\mathbb{Z}$ has a solution in $\mathbb{Z}$.

Example with 2 variables

$$E := \begin{cases} x_1 + 3x_2 & \geq 5 \\ 3x_1 + x_2 & \leq 6 \\ 3x_1 - 2x_2 & \geq 0 \end{cases}$$

NB. Has solutions in $\mathbb{R}$, but not in $\mathbb{Z}$

Theorem (Exercises 34.5-2 and 34.5-3 of CLRS)

ILP is **NP**-complete

NB. The problem of finding solutions in $\mathbb{R}$ is polynomial!

Radboud University Nijmegen

ILP is **NP**-complete (II)

Theorem

ILP is **NP**-complete

- ILP $\in$ **NP**. A certificate is a tuple of integers $(r_1, \ldots, r_n)$ that we substitue for $x_1, \ldots, x_n$ and check that E holds.
- We show that 3CNF-SAT $\leq_P$ ILP by defining for $\varphi \in$ 3CNF with boolean atoms $x_1, \ldots, x_n$ a set of inequalities E_φ such that φ is satisfiable iff E_φ has a solution in $\mathbb{Z}$.
- Add $x_i \geq 0$ and $x_i \leq 1$ to E_φ.
- For every clause $\ell_1 \vee \ell_2 \vee \ell_3$, add $\ell_1 + \ell_2 + \ell_3 \geq 1$ to E_φ, where we replace negative literals $\neg x_i$ by $1 - x_i$.
- We now have

$$\varphi \text{ is satisfiable } \iff E_\varphi \text{ has a solution (in } \mathbb{Z}).$$

- Concluson: 3CNF-SAT $\leq_P$ ILP and so ILP is **NP**-complete.

Clique is **NP**-complete

Definition

Given an undirected graph $G = (V, E)$ and a number k, is there a clique of size k in G? A clique is a set of points $W \subseteq V$ such that each pair of points in W is connected.

Clique(G, k) is the problem of deciding whether there is clique of size k in G, that is

$$\exists W \subseteq V(\#W = k \wedge \forall u, v \in W(u \neq v \to (u, v) \in E)).$$

Theorem

Clique is **NP**-complete (Theorem 34.11 of CLRS).

- Clique $\in$ **NP**.The certificate is the subset $W \subseteq V$ that forms a k-clique. Checking whether W constitutes a k-clique can easily be done in polynomial time.
- We prove Clique is **NP**-hard by showing 3CNF-SAT $\leq_P$ Clique.

Clique is **NP**-hard (I)

We define f : 3CNF $\to$ Graphs such that $\varphi = \bigwedge_{i=1}^{k} C_i$ is satisfiable iff $f(\varphi)$ has a k-clique. (Assume atoms occurs uniquely in a clause.)

- We write $C_i = \ell_1^i \vee \ell_2^i \vee \ell_3^i$ for each clause in φ ($i = 1 \ldots k$).
- $f(\varphi)$ is a graph with $3k$ vertices; each vertex corresponds with a literal ℓ_p^i ($i = 1, \ldots, k$, $p = 1, 2, 3$) in φ.
- The edges in $f(\varphi)$ are as follows.
  There is an edge between ℓ_p^i and ℓ_q^j iff $i \neq j \wedge \ell_p^i \neq \neg \ell_q^j$.

Claim: if φ has satisfying assignment v, then $f(\varphi)$ has a k-clique.
Proof:
From each clause we choose a literal ℓ_p^i for which $v(\ell_p^i) = 1$.
This gives us a k-clique in the graph $f(\varphi)$.

Clique is **NP**-hard (I)

We define f : 3CNF $\rightarrow$ Graphs such that $\varphi = \bigwedge_{i=1}^{k} C_i$ is satisfiable iff $f(\varphi)$ has a k-clique. (Assume atoms occurs uniquely in a clause.)

- We write $C_i = \ell_1^i \vee \ell_2^i \vee \ell_3^i$ for each clause in φ ($i = 1 \ldots k$).
- $f(\varphi)$ is a graph with $3k$ vertices; each vertex corresponds with a literal ℓ_p^i ($i = 1, \ldots, k$, $p = 1, 2, 3$) in φ.
- The edges in $f(\varphi)$ are as follows.
  There is an edge between ℓ_p^i and ℓ_q^j iff $i \neq j \wedge \ell_p^i \neq \neg \ell_q^j$.

Claim: if $f(\varphi)$ has a k-clique W, then φ is satisfiable.
Proof:

- A k-clique V contains exactly one literal from each clause.
- If $\ell_p^i \in W$, then its negation does not occur in W.
- So a clique W gives us a v : Atoms$(\varphi) \rightarrow \{0, 1\}$ that makes φ true.

VertexCover is **NP**-complete

Definition

Given an undirected graph $G = (V, E)$ and a number k, is there a vertex cover of size k in G?
A vertex cover is a set of points $W \subseteq V$ such that each edge has an endpoint (or both) in W.
VertexCover(G, k) is the problem of deciding whether there is a vertex cover of size k in G, that is

$$\exists W \subseteq V\, (|W| = k \wedge \forall (u, v) \in E (u \in W \vee v \in W)).$$

Theorem

VertexCover is **NP**-complete

Proof. (1) VertexCover $\in$ **NP**.The certificate is the subset $W \subseteq V$ that forms a vertex cover of size k.
(2) We will now prove that VertexCover is **NP**-hard.

VertexCover is **NP**-hard

We prove Clique $\leq_P$ VertexCover.
We define f : Graphs $\rightarrow$ Graphs such that

$G = (V, E)$ has a k-clique $\Longleftrightarrow$ $f(G)$ has a $(|V| - k)$-vertex cover.

Define $f(V, E) := (V, \overline{E})$ where

$$\overline{E} := \{(u, v) \mid u \neq v \wedge (u, v) \notin E\}.$$

Claim: (V, E) has a clique of size k iff $(V, \overline{E})$ has a vertex cover of size $|V| - k$.

Proof.

$$\begin{array}{lcl} W \text{ is a clique in } (V, E) & \Leftrightarrow & \forall (u, v) \in W \times W (u \neq v \rightarrow (u, v) \in E) \\ & \Leftrightarrow & \forall u \neq v ((u, v) \notin W \times W \vee (u, v) \in E) \\ & \Leftrightarrow & \forall (u, v) \in \overline{E} (u \in V \setminus W \vee v \in V \setminus W) \\ & \Leftrightarrow & V \setminus W \text{ is a vertex cover in } (V, \overline{E}) \end{array}$$

3Color is **NP**-complete

Definition

3Color: given an undirected graph $G = (V, E)$, is there a 3-coloring of G, that is, a map $c : V \to \{r, y, b\}$ such that $\forall (u, v) \in E\,(c(u) \neq c(v))$.

Theorem

3Color is **NP**-complete

Proof.

- 3Color $\in$ **NP**.The certificate is the map $c : V \to \{r, y, b\}$. Checking that, for a given c, we have $\forall (u, v) \in E\,(c(u) \neq c(v))$ can be done in polynomial time.
- We prove that 3Color is **NP**-hard by proving 3CNF-SAT $\leq_P$ 3Color. The construction of f : 3CNF $\to$ Graphs will be done on the board, and also see the separate note on the webpage.